SpaceWire 2014

MECHANISM ANTIKYTHERA

6th International Conference
22nd - 26th of September 2014
Athens, Greece

Prof. Steve Parkes, University of Dundee, Scotland
Martin Suess, European Space Agency
Vangelis Kollias, TELETEL, Greece

2014.spacewire-conference.org

esa NASA ROSCOSMOS JAXA teletel UNIVERSITY OF DUNDEE

SpaceWire-2014

Proceedings of the 6th International SpaceWire Conference

Athens 2014

Editors: Steve Parkes and Carole Carrie

Space
Technology
Centre
University of Dundee

SpaceWire-2014

Proceedings of International SpaceWire Conference

Athens 2014

ISBN: 978-0-9557196-7-7

Space Technology Centre
University of Dundee

Preface

These proceedings contain the papers presented at the 2014 International SpaceWire Conference, held in Athens, Greece, between 22nd and 25th September, 2014. The International SpaceWire Conference aims to bring together SpaceWire product designers, hardware engineers, software engineers, system developers and mission specialists interested in and working with SpaceWire to share the latest ideas and developments related to SpaceWire technology. SpaceWire technology is now being used or designed into over one hundred spacecraft, covering science, exploration, Earth observation and commercial applications. High profile missions like James Webb Space Telescope, Astro-H, GAIA, ExoMars, Bepicolombo, Solar Orbiter, Sentinels 1, 2, 3 and 5 precursor, and GOES-R are using SpaceWire extensively. SpaceWire is being used in Europe, Japan, USA, Russia, China, India, and other countries of the World.

The conference covers many different aspects of SpaceWire technology and includes both academic and industrial presentations. Sessions address recent developments of the SpaceWire set of standards, space missions and other applications using SpaceWire, new components, sensors and cables which support the SpaceWire standard; products supporting SpaceWire including onboard equipment, instruments and related onboard software; methods and equipment to aid the test and verification of SpaceWire components, units and systems; and SpaceWire networks, their architecture, configuration, and discovery, as well as "plug and play" concepts, other higher level protocols and related hardware and software design issues.

Technical seminars at the conference covered SpaceWire Missions and SpaceWire Protocols. The SpaceWire Missions tutorial looked at how SpaceWire has been used in several NASA, JAXA and ESA missions. The SpaceWire Protocols tutorial covered several protocols that run on top of SpaceWire. Each protocol was described in detail to provide a good understanding of its purpose and how it works. Part 1 of this tutorial looked at protocols that have been approved or that are in the process of being approved, while part 2 covered new protocols that are in the process of being specified.

The community of engineers working on SpaceWire meet regularly at the SpaceWire Working Group meetings to help with the further development of SpaceWire and related standards and technologies. This group includes engineers from many parts of the World with substantial contributions from Europe, USA, Japan, and Russia. The SpaceWire Conference complements these Working Group meetings with more formal presentations from a wider range of contributors.

There is growing interest in the SpaceFibre which aims to provide multi-gigabit/s network technology for future space flight application like high-resolution multi-spectral imaging and synthetic aperture radar. The number of papers presented at the conference on SpaceFibre and related technologies continues to grow.

The conference committee would like to acknowledge the support and hard work of the many individuals who made International SpaceWire Conference 2014 a reality. First, we thank the authors and the keynote speakers for their contributions. We express our gratitude to the Technical Committee for their assistance in the review process. We thank all people supporting us at Teletel, the Space Technology Centre of the University of Dundee, and the European Space Agency.

The Conference Chairpersons,

Martin Suess, *European Space Agency, The Netherland*
Steve Parkes, *Space Technology Centre, University of Dundee, UK*
Vangelis Kollias, *Teletel, Greece*

Technical Committee

Brice Dellandrea – Thales Alenia, France

Omar Emam - Astrium, UK

Wahida Gasti - ESA, The Netherlands

Sandi Habinc – Aeroflex Gaisler

Hiroki Hihara – NEC, Japan

Christophe Honvault - ESA

Torbjörn Hult - RUAG Space, Sweden

Jørgen Ilstad - ESA, The Netherlands

Paul Jaffe - Naval Research Laboratory, USA

David Jameux- ESA, The Netherlands

Gerald Kempf - RUAG Space, Austria

Clifford Kimmery – Honeywell Inc.

Alexander Kisin - MEI, USA

Robert Klar - South West Research Institute, USA

Jerome Lachaize – Astrium, France

Jennifer Larsen - Aeroflex

Jim Lux - NASA JPL, USA

Masaharu Nomachi – University of Osaka, Japan

Olivier Notebaert - Astrium SAS, France

Steve Parkes - University of Dundee, Scotland, UK

Manuel Prieto - Alcala University, Spain

Paul Rastetter - Astrium GmbH, Germany

Derek Schierlmann - Naval Research Laboratory, USA

Alan Senior - SEA, UK

Yuriy Sheynin - St. Petersburg State University of Aerospace Instrumentation, Russia

Tatiana Solokhina - ELVEES, Russia

Martin Suess - ESA, The Netherlands

Antonis Tavoularis - Teletel

Raffaele Vitulli - ESA, The Netherlands

Takahiro Yamada - JAXA/ISAS, Japan

Takayuki Yuasa – JAXA, Japan

Programme Overview

Monday 22 September

14:00 – 18:00 Registration

15:00 – 19:15 Tutorials of SpaceWire Missions and SpaceWire Protocols

Tuesday 23 September

09:00 – 10:15 Conference Opening / Keynote Presentations (75 min)

10:35 – 12:15 Networks & Protocols Long 1 (100 min)

13:40 – 14:25 Components Short (45 min)

14:25 – 15:10 Missions & Applications Short (45 min)

15:30 – 16:20 Standardisation Long (50 min)

16:20 – 16:45 Test & Verification Long (25 min)

Wednesday 24 September

08:45 – 10:15 Networks & Protocols Short (105 min)

10:35 – 12:15 Components Long (100 min)

13:15 – 14:30 SpaceFibre Long (75min)

14:30 – 16:00 Poster Session (90 min)

Thursday 25 September

09:00 – 09:50 Networks & Protocols Long 2 (50 min)

09:50 – 10:50 SpaceFibre Short (60 min)

11:10 – 12:10 Onboard Equipment & Software Short (60 min)

13:25 – 14:55 Test & Verification Short (90 min)

15:15 – 16:00 Standardisation Short (45 min)

16:00 – 16:25 Missions & Applications Long (25 min)

Programme is subject to change

Tuesday 23 September

Networks & Protocols 1 (Long)

FDIR Techniques for Payload Streaming Applications using SpaceWire-based Networks

Networks and Protocols, Long Paper

Felix Siegle, Tanya Vladimirova
University of Leicester
Leicester, LE1 7RH, United Kingdom

Jørgen Ilstad
European Space Agency / ESTEC
2200 AG Noordwijk, The Netherlands

Omar Emam
Airbus Defence and Space
Stevenage, SG1 2AS, United Kingdom

***Abstract*—This paper is concerned with a novel Fault Detection, Isolation and Recovery (FDIR) methodology for multi-Field-Programmable Gate Array (FPGA) systems. It features an embedded hardware platform, which supports adaptive redundancy whereby redundant processor instances can be distributed over multiple FPGA devices. This is achieved by utilising a Network-on-Chip (NoC), which is heavily based on SpaceWire.**

***Index Terms*—FDIR, Majority Voting, Redundancy, SoCWire, Spacecraft Electronics, SpaceWire, SRAM-based FPGAs**

I. Introduction

Modern approaches to satellite payload data processing demand increased processing capabilities. Ideally, the payload data can be processed in real time while being streamed from an on-board sensor, e.g. a camera to a mass memory device. The processing data path may contain several processor nodes connected in series.

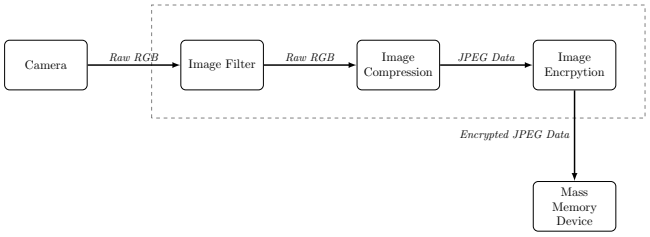

Figure 1: Example for an image-processing pipeline.

An example would be an image processing pipeline, as outlined in Figure 1, in which video data is first filtered, then compressed and finally encrypted. To make such a processing pipeline adaptable in terms of functionality and reliability, the different processing steps can be implemented on reconfigurable Field-Programmable Gate Arrays (FPGAs). Since fast hardware implementations of the processing steps can be rather resource demanding, techniques are necessary to also exploit multi-FPGA systems. An example for such a system is the Dynamically Reconfigurable Processing Module (DRPM) developed by University of Brunswick, Germany and Airbus Defence and Space, UK [1], shown in Figure 2. This hardware development platform comprises a scalable number of payload data processing units with two reconfigurable SRAM-based Virtex-4 FPGAs and one LEON3 microprocessor per unit. The DRPM platform is being used in a research project carried out at the University of Leicester, which is aimed at the development of Fault Detection, Isolation and Recovery (FDIR) techniques for payload streaming applications implemented on SRAM-based FPGAs.

A novel FDIR technique called *Distributed Failure Detection* aimed at utilising multi-FPGA systems more efficiently was presented in [2] that makes use of a SpaceWire-based Network-on-Chip (NoC). However, the technique was not capable of dealing with the asynchronous network streams, which usually occur when the network nodes are located in different clock domains. By introducing a stand-alone failure detector, which is able to synchronise incoming data streams automatically, the technique was further developed in [3]. Based on the results of a Failure Mode and Effects Analysis (FMEA), the detector module is designed in such a way that it can handle typical failure modes occurring in network architectures.

In this paper, an upgraded version of this technique is presented, in which the failure detector modules are embedded into routing switches. By doing so, it is finally possible to detect failures in redundant asynchronous network streams, which are provided by network nodes that can be arbitrarily placed within the network. In addition, a novel scheme for the data synchronisation after failure recovery is discussed and a new evaluation of our technique in terms of power, area and performance is presented.

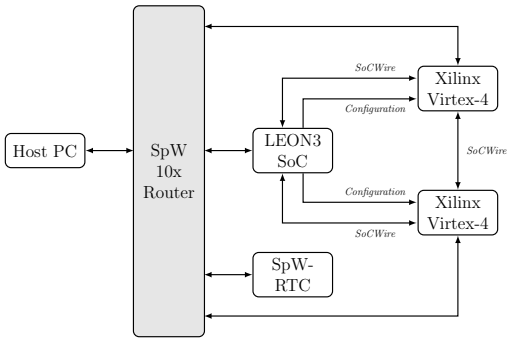

Figure 2: Block Diagram of the DRPM Demonstrator Platform.

The paper is structured as follows. In Section II, the architecture of the basic building blocks, the so-called stream processors, is described. In Section III, an example is given of how the Distributed Failure Detection technique allows the distribution of redundant processors throughout the network. In Section 0, results of the conducted FMEA [3] are presented. Then, the designs for a majority voter module and a broadcast mechanism, both based on the results of the FMEA and now integrated into routing switches, are described in Section V. In Section VI, a novel data resynchronisation scheme for freshly repaired stream processors is discussed. Section VII evaluates the power, performance and area overhead of the proposed techniques before Section VIII concludes the paper.

II. Stream Processor Architecture

In the proposed FDIR framework, different processing steps are executed by dedicated stream processors, which can process incoming data streams independently.
A typical architecture of such a stream processor is shown in Figure 3. An Intellectual Property (IP) core of the desired functionality, e.g. for data compression, encryption or filtering, is embedded into a wrapper.

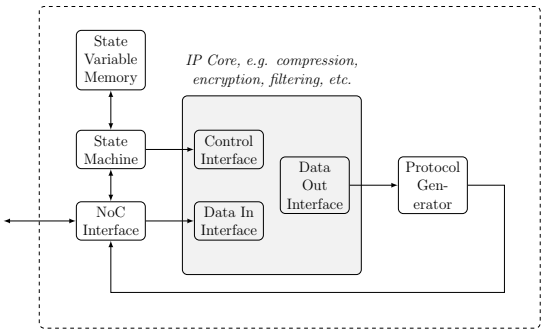

Figure 3: Stream Processor Architecture.

This wrapper comprises a NoC interface for the data exchange, some state machine logic and a memory for state variables. The state machine interprets input control words whereas input data words are directly fed into the IP core. An additional memory holds all variables that are necessary to configure the IP core. If the processing chain uses a specific network protocol, a protocol parser and/or protocol generator may be added to the input and output of the core (here, the CCSDS Space Packet Protocol [4] was adopted). Partitions, which can host such a stream processor, are implemented on SRAM-based FPGAs. They are connected to a packet-switched, flow-controlled NoC and can be reconfigured during operation by means of dynamic partial reconfiguration.

The here presented work is based on a NoC implementation called SoCWire [5]. SoCWire is a minimal version of SpaceWire. In this protocol, each network packet may start with a logical address (that is typically used for routing purposes) and is terminated by an End of Packet (EOP) marker. Every time the receive buffer has space for eight more characters, the receiving node sends out a Flow Control Token (FCT). Therefore, the receiving node can apply backpressure to a communication channel, i.e. it can force the source node to freeze by simply ceasing the transmission of further FCTs.

III. Novel Distributed Failure Detection Method

With our *Distributed Failure Detection* methodology, first outlined in [2], failure detectors become part of a network. This novel approach allows the free distribution of redundant processors throughout the network because the output of each processor can be routed to any failure detector, independent of its location in the network. The network can even span over several FPGAs, i.e. links may connect network nodes on-chip but also off-chip.

The possibility to place redundant stream processors on several FPGAs has many advantages, for example in cases where the chip area of one FPGA is not sufficient to host a full fault-tolerant design.

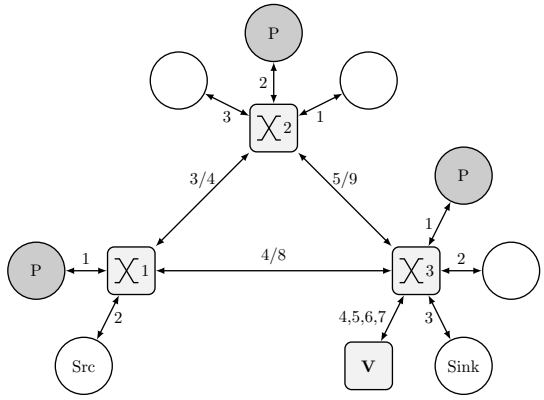

Figure 4: Distributed Failure Detection example.

An example network topology is shown in Figure 4. Several partitions (circles) are interconnected via routing switches. A processor has been triplicated and the resulting instances (grey circles) are placed on some of these partitions. Say, data is sent from a source node *Src* to the processor and the processor sends the processed data to sink node *Sink*. As the processor is triplicated, the data must first be broadcast within the routing switches. For instance, routing switch 1 broadcasts the packets to output port 1, 3 and 4. In switch 2 and 3, the packets are then routed to the other two redundant instances. After processing, the resulting packets are routed to the failure detector, which in this case is the voter module **V**, connected to routing switch 3. Finally, the output of the voter is routed to the sink node.

Typically, the network packets do not arrive simultaneously at the redundant processors. The latency between each redundant processor and the failure detector may differ too. In addition, the partitions might be implemented in different clock domains. As a result, the voter module must be able to deal with asynchronous network streams.

IV. Failure Mode and Effects Analysis

In a recent Failure Mode and Effects Analysis [3], it was found that two types of failure modes must be expected, those, which affect the payload of network packets (i.e. the

application data), and those, which affect the network traffic itself. A summary of the FMEA results is shown in Figure 5.

Case (A): Typical operation. The network packets are identical but may arrive at different points in time at the voter module. This non-synchronicity can be handled by exploiting the flow control of the network architecture. The voter module applies backpressure to the network channels that already received some data until at least one data character has arrived in all slots.

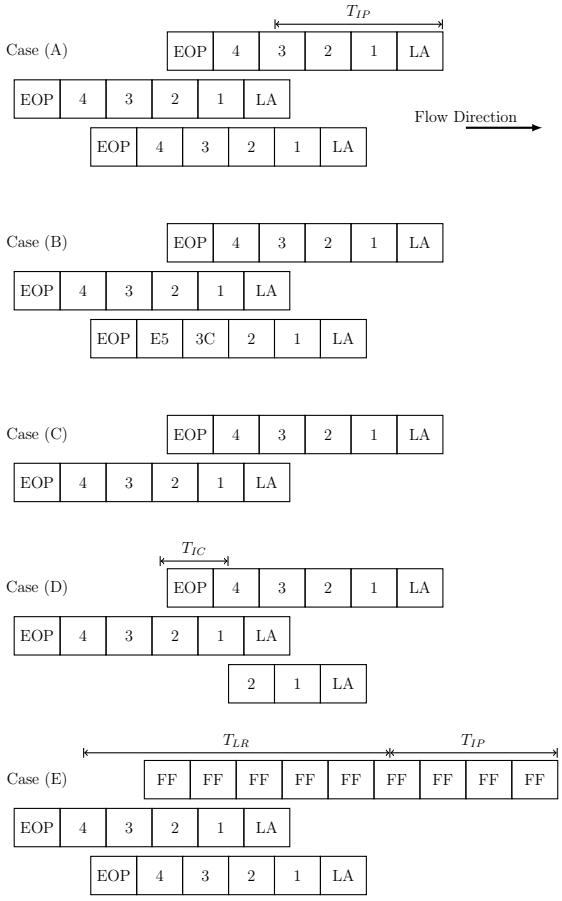

Figure 5: Different observable failure modes in network streams.

Case (B): The network packets have an identical structure, i.e. the network protocol is faultless, but their payload differs due to a failure in the application. In most applications, this case will be the most observed one because the probability of a failure in the application is usually larger than the probability of a failure in the network related components. This failure mode can be detected by a voter mechanism that compares the (synchronised) network streams character by character.

Case (C): One of the network packets does not arrive at all. It seems as if the corresponding processor became faulty and ceased the transmission for some reason. This case can be handled by a timeout mechanism with a timeout value T_{IP}, hereafter also referred to as *Inter-Packet Timeout*, which is triggered once the first redundant packet arrives in one of the *slots* (i.e. a receive buffer assigned to a particular processor) of the voter module. If the timeout elapses and one of the redundant network packets has not arrived in its slot, this slot is marked as faulty.

Case (D): The transmission of one of the network packets suddenly stops before the EOP marker is reached. This case can be handled by a second timeout mechanism with a timeout value T_{IC}, hereafter also referred to as *Inter-Character Timeout*, which is always retriggered when data character(s) are available in some slots but not in others. If the timeout expires, it must be assumed that the processor associated with the still empty slot suddenly stopped the transmission and thus this slot is marked as faulty.

Case (E): One processor becomes a *babbling idiot* and is transmitting undefined data at undefined points in time. Dealing with this case can be problematic if the data from the babbling idiot arrives much earlier than the data from the two other healthy processor instances. Then, the Inter-Packet Timeout would expire first and the two healthy slots would be spuriously marked as faulty (actually all slots would be marked as faulty because further voting is not possible). As it is rather unlikely that two processors fail at the same time, this case is handled by assuming that the early packet is wrong, i.e. the corresponding slot is temporarily marked as faulty. Then, a second timeout value T_{LR}, hereafter also referred to as *Last Resort Timeout*, is started. If the packets from the healthy processors arrive within this timeout period, no further action is required. If they do not arrive, however, all slots must be marked as faulty.

Aside from the aforementioned failure modes, another mode must be considered when broadcasting data. If a processor becomes faulty, it may block incoming traffic. This case can be handled by using a non-blocking broadcast mechanism that comprises a *Broadcast Timeout*. If one of the processors blocks incoming data throughout the timeout period, it is afterwards excluded from the broadcast until the end of the current packet transmission.

V. Voter and Broadcast Design

A. Addressing Scheme

For this work, a NoC routing switch has been prototyped that comprises a logical address table similar to the one known from SpaceWire devices like the SpaceWire 10X Router ASIC [6]. The table can be remotely programmed using a simple protocol and assigns one or more physical output ports to a specific logical address. If more than one port is assigned to an address, network packets with this address are broadcast to all output ports (this behaviour is in contrast to some SpaceWire devices that implement adaptive routing).

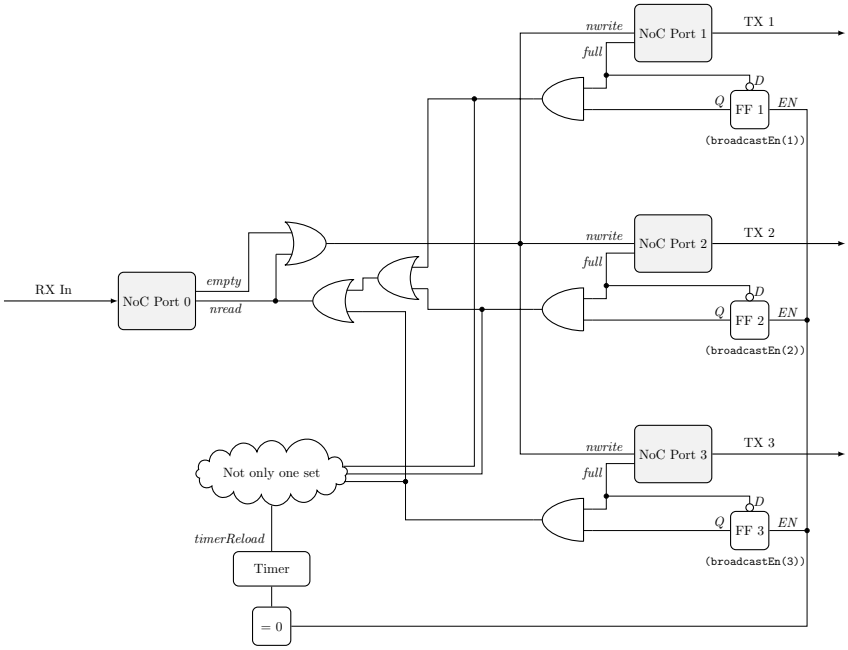

Figure 6: Circuit diagram of the broadcast mechanism.

The proposed concept implements a double addressing scheme. In the targeted streaming applications, network nodes are arranged in a processing pipeline and hence each node sends its data to only one remote node. To simplify the implementation, each network node is fixed to one logical address, i.e. instead of assigning a logical address to the remote node the logical address is actually assigned to the source node. This addressing technique was chosen because it allows a network node to be not aware of the logical address of its successor node and it is therefore sufficient to hardcode the logical address into the hardware module. However, there is one exception: The data resynchronisation mechanism, see Section 0, necessitates that a network node is also directly accessible. Thus, a second logical address, also referred to as *synchronisation address*, is assigned to each node.

The FMEA revealed that babbling idiots must be expected. In theory, such a node can send out any random data but in practice it is more likely that these packets are filled with zeroes or ones due to stuck-at faults. Therefore, to isolate such failures, the logical addresses `0x00` and `0xFF` are forbidden and any packet carrying one of these addresses is automatically spilled within a routing switch.

Aside from babbling idiots, addressing failures are especially critical since packets can potentially block essential parts of the network. Thus, we ensure that all used logical addresses have at least a hamming distance of 2. By doing so, a Single Event Upset (SEU) in an address register can only create an invalid address but not another valid address. Since packets with invalid addresses are spilled within the routing switches, such address failures are successfully isolated too.

B. Non-blocking Broadcast Mechanism

Broadcasting data to redundant stream processors is done in a distributed manner within the routing switches. As mentioned in Section 0, the broadcast mechanism must be of non-blocking nature to handle faulty processors that block incoming network traffic.

A conceptual circuit diagram of the non-blocking broadcast mechanism is shown in Figure 6. Say, a network packet arrives at port 0 and its logical address is assigned to physical port 1, 2 and 3. Then, the broadcast mechanism will transfer a data character to port 1, 2 and 3 if the receive buffer of port 0 is not empty and all transmit buffers of port 1, 2 and 3 are not full. This is done by using the handshake signals `full`, `empty`, `nread` and `nwrite`:

```
minOneFull  :=(full(1) and broadcastEn(1))or
              (full(2) and broadcastEn(2))or
              (full(3) and broadcastEn(3))
nwrite(1:3) := empty(0) or minOneFull
nread(0)    := minOneFull
```

To tolerate potentially faulty stream processors, a Broadcast Timeout mechanism is used which is always active if one and only one output port of all active output ports is full:

```
timerReload := true if not
only_one_set(full(1:3) and broadcastEn(1:3))
```

If this timeout elapses, it is assumed that the stream processor, which is associated with the blocking output port, is faulty. Then, the output port is removed from the current broadcast round by setting its `broadcastEn` flag to zero. As a result,

the remaining redundant stream processors receive input data, although some additional latency, equal to the broadcast timeout period, must be expected.

C. Voting Mechanism

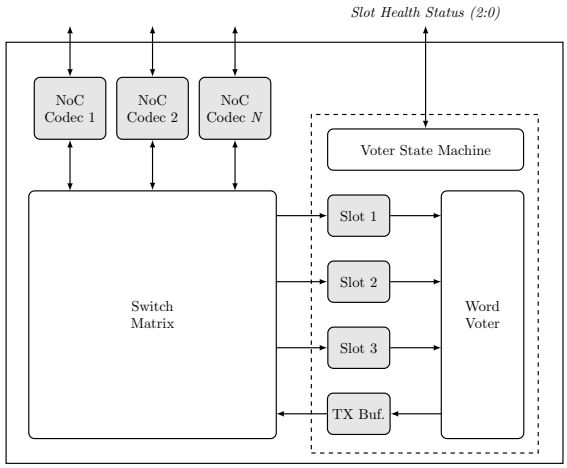

Figure 7: Voter module integrated into NoC routing switch.

The voter module that is used as failure detector is embedded into a NoC routing switch as outlined in Figure 7. It comprises three receive buffers, also referred to as slots, a transmit buffer, a combinational majority word voter and some sequential state machine logic. The voter module is connected to an external supervisor (here a LEON3 microprocessor) via bidirectional *Slot Health Status* flag signals. If one of the slots is detected to be faulty, the supervisor will initiate a recovery procedure of the corresponding processor. In the meanwhile, the voter module automatically degrades to a comparator module. Once the faulty stream processor is repaired, the supervisor can instruct the voter module to reintegrate the freshly recovered slot by simply updating the health status.

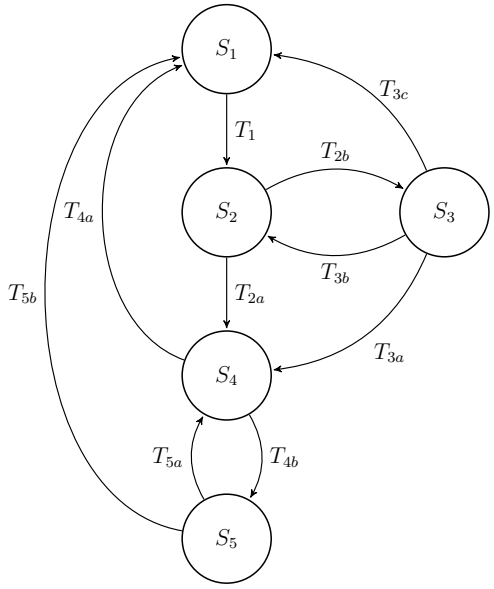

Figure 8: Voter module state diagram.

The state diagram of the voter module comprises five states as shown in Figure 8:

State S_1: The state machine remains in this idle state until the first character of the first redundant network packet arrives in one of the slots. Then, it moves via transition T_1 to state S_2.

State S_2: This is the synchronisation state, in which the state machine awaits all other redundant packets to arrive at the voter module. While being in this state, the Inter-Packet Timeout is active. Two possible exit conditions can be true: If all active slots received some data, the state machine moves via T_{2a} to state S_4. If one or two slots miss data, the state machine moves via T_{2b} to state S_3.

State S_3: First, this state determines which one of the aforementioned exit conditions is true. If two slots received data, the third slot without data is now marked as faulty and the state machine moves via T_{3a} to state S_4, i.e. the voter module just degraded automatically to a comparator and continues its normal operation. If only one slot received data, however, it is assumed that a babbling idiot sent this data and the slot is marked as faulty. Then, the Last-Resort Timeout is started and the state machine moves via T_{3b} back to state S_2, giving the two missing packets some additional time to arrive at the voter module. If they do arrive within this timeout period, the state machine moves via S_3 to normal operation state S_4 (now working as a comparator). If they do not arrive, the voter module cannot continue its operation (which is again determined in state S_3) and the state machine moves back to its idle state S_1.

State S_4: The state machine remains in this state during normal operation. The redundant network streams are synchronised and the state machine sends character by character through the combinational majority voter, or, depending on the mode, through the comparator circuit as long as data is available in all active slots. Every time data is available in some slots but not in others, the Inter-Character timeout mechanism is triggered. Two possible failure modes can occur in this state: Either the Inter-Character timeout expires or a voting / comparing mismatch occurs. In both cases, the state machine moves via transition T_{4b} to state S_5. If none of these failure modes occurs and an EOP marker is received, the state machine moves via transition T_{4a} back to its idle state S_1.

State S_5: This state determines which failure mode occurred in state S_4 and reacts accordingly. If the Inter-Character timeout elapsed, the slot that misses data is marked as faulty. In case of a voting mismatch, the slot that contains wrong data is marked as faulty, in case of a comparing mismatch both involved slots are marked as faulty. Two possible exit conditions can be true: If two slots are still functional, the voter module continues its work as a comparator in state S_4. Otherwise, the state machine moves back to its idle state S_1 via T_{5b}.

VI. Data Resynchronisation

An often-mentioned problem in connection with modular redundancy is the required data resynchronisation between the

redundant module instances after a module has been successfully repaired. Fortunately, typical payload processing applications do not depend on too many state variables. The number of state variables required for the initialisation of a processor after reset is often limited to a handful of configuration and feedback variables. For instance, an image compression core might need a variable storing the compression quality and one storing the image line width. Another example could be an encryption algorithm used in some feedback mode. Here, a feedback variable storing the last cipher text might be needed to initialise the freshly repaired processor.

Assuming that the processing chain has more network bandwidth available than the input data stream or alternatively, that well dimensioned buffers are available in the network, the data processing could be stopped for a short time period in which the state variables are shared between the currently functional stream processors and the freshly repaired stream processor. We propose to use the already available resources in the here presented FDIR methodology to accomplish this task.

As can be seen in Figure 3, each stream processor already comprises a state variable memory, which stores all required initialisation variables externally to the embedded IP core. Thus, it is sufficient to dump theses variables over the network to the freshly repaired processor, which can then store this variable set in its own state variable memory. To increase the reliability of this mechanism, the state variables of the two functional processors could be first compared before the freshly repaired processor registers them. Since each voter module also works as comparator, an elegant solution would utilise the voter module for this task. Consider the example shown in Figure 4. Say, the processor connected to routing switch 1 has been just repaired and needs to be updated with initialisation variables. The other two processors could stop the data processing after finishing the processing of the current block of data and send their state variables to voter module **V**. The voter module could compare the network packet, which contains the state variables, and forward all identical state variables to the freshly repaired processor which then updates its own memory.

However, there are two main issues that must be taken into account:

1. The two functional stream processors are not running synchronous and therefore a synchronous request to dump the state variable memory could lead to situations where one processor is dumping newer and hence other variables than the second processor.
2. The two functional stream processors must stop any data processing until the freshly repaired instance has updated its own state variable memory and resumed its operation. Otherwise, the shared variables might be already invalid once they become active in the freshly repaired processor.

To solve the first problem, the request to dump the state variables must be injected into the input data stream. For instance, a small hardware module placed at the front of the processing chain could send out a small synchronisation request packet containing the synchronisation address of the freshly repaired stream processor. This request packet would traverse the network like the regular network stream. Relative to this input data stream it would arrive at the same bit position and thus it could be ensured that the still functional processors receive this request packet when they are both in the exact same state. Then, the functional processors could bundle their state variables together with the received synchronisation address into a synchronisation packet, which is attached to the output data stream.

At some point, the voter module would receive the redundant synchronisation packets. The voter module is able to detect this special kind of packet and would move into a resynchronisation mode. While in this mode, the aforementioned second issue could be simply solved by applying backpressure to the slots associated with the two functional processors. In other words, after receiving the synchronisation packets, no more data characters are taken out from the slot buffers and therefore the functional processors would be forced to stop the data processing (with some latency as the buffers in the network path would fill up first). In addition, the voter module could start a *Data Synchronisation Timeout*. The timeout period must be chosen wisely to (i) give the freshly repaired module enough time to update its state variable memory but (ii) also take the buffer sizes and bandwidths within the network into account. The voter module would then send the synchronisation packet to the synchronisation address and thus to the freshly repaired stream processor. If no comparison mismatch occurred, the voter module would go into a special wait state afterwards.

A short time later, the synchronisation packet would arrive at the freshly repaired processor, which would update its state variable memory and resume operation. Once the first data is processed, its first output packet would arrive at the voter module, which is still in its wait state applying backpressure to the other two redundant processors.

Now, the voter module would reintegrate the freshly repaired stream processor because the first output packet of the freshly repaired processor would be identical to the output packets of the other two redundant processors. It would stop the backpressure and resume normal operation. However, if the Data Synchronisation Timeout elapsed, it would be assumed that something went wrong during resynchronisation. In this case, the backpressure would be released and normal operation would be resumed without reintegrating the freshly repaired processor.

VII. Power, Area and Performance Overhead

Aside from the practical benefits offered by the here proposed FDIR methodology, it is of interest how well it performs in terms of power, chip area and clock frequency, compared to classic mitigation approaches for SRAM-based FPGAs.

Typically, Triple Modular Redundancy (TMR) is applied at a very low level by triplicating FPGA building blocks and inserting bit voters into the netlist of a circuit. In the following, our approach is compared to this mitigation technique. The proof-of-concept system used here comprises a stream processor that implements JPEG image compression.

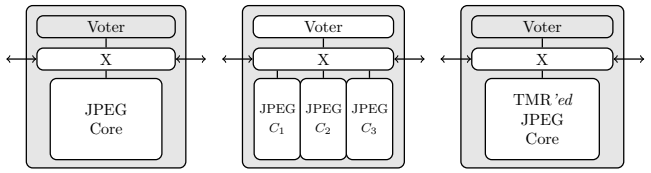

Figure 9: Failure Masking and Detection Techniques. From left to right: (a) No Redundancy, (b) Modular TMR as proposed here, (c) Classic Netlist-TMR approach.

Three scenarios, as depicted in Figure 9, are investigated. In scenario (a), a single JPEG stream processor is connected to the NoC routing switch; the voter module was removed (grey). In scenario (b), three redundant JPEG stream processors are connected to the NoC routing switch and the voter module is implemented. In scenario (c), Netlist-TMR is applied to a single JPEG stream processor by using the Xilinx TMRTool [7]; the voter module was again removed. The used FPGA is a Xilinx Virtex XC4VSX55-10. In the following, the power measurements were conducted using a Tektronix TDS5054B oscilloscope that was connected to a high precision current probe TCP312. All results are post-place & route.

TABLE I. REQUIRED CHIP AREA.

Scenario	Slices	RAMBs	DSP48s
(a) No redundancy	9,953 (40%)	90 (28%)	10 (1%)
(b) Modular TMR	21,852 (88%)	260 (81%)	30 (5%)
(c) Netlist-TMR	20,909 (85%)	258 (80%)	30 (5%)

In Table I, the required chip area is listed for the three scenarios in terms of Slices, Block RAMs (RAMBs) and Digital Signal Processing Blocks (DSP48s). The Modular TMR approach used by the proposed FDIR framework requires slightly more slices and one additional Block RAM compared to the classic Netlist-TMR approach.

TABLE II. PERFORMANCE RESULTS.

Scenario	Min. Period [ns]	Max. Freq. [MHz]
(a) No redundancy	9.04	110.66
(b) Modular TMR	9.52	105.02
(c) Netlist-TMR	10.10	98.97

Table II lists the maximum clock frequency, respectively the minimum period for the three scenarios. Both TMR approaches perform naturally worse than systems to which no redundancy is applied. Main reason for scenario (b) is the fact that the three partitions hosting the stream processors are area constraint, which limits the capabilities of the place & route tool. In scenario (c), the critical path length is increased because many one-bit voters are inserted into the netlist. The Modular TMR approach used by the proposed FDIR methodology performs much better than the classic Netlist-TMR approach. This is especially an advantage for payload data processing systems where performance is more important than area overhead.

TABLE III. POWER CONSUMPTION

Scenario	Relative Power [W]
Not configured	0.00
(a) No redundancy	1.01
(b) Modular TMR	1.87
(c) Netlist-TMR	2.08

The figures in Table III show that the proposed Modular TMR approach also consumes less power than the Netlist-TMR approach, another beneficial aspect for payload data processing systems with power demanding circuits.

VIII. CONCLUSIONS

The Distributed Failure Detection technique in its final development stage offers many advantages compared to classic mitigation approaches for SRAM-based FPGAs, as presented above. On the one hand, the methodology is adaptive, i.e. redundancy can be added or removed during flight to either increase the system availability or the power consumption of the system. On the other hand, by utilising a NoC as communication architecture, redundant stream processors can be distributed over several FPGAs and hence, multi-FPGA systems can be utilised more efficiently. Although data resynchronisation after repair is a serious issue, a novel resynchronisation scheme was proposed, which results in low implementation complexity and area overhead. Furthermore, it was shown that for some applications, the modular redundancy approach performs better than the classic Netlist-TMR approach in terms of power consumption and maximum clock frequency while the required chip area is only slightly increased.

The FDIR methodology is not technology-dependent and could be applied in a similar way to other systems consisting of multiple processing elements by choosing, for instance, SpaceWire or SpaceFibre as the network architecture.

ACKNOWLEDGMENT

Sponsorship from ESA under the NPI Programme, Airbus Defence and Space, UK and the University of Leicester is gratefully acknowledged.

REFERENCES

[1] F. Bubenhagen, B. Fiethe, J. Ilstad, H. Michalik, P. Norridge, B. Osterloh, W. Sullivan, and C. Topping, “Enhanced Dynamic Reconfigurable Processing Module for Future Space Applications,” in International SpaceWire Conference. International Space Wire Conference, 2010, pp. 475–482.

[2] F. Siegle, T. Vladimirova, O. Emar, and J. Ilstad, “Adaptive FDIR Framework for Payload Data Processing Systems using Reconfigurable FPGAs,” in NASA/ESA Conference on Adaptive Hardware and Systems, 2013.

[3] F. Siegle, T. Vladimirova, O. Emar, and J. Ilstad, “New Voter Design Enabling Hot Redundancy for Asynchronous Network Nodes,” in NASA/ESA Conference on Adaptive Hardware and Systems, 2014.

[4] Consultative Committee for Space Data Systems, “Space Packet Protocol”. Blue Book CCSDS 133.0-B-1, 2003.

[5] B. Osterloh, H. Michalik, B. Fiethe, and K. Kotarowski, “SoCWire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications,” in NASA/ESA Conference on Adaptive Hardware and Systems, 2008, pp. 51 – 56.

[6] C. McClements, S. Parkes, and G. Kempf, “SpW-10X Space Wire Router”. User manual issue 3.4, 2008.

[7] Logic Design: TMRTool. [Online]. Available: http://www.xilinx.com/ise/optional prod/tmrtool.htm

SPACEWIRE TIME DISTRIBUTION PROTOCOL IMPLEMENTATION AND RESULTS

SpaceWire Networks and Protocols / SpaceWire Standardization, Long Paper

Anandhavel Sakthivel, Jonas Ekergarn, Daniel Hellstrom, Sandi Habinc
Aeroflex Gaisler AB
Kungsgatan 12, SE-411 19 Gothenburg, Sweden
anand@gaisler.com ekergarn@gaisler.com
daniel@gaisler.com sandi@gaisler.com

Martin Suess
European Space Agency
Keplerlaan 1, 2220AG Noordwjik ZH, Netherlands
martin.suess@esa.int

***Abstract*—Aeroflex Gaisler has developed, under European Space Agency (ESA) contract 4000104519, a draft ECSS protocol for the transmission and synchronization of CCSDS Unsegmented Code (CUC) time in SpaceWire networks. The working name of the protocol is "Time Distribution Protocol". Apart from transmission and synchronization of time across the SpaceWire network, the protocol also provides guidelines to achieve highly accurate time synchronization by mitigating jitter and latency affecting SpaceWire Time-Code transmission in a SpaceWire network. The protocol also provides guidelines for correcting clock drift appearing in local SpaceWire nodes. A prototype implementation of the protocol was performed and analyzed for intended functionality. The implementation of jitter and drift mitigation is based on a simple time interval measurement of incoming SpaceWire Time-Codes using local clock. Statistical information is gathered, which is then used to calculate an average correction value that is applied to modify a frequency synthesizer which provides inputs for the time generation. By controlling the frequency synthesizer the time is maintained stable without drift arising from oscillator or crystal used to generate the local clock. Distributed interrupts are used for latency measurement between two nodes in a SpaceWire network. Time-stamping of reception and transmission of distributed interrupts provides the values needed to calculate latency. The calculated latency value is used for correcting the time maintained in the system. By performing CCSDS Unsegmented Code (CUC) transmission and latency, jitter, drift mitigation a stable time keeping in a system is achieved.**

***Index Terms*— SpaceWire, Network, Time-code, Jitter, Latency, Distributed Interrupts.**

I. INTRODUCTION

Time synchronization in spacecraft is becoming increasingly important. Traditionally time synchronization has been done via dedicated signals or via deterministic on-board buses. With the advent of SpaceWire links and router switches being used for critical control functions, the need for accurate time synchronization via this network has arisen. The SpaceWire protocol provides rudimentary time-code transmission for time synchronization but the transmission and reception of time codes suffer from time distribution delay (or latency) and jitter in a system. Further the time is generated using the local clock available in a system. The oscillator or crystal used to generate the local clock may not only have an incorrect frequency, it may also experience frequency variations over time (drift), which will directly affect the time keeping in a system [1].

The aim of this implementation is to synchronize the time in an initiator to the time in a target. The initiator being the master maintains its time and no changes is made to the master time after setting it up with an initial reference. The target time is synchronized by using the time and SpaceWire Time-Code transmitted from initiator. The time mentioned here is based on CCSDS Time Code (time message) and CCSDS Unsegmented Code (CUC) is used in this implementation [2]. The SpaceWire Time-Code is mapped from this CUC time and the structure of the SpaceWire Time-Code is as per the SpaceWire standard [3]. The time synchronization protocol defined is briefly explained in the next paragraph.

The Time Distribution Protocol provides the means for transferring time of initiator to targets and for providing a synchronization point in time. The time is transferred by means of an remote memory access protocol (RMAP) write command carrying a CCSDS Time Code (time message) [4]. The synchronization event is signaled by means of transferring a SpaceWire Time-Code. The transfer of the SpaceWire Time-Code is synchronized with time maintained by the initiator. To distinguish which SpaceWire Time-Code to use for synchronization, the value of the SpaceWire Time-Code is transferred from initiator to target by means of an RMAP write command prior to the actual transmission of SpaceWire Time-Code itself. When there is more than one target the CCSDS Time Code needs to be transferred to each individual target separately [5].

Accurate time synchronization through SpaceWire should enable and promote the use of SpaceWire for critical control functions on-board a spacecraft. It will also allow

reduction of the number of on-board buses required in on-board systems.

This paper will discuss about the details of protocol implementation, methodology to achieve highly accurate time synchronization, prototype hardware developed, testing and validation and provides results of accuracy in time synchronization achieved.

II. Implementation

The VHDL IP core (named SPWTDP) developed implements the draft Time Distribution Protocol (TDP) [4]. As shown in Fig. 1 the SPWTDP sends and receives SpaceWire Time-codes and distributed interrupts by interacting with a SpaceWire interface. The SpaceWire interface is also responsible for RMAP command processing. Figure 1 also explains the complete system which can act as initiator or target.

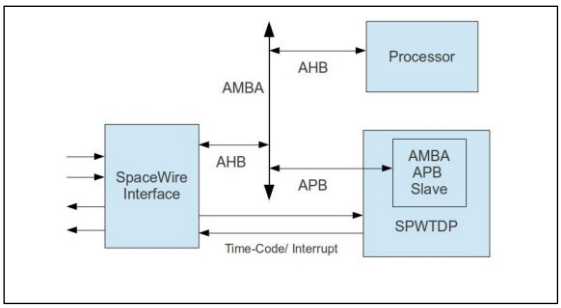

Fig. 1. SPWTDP IP core with a SpaceWire Interface

A. Initiator

The initiator which acts as time master is the only node capable of sending time message and SpaceWire Time-Codes in a network. There can be only one initiator in a SpaceWire network during a mission phase. The initiator also requires SpaceWire link interface implements a RMAP initiator capable of transmitting RMAP commands and receiving RMAP replies.

1) The initiator performs the following task

a) Send SpaceWire Time-Codes. The SpaceWire Time-Codes are provided by the SPWTDP component and transmission of those codes to targets should be performed by a SpaceWire interface.

b) Send and receive distributed interrupts

c) Transmission of time messages using RMAP

d) Latency measurement and transmission of latency value using RMAP.

B. Target

The target should implement RMAP target and capable of receiving RMAP commands and transmitting RMAP replies. There can be one or more targets in a SpaceWire network.

1) The target performs the following task

a) Reception of SpaceWire Time-Codes. The SpaceWire Time-Codes sent from initiator are received by SpaceWire interface and provided to SPWTDP component in target.

b) Reception of time messages through RMAP

c) Qualification of received time messages using SpaceWire Time-Codes

d) Initialization and Synchronization of received CCSDS Time Codes with local time in SPWTDP component

e) Latency measurement and correction

f) Jitter and drift mitigation

C. Generation of local time from local clock

The local time counter (time) is implemented complying with the CUC T-Field. The counter is incremented on the system clock only when enabled by the frequency synthesizer. The binary frequency required to determine the counter increment is derived from the system clock using a frequency synthesizer. The frequency synthesizer is incremented with a pre-calculated increment value, which matches the available system clock frequency. The frequency synthesizer generates a tick every time it wraps around, which makes the local counter to step forward with the pre-calculated increment value. The output of frequency synthesizer is used for enabling the increment of local counter. The increment rate of the local time counter and frequency synthesizer counter is set according to the system clock frequency. The frequency synthesizer increment value is calculated as in Eq. 1, where FT_W - Fine time width of the CUC T-Field, FS_W - Frequency Synthesizer width, FS_{INC} Frequency Synthesizer increment value, F - Frequency of the system clock.

$$FS_{INC} = ((2 \wedge FS_W) * (2 \wedge FT_W)) / F \quad (1)$$

Both the initiator and target will have its respective local time and frequency synthesizer counters. After an initial value the initiator counters remain constant but the target counters are varied to achieve time synchronization (the variations are explained in detail later in this paper).

D. Generation of SpaceWire Time-Codes

SpaceWire Time-Codes are continuously transmitted from an initiator node (time master) to all target nodes. The transmission of the SpaceWire Time-Code is synchronized with the local time counter in the initiator node. The six bits of the Time-Code time information corresponds to six bits of the local time counter. The local time bits with lower weights than the size bits mapped to Time Code time information bits are all zero at the time of SpaceWire Time-Codes transmission.

E. Initialization and synchronization of target through RMAP

The Local time available in an initiator is transmitted to synchronize time across a SpaceWire network. The initiator

transfers time message using RMAP across the SpaceWire network and the target extracts the time message. The Time message transmitted using RMAP should be an exact mapping of the command field available in the SPWTDP component [5]. The Time message transmitted writes the command field available in target. Control register available in command field specify whether the target should be initialized or synchronized, at which SpaceWire Time-Codes it should happen (synchronization event) and details of coarse and fine time available in the time message.

In target, the command field will contain the time message when it is written by the initiator through RMAP. When the control register with a Time-Code value in command field matches with a received SpaceWire Time-Code then initialization or synchronization will occur to the local time counter available in the target SPWTDP component. Initialization completely writes the time message command time values into the implemented local time counter whereas synchronization verifies whether the time message command time and local time counter matches till the mapped SpaceWire Time-Code level (with a tolerance of previous value) and only modifies the local time if there is a mismatch.

After the time in target is initialized, the time needs to be corrected for time distribution delay (latency) introduced by the time qualification process i.e. the SpaceWire Time-Codes are used for time qualification as the time codes undergo distribution delay the time maintained also delayed and any variation in the oscillator and local clock drift must also be corrected for keeping the time synced.

F. Latency measurement using Time-Stamps

The SpaceWire interface available in both the initiator and target has the capability to send and receive distributed interrupts. The incoming and outgoing SpaceWire distributed interrupts are time stamped in initiator and target. The initiator calculates latency based on these time stamp values. The time stamp values in target are obtained from initiator through RMAP.

The distributed interrupt transmission from initiator (which performs the latency calculation) can be configured to set how often and at which time code distributed interrupts are transmitted and time stamping is performed. The time stamping can be performed in two methods (only Interrupts or Interrupts and Acknowledgement). Initially initiator sends a distributed interrupt and when the target received this interrupt it will send another interrupt (or acknowledgement is provided by the interrupt handler) which will be received by the initiator. At each end transmission and reception is time stamped i.e. the current local time is stored as time stamp values. The latency is calculated from these time stamp values based on Eq. 2, where I_{RX} - initiator time stamp received, I_{TX} - initiator time stamp transmitted, T_{TX} - target time stamp transmitted and T_{RX} - target time stamp received.

$$\text{Latency} = ((I_{RX} - I_{TX}) - (T_{TX} - T_{RX})) / 2 \quad (2)$$

By calculating the latency value repeatedly (at least for about 128 times) and taking an average of it will provide the final latency value. The initiator then transfers the latency correction information to the latency field available in the target by means of RMAP transfer. When the latency values are written it will be adjusted to local time in the target which cancels the distribution delay. The calculations are performed by the software by accessing the time -stamp values and written in the target latency field using RMAP.

The transmission of SpaceWire Time-Codes and distributed interrupts should be separated by a delay; the transmission of a Time-Code should not influence the transmission of distributed interrupt in order to obtain the exact latency of time code transmission. The delay must be greater than the time required to transmit the SpaceWire Time-Code in the initiator.

G. Mitigation of jitter and drift

The jitter and drift correction is performed only in the target. The frequency synthesizer clocked by the local clock drives the local time at a given rate. By changing the frequency synthesizer settings one can adjust the local time. The coupling between local clock and the local time (frequency synthesizer increment value FS_{INC}) is adjusted to the amount of variations seen in the target due to drift or jitter. The variations are obtained in local clock count and adjusted to the frequency synthesizer [6].

The correction needed to be performed for time synchronization are initial offset difference in local oscillator (incorrect frequency), jitter and drift variations. The variations are calculated as differences in local ticks and external ticks. The external ticks are provided by the SpaceWire interface to the SPWTDP component when SpaceWire Time-Codes are received from a remote initiator. The local ticks in target are provided internally in the SPWTDP component, it happens when a local SpaceWire Time Code is generated (generated from the local time) which is only used for internal calculations.

1) Initial offset difference

The number of local clock counts between two local ticks is obtained; similarly number of local clock counts between two external ticks is obtained. These two values are subtracted and the difference is collected over a 64 samples and averaged to get a variation value in local clock count. The variation obtained is based on the local clock of the target node. This variation is multiplied with compensation value and provided to frequency synthesizer to get the initial offset difference corrected. The compensation value is calculated from Eq. 1, considering the target system clock frequency and the number of SpaceWire Time-Codes transmitted every second.

2) Jitter correction and drift mitigation

The number of system clock ticks between local tick and external tick is obtained and averaged over a number of samples (512 samples is used in this implementation). The averaged value is multiplied with compensation value to obtain the correction value and fed into frequency

synthesizer. The main aspect of jitter correction is to keep the local tick in the center of arriving external ticks (or jitter free), the correction value is immediately applied and the local tick is got back to the center, further the correction term is equally distributed to the entire correction interval. The variation in local clock drift is seen as local tick movement from the center which is caught by the averaging process and correction values are fed back as explained above. This will keep the local ticks jitter and drift free. Figure 2 shows the jitter and drift mitigation process and FS stands for Frequency Synthesizer.

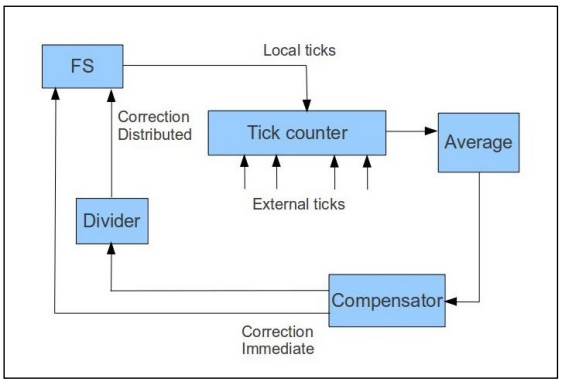

Fig. 2. Jitter and drift correction

H. Time keeping complete process

The initiator initiates the target node through a time message transfer, calculates latency using distributed interrupts and provides latency correction value and starts the jitter and drift mitigation process in the target. The initiator can also send synchronization time messages at regular intervals and the target checks the local time with the received synchronization time messages and adjust the local time if any variations. The transmission of SpaceWire Time-Codes at regular intervals helps to correct any clock drift in the target.

III. VERIFICATION

The IP core developed is verified using simulation for proper functionality. The following section explains about the accuracy achieved in simulation and explains about the verification process.

A. Verification of functionality

A VHDL test bench was developed to verify the functionality of the VHDL IP core. The Test bench consists of initiator and target each with GRSPW2 SpaceWire interface, SPWTDP, AMBA controllers and other components needed for verification. The time synchronization achieved between initiator and target is verified using this test bench. The Time messages from initiator are transferred to target using RMAP writes through SpaceWire link and qualification of these time messages is performed by the SpaceWire Time-Codes transmitted from initiator to target. The target local time is initialized and synchronized (using time messages). The latency is calculated based on the values obtained by time-stamping of received and transmitted distributed interrupts and calculated value is transmitted using RMAP writes to the target.

The local time maintained in both initiator and target is nearly equal, only a single difference (local time least represented value) between the initiator and target local time was noticed. The number of bits used to represent coarse time is 32 bits and fine time is 24 bits, system clock used is 50 MHz and verification is performed for 10 Mbps and 200 Mbps transmission data rate, in both cases only a single difference between the initiator and target local time was noticed. This corresponds to an accuracy of 60 ns, i.e. the difference seen in the 24th fine time bit which represents 2^-24 (~60 ns). The simulation is performed between two SpaceWire nodes without any routers and no additional data traffic in the network other than NULL control codes and Time-Codes.

B. Verification of jitter and drift correction unit

The Jitter and drift correction unit is verified in simulation using a separate VHDL test bench before integrating into the complete system. The local ticks and external ticks are generated from different counters (local time) and external ticks are provided with delay (latency) and variations in delay (jitter) similar to latency and jitter experienced by a SpaceWire Time-Code passing through the network. The clock provided to one of counters (the one which acts as target) is also modified slowly to simulate drift in the local clock of target.

The correction unit must perform the following,

- move the local tick in the center of arriving external ticks (or jitter free)
- adapt the frequency synthesizer according to the drift introduced in the local clock

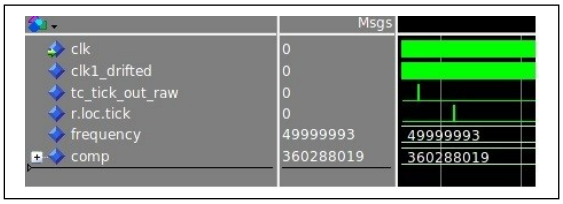

Fig. 3. Adapting to variations in clock drift

The correction unit performed both the needed corrections. The FS_{INC} provided to the frequency synthesizer is also monitored whether it varies according to the amount of drift induced in the local clock, we know the local clock frequency variation and by using the Eq. 1, we can calculate what the FS_{INC} value should be for this frequency variation, the FS_{INC} value varied accordingly and local time remained stable and the influence of drift from local clock is nullified.

Figure 3 shows an image taken from the simulation tool, the comp is the FS_{INC} value provided to frequency synthesizer, initially the simulation started with a frequency of 50 MHz and corresponding FS_{INC} value of 360287970 and the FT_W value is 24 (fine time width). The local clock frequency is increased and decreased to simulate the drift in

either direction and verified for adaptation in frequency synthesizer. Also the real-world data collected during the independent ESA measurements have been used as stimuli to validate the jitter and drift mitigation technique implemented in the correction unit [7].

IV. FPGA BASED PROTOTYPING

FPGA based rapid prototyping has been used during the development. The developed VHDL IP core is integrated into reference avionics system testbed architecture (RASTA) testbed [8]. The testbed consists of GRSPW2 SpaceWire interface with RMAP target and cores like AMBA controllers required for the implementation of protocol. Figure 4 shows the test setup.

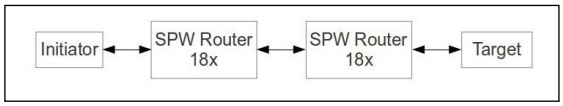

Fig. 4. Test Setup

The necessary RTEMS drivers required for the RASTA systems to operate the added functionality is developed. A test application is developed to demonstrate the time synchronization functionality.

A. Board setup

The actual picture of the test setup used in this implementation is shown in Fig. 5. The setup consists of GR-RASTA-105 acting as an initiator consist of GRSPW2 SpaceWire interface integrated with the newly developed SPWTDP IP core, the GR-RASTA-TMTC act as target which also consist of GRSPW2 SpaceWire interface and SPWTDP IP core. Figure 5 shows 2 GR718 SpaceWire 18x routers which are connected in between the initiator and target. The system clock used in all the hardware is 50 MHz except the target system which used 33 MHz system clock. The time synchronization functionality is tested without any routers in the middle and also with 1 and 2 routers in the middle, the functionality is tested for varying link data rate 2, 10, 50, 100 Mbps. The number of time codes transmitted per second is 64. The target was also replaced with GR-RASTA-105 with system clock 50 MHz and tested similar to the previous set up with 2, 10, 50, 100 and 200 Mbps link data rate.

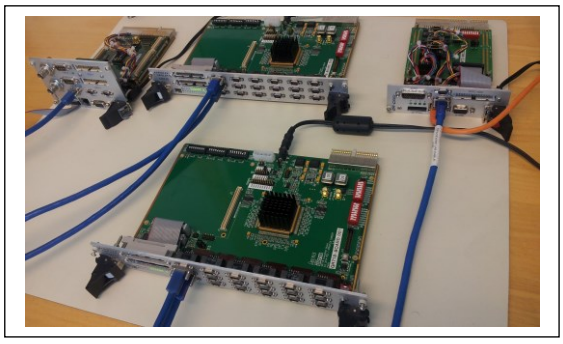

Fig. 5. Picture of test setup

V. RESULTS

The tick generated in initiator during SpaceWire Time-Code transmission and similar diagnostic tick from target (SpaceWire Time-Code and tick is generated just for diagnostics in target) is pulled out and monitored using an oscilloscope. When a tick occurred the local time with lower weights than the size bits mapped to SpaceWire Time-Code time information bits are all zero, so comparing the instance at which ticks generated provides the accuracy in time maintained between the initiator and target.

The Initial oscillator frequency offset in the target is nullified for all the cases mentioned in the previous section and a stable time is maintained between the initiator and target. When the mitigation is disabled (correction unit disabled) in target due to the differences in the local oscillators of initiator and target the tick moved away and time maintained between the system is incorrect, but when the mitigation is enabled the target ticks does not move away from the initiator ticks and maintain a stable time difference. This proves that the effect of oscillator frequency offset is nullified by the mitigation unit.

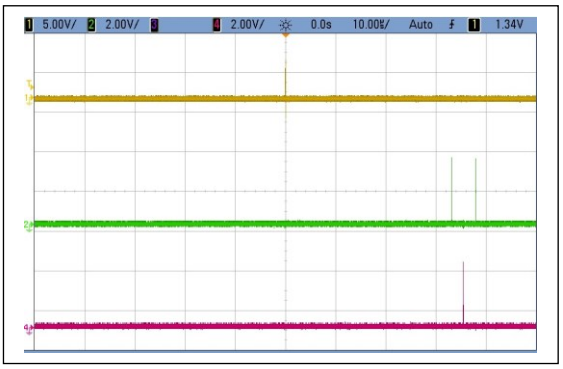

Fig. 6. Before latency correction

Figure 6 shows the ticks viewed through an oscilloscope, yellow in the top (1) is the initiator tick, green in the middle (2) is the incoming ticks with jitter in the target (generated when time codes are received) and pink in the bottom (4) is the final diagnostic tick from the target. Figure 6 shows before latency correction in target and Fig. 7 shows after latency correction. For Fig. 7 the data rate is 2 Mbps with 2 routers in the middle and the initiator running at 50 MHz system clock and target running at 33 MHz system clock.

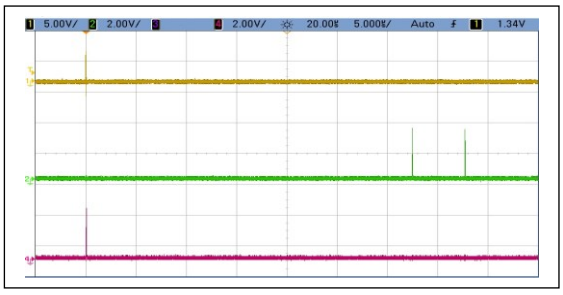

Fig. 7. After latency correction, link data rate is 2 Mbps

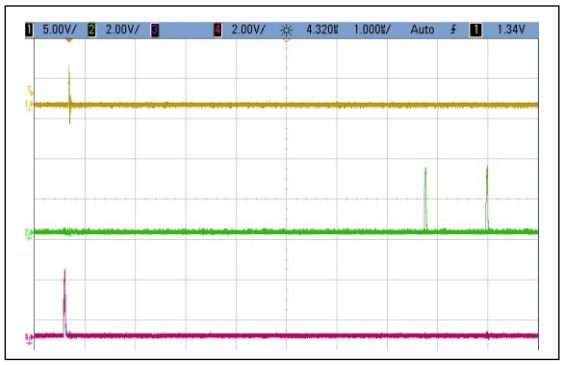

Fig. 8. After latency correction, link data rate is 10 Mbps

For Fig. 8 the data rate is 10 Mbps with 2 routers in the middle and the initiator and target running at 50 MHz system clock.

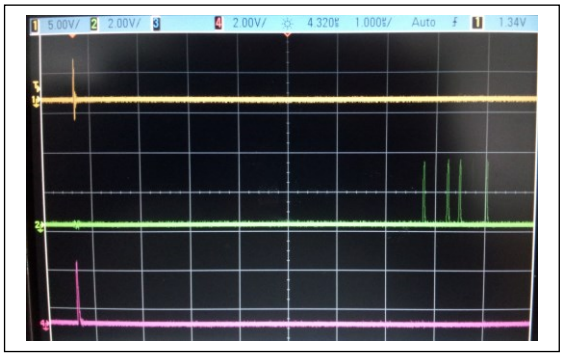

Fig. 9. Camera image of oscilloscope output

In order to depict the incoming time code jitter a direct image of oscilloscope output is shown in Fig. 9.

The time in initiator and target was monitored directly by freezing them to a register by an external trigger which occurs at same instance to initiator and target. The contents of the registers are read out using two debug monitor (GRMON) and the values of local time are compared [9]. The local time differences seen are in correspondence to the time differences seen between the ticks of initiator and target.

Figure 10 shows an image of the two debug monitors, initiator in the left and target in the right. The frozen time value is marked with a box and the difference in time is only a single unit difference between the initiator and target time is noticed.

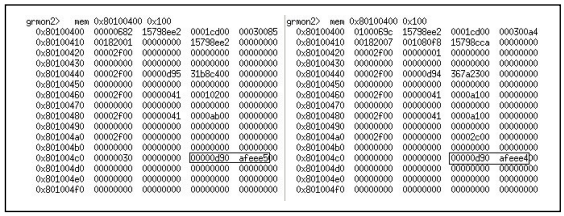

Fig. 10. Time in initiator and target frozen and stored in a register.

Initially the accuracy was measured between the initiator and target without any additional traffic (other than NULL control codes and SpaceWire Time-Codes) the time maintained in both initiator and target have only a single difference between the initiator and target time was noticed and this corresponds to an accuracy of 60 ns, i.e. the difference seen in the 24th fine time bit which represents 2^-24 (~60 ns).

The same level of accuracy was not able to achieve with data traffic in the network. The variation in jitter because of data traffic influences this accuracy. The data characters are 10 bits length whereas the NULL's are 8 bits, the jitter varies from 8 to 10 bits of transmission period. The jitter mitigation technique implemented in this design tries to nullify the jitter by being in the center of the incoming Time-Codes, the variation in jitter from 8 to 10 bits due to variation in traffic results in an inaccuracy of single transmission bit period per link.

CONCLUSION

The implementation successfully mitigates effects of the oscillator in the target and maintains a stable time between initiator and target i.e. the time in the target and time in the initiator is maintained at a constant rate. The implementation also nullifies the impact of drift (local clock oscillator) in local time in the target. A methodology to calculate latency using distributed interrupts is defined, implemented and verified. The inaccuracy resulting from the jitter variation have a significant impact for low link data rate like 2 Mbps (500 ns) the effect of jitter variations will have less impact for higher data rate like 200 Mbps (5 ns). Even with a defect in correction principle the jitters impact on time keeping is reduced by a factor of 10 for any number of links.

REFERENCES

[1] S. Habinc, M Isomaki, D. Hellstrom, "CCSDS Time Distribution over SpaceWire," International SpaceWire Conference, [November 2011]

[2] "Time Code Formats.", Internet: http://public.ccsds.org/publications/archive/301x0b4e1.pdf, [November 2010].

[3] Space Engineering; SpaceWire Links, nodes, routers and networks, ECSS-E-ST-50-12C, July 2008.

[4] Space engineering: SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C, [February 2010].

[5] S. Habinc, A. Sakthivel, M. Suess, "SpaceWire – Time Distribution Protocol," International SpaceWire Conference, [June 2013].

[6] H. Kopetz, A. Ademaj, A. Hanzlik, "Integration of internal and external clock synchronization by the combination of clock-state and clock-rate correction in Fault-Tolerant Distributed Systems," rtss, pp.415-425, 25th IEEE International Real-Time Systems Symposium (RTSS'04), 2004.

[7] M. Suess, F. Siegle "SpaceWire Time Code Latency and Jitter," International SpaceWire Conference, [June 2013].

[8] “GR-RASTA.” Internet: http://gaisler.com/doc/gr-rasta_product_sheet.pdf

[9] “GRMON2 User’s Manuel.” Internet: http://gaisler.com/doc/grmon2.pdf, [June, 2014].

STP-ISS Transport Protocol for Spacecraft On-board Networks

SpaceWire networks and protocols, Long Paper

Yuriy Sheynin, Irina Lavrovskaya, Valentin Olenev, Ilya Korobkov
Saint-Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
sheynin@aanet.ru, {irina.lavrovskaya, valentin.olenev, ilya.korobkov}@guap.ru

Dmitry Dymov, Sergey Kochura
JSC "Academician M.F. Reshetnev" Information Satellite Systems"
Zheleznogorsk, Russia
dymovdv@mail.ru, kochura@iss-reshetnev.ru

***Abstract*— SpaceWire is a data-handling network for spacecraft which combines simple, low-cost implementation, with high performance and architectural flexibility. SpaceWire is intended for data-handling applications but does not address such aspects of quality of service as robustness, determinism and durability that are essential requirements. Nowadays there is a number of transport protocols intended to operate over SpaceWire. They are: RMAP, CCSDS PTP, STUP, JRDDP and STP. Each of them is designed to solve its particular tasks. However, there is no SpaceWire oriented transport protocol providing reliability, guaranteed services and scheduling.**

The paper presents the new Transport protocol STP-ISS for SpaceWire networks. Firstly, it gives an overview and analysis of SpaceWire oriented transport protocols, then, considers general requirements for the Transport protocol to operate over the SpaceWire network technology. Finally, we describe the current status of the STP-ISS specification and consider the future evolution of the standard.

***Index Terms*— SpaceWire, STP-ISS, Transport Protocol, On-board Network, Quality of Service.**

I. Introduction

SpaceWire is a data-handling network for the spacecraft which combines simple, low-cost implementation with high performance and architectural flexibility [1]. MIL-STD 1553 has long been the communications bus of choice for spacecraft avionics. Limited to 1 Mbits/s aggregate data rate and constrained to the bus topology, MIL-STD 1553 is struggling to cope with today's spacecraft requirements. So new technologies are being actively integrated into new spacecrafts, and SpaceWire is one of them. SpaceWire is now being used on more than 30 high profile missions and by all of the major space agencies and space industry over the world.

The basic SpaceWire protocol standard covers three bottom layers of the OSI model and does not provide transport services. There are a number of transport protocols that had been specially developed to operate over SpaceWire. So the first part of the paper gives the overview and analysis of these protocols.

II. Space Oriented Transport Protocols Review

A. Remote Memory Access Protocol

The Remote Memory Access Protocol (RMAP) has been designed to support a wide range of SpaceWire applications. Its primary purposes however are to configure a SpaceWire network, to control SpaceWire nodes and to gather data and status information from those nodes [2]. RMAP can be used for the SpaceWire configuration, setting the parameters of a device and network information gathering. Also it can be used for data transmission, with polling as the main mode of operation.

The RMAP protocol can be described by its following general features:

- RMAP is a connectionless transport protocol;
- supports path, logical and regional addressing;
- write commands can be acknowledged or not acknowledged, verified and not verified;
- provides means for reading and writing of data into the memory by just one command (read-modify-write command);
- no timeouts mechanism;
- no flow control.

RMAP defines three types of commands:

- write commands;
- read commands;
- read-modify-write commands [2, 3].

The RMAP protocol provides guaranteed delivery service in the acknowledged mode and best effort service in a non-acknowledged mode.

B. CCSDS Packet Transfer Protocol

CCSDS Packet Transfer Protocol (CCSDS PTP) – is a packet transfer protocol which encapsulates a CCSDS Space Packet into a SpaceWire packet, transfers it from an initiator to

a target across a SpaceWire network, extracts it from the SpaceWire packet and passes it to the target user application [4].

The CCSDS PTP protocol can be described by its following general features:

- connectionless protocol;
- user may request data transfer at any time;
- variable or fixed packet length (minimal length is 7 bytes, maximal – 65542 bytes);
- unidirectional data transfer without acknowledgments;
- no data retransmission mechanism;
- no packet verification (it's a user application functionality) [4].

CCSDS PTP does not provide any mechanisms for guaranteeing a particular quality of service [4].

C. Serial Transfer Universal Protocol

Serial Transfer Universal Protocol (STUP) is intended for data transfer over the SpaceWire network. Its main feature is a minimized complexity [5].

The general features of the STUP protocol are:

- connectionless protocol;
- easy to implement protocol (minimized complexity);
- just 2 types of commands: write and read;
- does not provide guaranteed delivery services.

STUP commands have checksum fields for verification of received data [5].

D. Joint Reliable Data Delivery Protocol

The Joint Architecture Standard Reliable Data Delivery (JRDDP) is a protocol which provides reliable data transmission. It uses the lower-level SpaceWire data link layer to provide reliable packet delivery services to one or more higher-level host application processes [6].

The JRDDP protocol has the following main features:

- connection-oriented protocol;
- multiple logical connections;
- reliable data delivery;
- detection of missing packets;
- out-of-sequence packet reordering;
- buffer fragmentation and reassembly [6].

The JRDDP defines the following packet types:

- application data;
- acknowledge;
- open/reset command;
- close command;
- urgent.

JRDDP provides three types of quality of service: priority, guaranteed and best-effort data delivery. According to JRDDP specification the data flows should have the following priorities: acknowledgment packets, control packets, urgent packets, retransmit packets, data packets.

Best-effort QoS is optionally used for urgent messages delivery such as time broadcasts, messages with exceptions and errors control, meta-messages, etc. [6].

The JRDDP protocol provides fault detection and fault tolerance by means of CRC checksum and packet sequence numbering. Moreover, it uses timeouts for detection of missing and duplicate packets and acknowledgements for indication a successful packets delivery.

E. Streaming Transport Protocol

The Streaming Transport Protocol (STP) is developed for streaming data transmission over SpaceWire network. This protocol also supports simultaneous transmission of multiple coherent data flows [7].

The STP protocol is oriented for asymmetric establishment of transport connection: on the one side there is a host (master), and the slave device is on the other side. The host device is an initiator of a transaction session. The master performs the connection establishment, configuration of connection parameters and packets flow control [7].

The STP protocol can be described by its following general features:

- connection-oriented protocol;
- reliable handshake for connection establishment and teardown (3-way handshake);
- asymmetric connection (data transmission is performed from slave to host device);
- multi-streaming (up to 65535 connections);
- fixed length of transmitted data;
- periodical data transfer in specified time period in accordance with the configuration parameters and during the whole duration of the connection;
- data delivery without acknowledgements and retransmission;
- data flow control.

The STP protocol uses the following mechanisms to provide fault detection and fault tolerance:

- packet fields verification, header and payload CRCs;
- timeouts mechanism;
- terminal node status monitoring procedure (status command sending).

F. Protocols comparison

General features of each overviewed protocol are given in the Table I [8].

It is clear from the Table 1, that there is no such a protocol existing for the SpaceWire networks which provides reliability, guaranteed data delivery, scheduling and configuration flexibility. Therefore, a new Transport protocol should be developed to operate over SpaceWire.

TABLE I. PROTOCOLS COMPARISON

Feature \ Protocol	RMAP	PTP	STUP	JRDDP	STP
Configuration flexibility	√	–	–	–	√
Multiple applications	–	–	–	√	√
Data flows of different priorities	–	–	–	√	–
Data flow control	–	–	–	√	√
Transport connection establishment	–	–	–	√	√
Segmentation	–	–	–	√	–
Data correctness check	√	–	√	√	√
Data sequence check	–	–	–	√	–
Data retransmission	–	–	–	√	–
Acknowledgements	√	–	–	√	–
Scheduling	–	–	–	–	–

III. GENERAL REQUIREMENTS FOR THE TRANSPORT PROTOCOL

This section gives a list of the main requirements to the new Transport protocol. These requirements were elaborated in such a way that the new Transport protocol will cover all previously unsolved problems.

A. Transport interface

The Transport layer protocol should provide transmission of the following general data flows passing from the Application layer: control commands, application messages, SpaceWire time-codes, SpaceWire interrupt codes and interrupt-acknowledge codes.

B. Segmentation

Segmentation of large messages should be performed by the Application layer. The target segments with the additional service information should be passed from the Application layer to the Transport layer. The transport layer should give ability for the Application layer to assemble the message from a number of segments, so it should support transmission of additional information in the secondary header (for example, segment number).

C. Data flows and priorities

Each data flow should have its particular priority. The data flows should have the following precedence:

1. Control commands – the highest priority;
2. Urgent messages;
3. Common messages – the lowest priority.

D. Buffering on the transmitter side

Transport protocol should contain a separate logical buffer for each data flow priority.

E. Quality of service

The target Transport protocol should provide additional fault detection over the SpaceWire connection by means of the following mechanisms:

- CRC checksum;
- successful packet transmission acknowledgements;
- detection of lost packets by timeouts.

Each transport data flow is characterized by particular features and, consequently, requires its particular quality of service. The priority quality of service is required by all data flows. Control commands and urgent messages should be delivered with guaranteed quality of service. In turn, common messages can be transmitted with guaranteed or best effort quality of service.

Taking into account the transport protocol analysis and the technical requirements, we proposed a list of technical solutions for the first revision of STP-ISS transport protocol. At the moment a first version of the STP-ISS specification is in the modeling stage. The next section of the paper gives an overview of the STP-ISS transport protocol and shows its further evolution plans.

IV. STP-ISS TRANSPORT PROTOCOL

Figure 1 shows an example of the on-board network for a small-sized satellite. Dotted lines show the information flows from sensors to the other parts of the satellite. It is just an example of the applied topology, and STP-ISS can be used for the much more complex networks also.

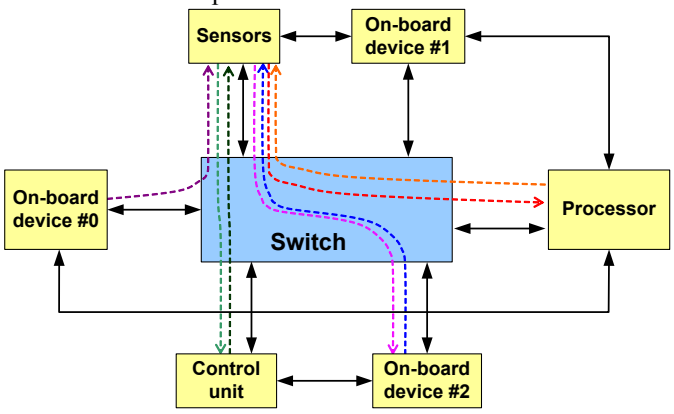

Fig. 1. An example of the onboard network topology

A. STP-ISS general description

STP-ISS is a transport layer protocol that describes the informational and logic interaction between onboard devices, packets' formats and packet transmission rules for the SpaceWire network. The onboard software performs the functions of the session, presentation and application layers according to the OSI model [9]. STP-ISS protocol corresponds to the Transport layer and provides means for transmission of data between the nodes of the network with the required quality of service type and data flow priority. This protocol gives ability for data resending in case of the error detection in the received data. This procedure is called resending. The place of

the STP-ISS protocol in the SpaceWire standard's family and conformity to the OSI model is shown in Fig. 2.

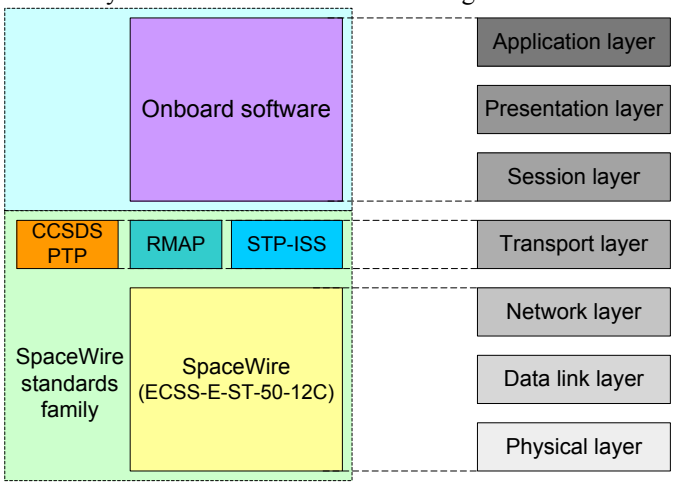

Fig. 2. STP-ISS protocol and OSI model

B. STP-ISS interfaces

There are three interfaces for the interaction between the STP-ISS and Applications: Data Interface, Configuration Interface and Control Codes Interface. In addition, there are two interfaces for the interconnection with the SpaceWire: SpaceWire packets interface and Control Codes Interface (see Fig. 3).

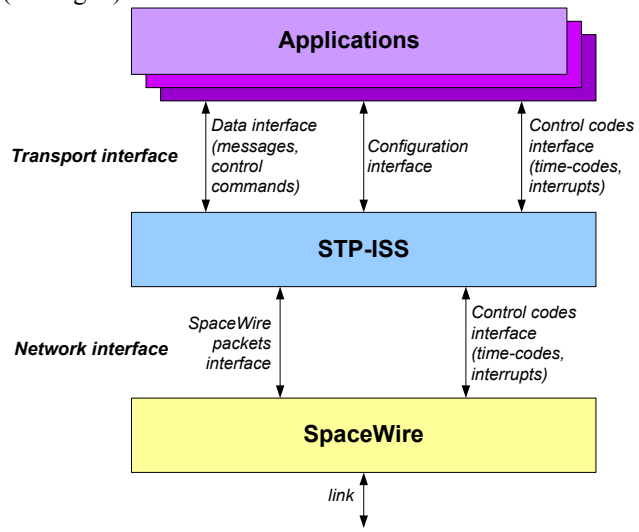

Fig. 3. STP-ISS interfaces

STP-ISS provides transmission of the following types of data through those interfaces:

- control commands;
- data packets;
- SpaceWire time-codes;
- SpaceWire distributed interrupts and interrupt-acknowledges.

The data interface provides transmission of control commands and data messages. Messages and control commands are transmitted to the remote node by encapsulation into SpaceWire packets.

The configuration interface provides means for the STP-ISS configuration parameters change and for transmission of status information and reset commands.

The control codes interface passes the SpaceWire time-codes and distributed interrupts to the SpaceWire and then – to the other nodes of the network.

C. STP-ISS application messages

One of the main tasks of the STP-ISS transport protocol is to provide the transmission of messages from the Applications to the remote nodes of the SpaceWire network. The message is a data block that is passed to the STP-ISS from the application layer. There are two types of application messages:

- urgent messages (higher priority);
- common messages (lower priority).

Messages from Applications are encapsulated into SpaceWire packets at the transport layer (see Fig. 4).

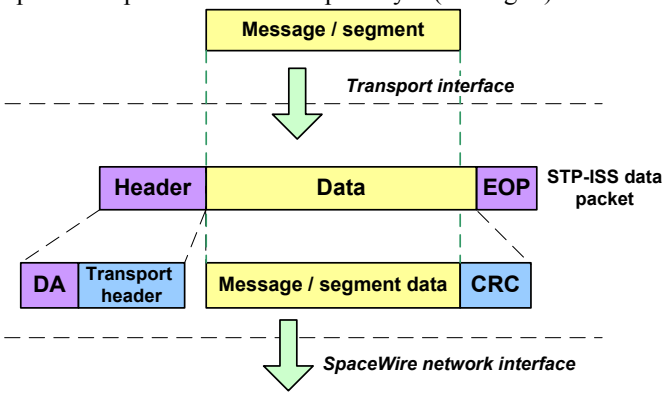

Fig. 4. STP-ISS encapsulation of a message into a SpaceWire packet

The length of each message should not exceed 2048 bytes, because the STP-ISS transport protocol does not perform segmentation. Segmentation of messages is done by the application layer and STP-ISS processes these segments as usual independent messages. The Application layer of the remote node assembles the segments into the original message. The message should be assembled basing on the segment identifiers that should be transmitted in the segment header. For this purpose STP-ISS packet has a secondary header, which should be used by the Application to transmit the information for the messages assembling (for example, a number of the segment).

STP-ISS provides the reliable data transmission by using CRC-16 for protection of payload and packet header and for errors detection. CRC-16 covers the packet starting from the first byte of the STP-ISS packet header and finishing with the last byte of data, excluding the end of packet symbol EOP (see Fig. 5).

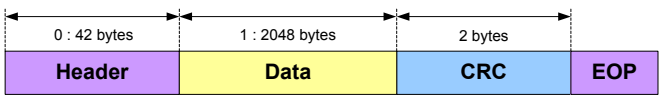

Fig. 5. STP-ISS data packet format

D. STP-ISS lifetime timers

STP-ISS protocol has a special packet lifetime timer, which counts the time, when the packet is still actual in the SpaceWire network. Each packet is stored in the buffer during its lifetime. The value of the lifetime timer is an STP-ISS configuration parameter and it could be set during the configuration stage. Each packet type could have different values of lifetime timer. The lifetime timer should start when the packet is written to the buffer. The packet should be deleted from the buffer when the lifetime timer expires.

E. Resend buffers

The transmitter side of the protocol has separate buffers for each priority of the transmitted data:

- control commands buffer;
- urgent messages buffer;
- common messages buffer;

The size of these buffers should be set depending on the message or segment size, which the node uses for the data exchange. Also the size of the buffer depends on the type of the device, which implements STP-ISS, also. But for each buffer (on the transmitter of receiver side) it is recommended to set the size such a way, that buffer should be able to store minimum two packets. These buffers are shown in Fig. 6.

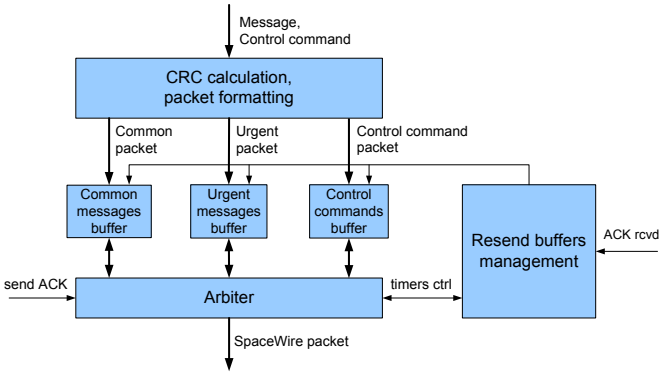

Fig. 6. STP-ISS resend buffers

The packet should be stored in the buffer until one of the following events occurs:

- the STP-ISS transmitter received an acknowledgement for this packet;
- transmission of the packet with the best effort quality of service to the SpaceWire network;
- lifetime timer for this packet expired.

The receiver side of the transport protocol has one buffer for all types of the packets, because SpaceWire packets come from the SpaceWire interface sequentially.

F. STP-ISS quality of service

One of the benefits of the STP-ISS is the possibility to transmit data using the following quality of service types:

- Priority quality of service;
- Guaranteed delivery quality of service;
- Best effort quality of service.

G. Priority quality of service

Priority quality of service is the main quality of service type that should be supported by all the network end-node devices, which communicate with STP-ISS. According to this quality of service type, the data with the higher priority should be transmitted first. Current STP-ISS specification supports 7 levels of priorities:

1. Acknowledgement packets;
2. Control command packets;
3. Resend control command packets;
4. Urgent data packets;
5. Resend urgent data packets;
6. Resend common data packets;
7. Common data packets.

H. Guaranteed delivery quality of service

Guaranteed delivery quality of service provides confirmation for the successful packet transmission by sending the acknowledgement packets. Also it resends the data from the transmitter end-node if the acknowledgement is lost (resending mechanism).

Guaranteed delivery is provided by a number of mechanisms such as resend timers and successful transmission acknowledges. Data resending is based on the packets numeration. This numeration is performed by the application layer by giving an identification number for each packet that is transmitted from a particular application. So the combination of the application identifier and a packet identification number uniquely identifies each packet.

If a packet is passed to the network layer with the guaranteed delivery quality of service, STP-ISS should start the resend timer for this packet. If a resend timer expires before the receipt of an acknowledgement, this means that the packet or its acknowledgement is lost, or the packet has been corrupted during the transmission. So when the resend timer expires, the corresponding packet should be sent to the network again. Each transmitted packet should have its own resend timer.

The acknowledgement packets are used for confirmation of the packet's successful receipt. Acknowledgements are sent when there is no CRC error, the data length field is correct and there is a flag "Guaranteed delivery packet" set to 1 in the received packet's header. Within the acknowledgement the receiver sends a combination of the application identifier and the transmitted packet's identification number. When the transmitter gets the acknowledgement, the corresponding packet should be deleted from one of the transmitter's resend buffers. All the timers associated with this packet should be stopped.

I. Best effort quality of service

Best effort quality of service provides data transmission without sending acknowledges. Such packets have the flag "Guaranteed delivery packet" set to 0 and they do not need resend timers. When STP-ISS receiver gets a best effort packet it checks the CRC and data length, but in case of error or if the packet ends with EEP, data packet still should be sent to the Application, but with an error indication.

J. STP-ISS configuration parameters

The important STP-ISS feature is the configuration flexibility. The protocol has a number of configuration parameters, which give ability to tune the protocol depending on the developer needs (required quality of service, onboard equipment type, resource constrains, etc.). Configuration of the STP-ISS protocol is performed via the configuration interface. Configuration is done in the following cases:

- switching-on/off the device;
- reset;
- switching to the redundant onboard device;
- emergency recovery.

The current STP-ISS specification describes 5 configuration parameters:

1. Control command lifetime;
2. Urgent message lifetime;
3. Common message lifetime;
4. Resend timeout;
5. Guaranteed / best effort data transmission.

K. Reset and Flush

There are two additional signals that could be passed from the application layer to the STP-ISS though the configuration interface: Reset and Flush. Reset corresponds to the warm reset, and Flush is used for the clearing of both transmit and receive buffers.

When STP-ISS gets the Reset, it should clear transmit and receive buffers, stop all the timers corresponding to deleted packets and set all the configuration parameters to the default settings.

When STP-ISS gets the Flush, it also should clear transmit and receive buffers and stop all the timers corresponding to deleted packets, but all the configuration parameters should not be changed.

V. Conclusion

The paper gives an overview of STP-ISS transport protocol for the onboard SpaceWire networks. The current revision of the STP-ISS is the first version and the protocol will be actively evolved and trialled; further updates are planned also. The following additions are considered to be included to the second revision of the STP-ISS:

1. Scheduled quality of service, when each node of the SpaceWire network will have a permission to send the data during the particular time-slot only.
2. Connection-oriented data transmission.
3. Flow control mechanism for each transport connection.
4. Duplicate packets detection.

Also we plan, that the second revision of STP-ISS would successfully work with the first revision.

References

[1] ESA (European Space Agency). Standard ECSS-E-50-12C, “Space engineering. SpaceWire – Links, nodes, routers and networks. European cooperation for space standardization”. Noordwijk: ESA Publications Division ESTEC, 2008. 129 p.

[2] ESA (European Space Agency). Standard ECSS-E-ST-50-52C, “Space engineering. SpaceWire — Remote memory access protocol. European cooperation for space standardization”. Noordwijk: ESA Publications Division ESTEC, 2010. 109 p.

[3] ESA (European Space Agency). Standard ECSS-E-50-11 Draft E, “Space engineering. SpaceWire — Remote memory access protocol. European cooperation for space standardization”. Noordwijk: ESA Publications Division ESTEC, 2006. 58 p.

[4] ESA (European Space Agency). Standard ECSS-S-ST-50-53C, “Space engineering. SpaceWire — CCSDS Packet Transfer Protocol. European cooperation for space standardization“. Noordwijk: ESA Publications Division ESTEC, 2010. 21 p.

[5] EADS Astrium (European Aeronautic Defence and Space Company), SMCS-ASTD-PS-001 1.1, “STUP SpaceWire Protocol Protocol Specification”, 2009. 7 p.

[6] Sandia National Laboratories, Sandia report SAND2011-3500, “Joint Architecture Standard Reliable Data Delivery Protocol Specification”. Sandia National Laboratories, Albuquerque, New Mexico, 2011. 39 p.

[7] Sheynin Y., Suvorova E., Schutenko F., Goussev V. “Streaming Transport Protocols for SpaceWire Networks”, International SpaceWire Conference 2010, Saint Petersburg, 2010.

[8] V. Olenev, I. Lavrovskaya, I. Korobkov, D. Dymov, “Analysis of the Transport Protocol Requirements for the SpaceWire On-board Networks of Spacecrafts”, Proceedings of 15th Seminar of Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program; Saint-Petersburg: Saint-Petersburg University of Aerospace Instrumentation, 2014. pp. 65-71.

[9] Tanenbaum, A. S., “Computer Networks”, Fifth Edition; Prentice Hall, 2011. 962.

Satellite / spacecraft on-board data handling by coupling ARINC-664 (AFDX) and SpaceWire

NOT PERMITTED TO PUBLISH PAPER

Components (Short)

Solutions for copper-based SpaceFibre Links

SpaceWire Components, Short Paper

Stéphane Hermant
'Axocom' Space Products Division
Axon' Cable SAS
Montmirail, France
s.hermant@axon-cable.com

Nigel Kellett
Axon' Cable Ltd
Rosyth, Scotland
n.kellett@axon-cable.co.uk

***Abstract*—This paper will primarily present a design for high speed copper-based links, capable of transmitting SpaceFibre signals at 2.5Gb/s or higher, using a custom interface connector not significantly larger than the 9 way Micro-D style used for existing SpaceWire links. The paper additionally looks briefly at the feasibility of creating miniature, multi-channel SpaceFibre links for short lengths or 'inside the box' applications.**

***Index Terms*—SpaceFibre, SpaceWire, Micro-D, Nano-D, AXOMACH.**

I. Introduction

Axon' Cable has been working recently on a copper-based solution for transmitting SpaceFibre signals at 2.5Gb/s or higher – using connectors (for a single channel version) with approximately the same space envelope as the current 9 way Micro-D style connector used for **SpaceWire** links. This work is being carried out in connection with the **SpaceFibre Demonstrator** contract A0/1-7489/13 for ESA and the University of Dundee.

An additional remit of this contract is to explore initial feasibility of creating miniature, multi-channel links for short length applications, similar in performance to existing commercial products such as eSATA or PCIe, but using space grade componentry.

Regarding the prime objective of the contract, Axon' has elected to propose a design based on a modified version of its **AXOMACH** product.

II. AXOMACH HERITAGE

The AXOMACH product range is a space grade family of cable assemblies capable of operation at up to 10 Gb/s per channel. Developed originally for a military space application, the product has since been extensively evaluated by the CNES French Space Agency, and is currently in the process of being created as a ESA ECSS component. It is based on two RF quality coaxial cables per channel, terminated into impedance-matched, EMC optimized and polarized connectors.

The existing 2 channel crossover version of this product has already been successfully demonstrated by Star Dundee in their SpaceFibre simulator, operating at 2.5Gb/s.

As well as the original (classified) military satellite use, AXOMACH has, among other applications, been integrated within the **Mars Atmosphere and Volatile EvolutioN (MAVEN)** mission, launched on November 18, 2013. The probe is scheduled to start orbiting Mars on September 21, 2014, to explore the planet's upper atmosphere, ionosphere and interactions with the sun and solar wind to find out how and why the Red Planet has been losing its atmosphere over billions of years.

III. DESIGN SOLUTION

For the SpaceFibre Demonstrator Links, Axon' has taken the existing 2 channel crossover version of AXOMACH and placed all four coaxial contacts, which together form the two channels, into one single D-Form on the connector face, thus minimizing the overall connector width. This brings the overall connector space envelope quite close to that of the existing 9 way Micro-D connector currently used for SpaceWire links.

For clarity, this solution, although similar in size, is completely different from a SpaceWire connector and is not, therefore, backwards compatible with SpaceWire.

Axon's intention is to make this family an open source solution and put the product forward for ECSS approval.

Fig. 1: Left: Reduced size SpaceFibre Demonstrator connector;
Right: Existing 2 channel AXOMACH connector

Fig. 2: Inline version of SpaceFibre connector

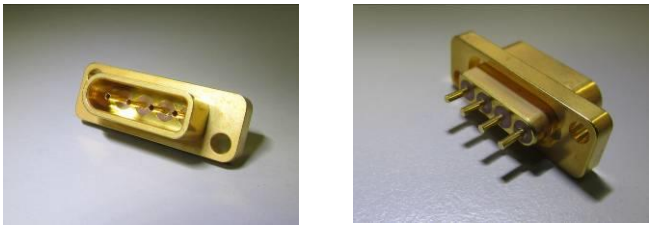

Fig. 3: PCB version of SpaceFibre connector

One of the main goals of the study is to reduce the size of the existing connector. The re-design from the AXOMACH starting point saves around 38% surface area on the equipment interface:

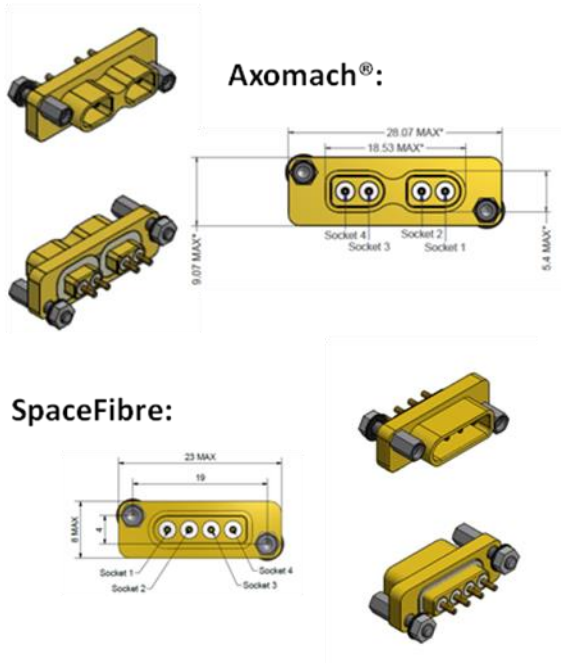

Fig. 4: Surface reduction: 28.07 x 9.07 to 23 x 8:
38% saved

IV. PERFORMANCE RESULTS – AXOMACH STYLE SPACEFIBRE LINKS

At the time of writing this paper the very first SpaceFibre Demonstrator links had just been manufactured and rapidly tested. In general, Axon' was expecting that the overall SpaceFibre performance would be acceptable, but particular attention was to be paid to crosstalk results due to the 'compressing' of all four contacts into the same D-form space.

Initial results are summarised later in **Table I**.

IV.1 Production of Different elements

The RF coaxial cable is exactly the same cable reference, Ax2.4, as already used for the AXOMACH range – no more difficult to manufacture.

The connector, however, needs very specific tooling to manufacture it in order to make the assembly of all the parts, including cable soldering, insulation parts and contact insertion. Particular attention is required on skew reduction during the cutting phases of the coaxial cables.

A transmission test is performed on each link using an Eye Pattern mask to verify the signal integrity. On the left in **Figure 5** below, a pattern generator up to 12.5Gb/s, and on the right, a sampling scope with 50Ghz BandWidth.

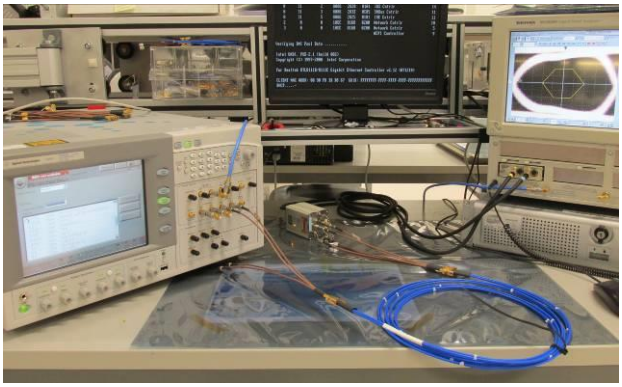

Fig. 5: SpaceFibre Links Transmission test general setup

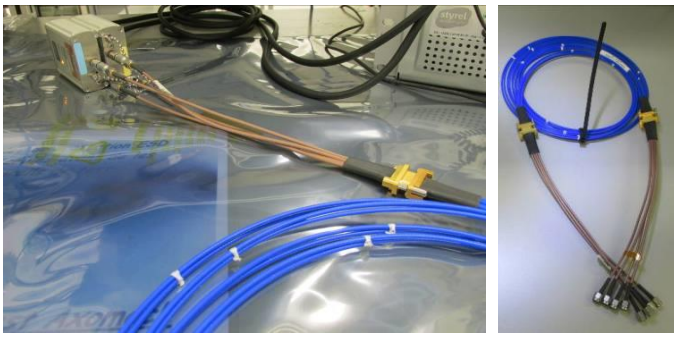

Fig. 6: Connection from the scope to the harness under test using a SpaceFibre Test Adaptor Harness to the equipment (with SMA connectors)

Ultimately, within the scope of the project, five different lengths will be tested in order to be compatible with the range of lengths to be used in the future. These test vehicles are each made with 4 coaxial cables (**Figure 7**) linked to the SpaceFibre dual channel (4 way) male connector at each end as described in **Figure 8**.

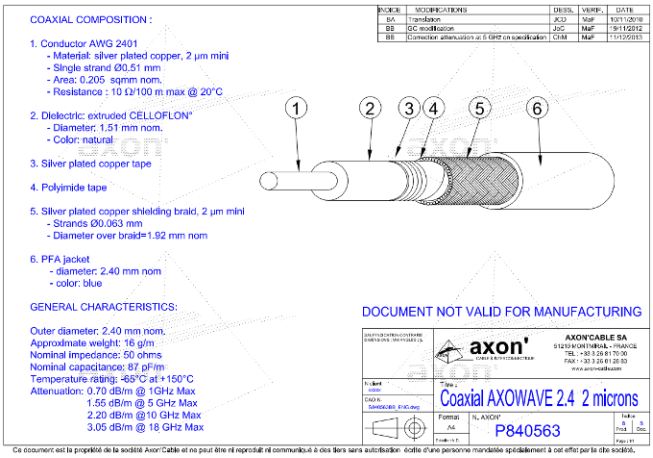

Fig. 7: Ax2.4 AXOMACH and SpaceFibre coaxial cable

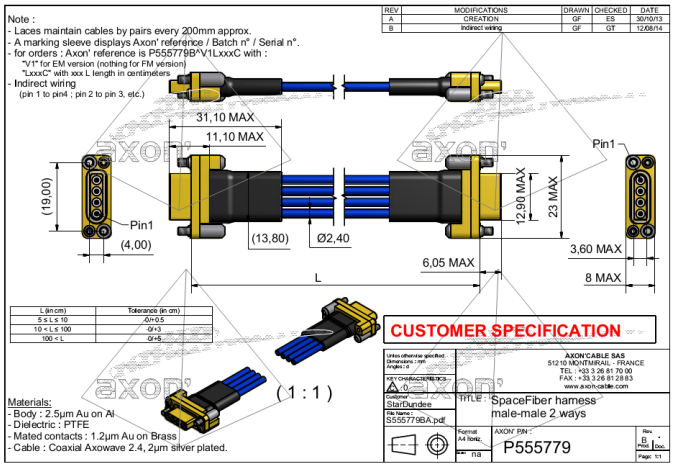

Fig. 8: General design of SpaceFibre links

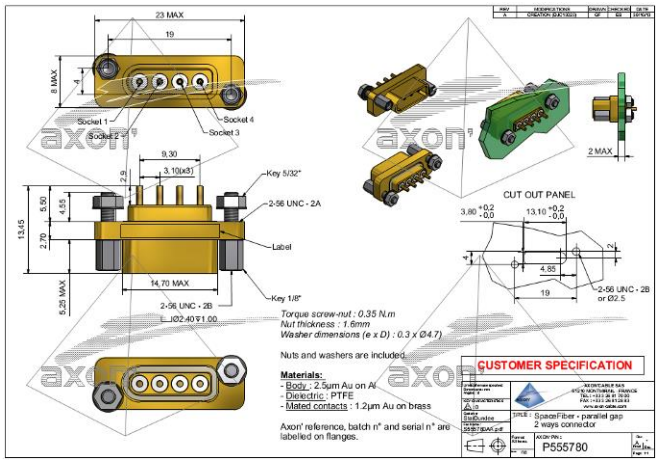

Fig. 9: PCB version of SpaceFibre connectors

The input signal is well defined to verify the signal integrity. In order to be working in the worst case, we set the signal level at the minimum. (300mV) (**Figure 10**)

The mask of the output signal from the harness is also well defined by the SpaceFibre specification (**Figure 11**)

Then the measurement can be performed easily on the scope using the mask test sequence.

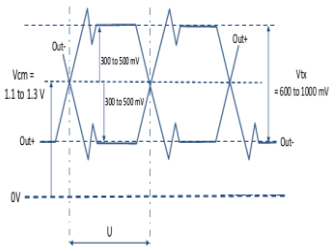

Fig. 10: Input signal

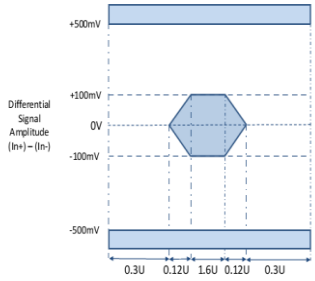

Fig. 11: Output signal mask

Table I, below shows the initial test results, comparing 2M long versions of a classic AXOMACH 2 channel crossover with the same product using the SpaceFibre connector.

TABLE I. COMPARATIVE RESULTS BETWEEN CLASSIC AXOMACH AND SPACEFIBRE CONNECTOR

CHARACTERISTIC	AXOMACH crossover, 2M	SpaceFibre crossover, 2M	Limits (From AXOMACH)
Insulation Resistance	Similar		>5000MΩ
Voltage proof Leakage current	Similar		<2mA/600Vrms
Contact resistance (Rated current)	Similar		<5mΩ
Coax cond. resistance	Similar		<110mΩ/m
Coaxial shield resistance	Similar		<45 mΩ/m
Metal shell conductivity	Similar		50mΩ
Characteristic Impedance	Similar		100 Ω +/-10
Skew between coaxial cables (4)	Similar		<20pS
Jitter PP (5)	Similar		<20pS
Jitter rms	Similar		<5pS
Quality factor	Similar		>7
Insertion loss	Similar		<1dB (5Ghz) <2dB (10Ghz)
Return loss	Similar		<12dB (5Ghz) <9dB (10Ghz)
Crosstalk (TBC)	**<-45dB@5Ghz** **<-35dB@10Ghz**	**<-30dB@5Ghz** **<-20dB@10Ghz**	
Mask test	Similar		Go/NoGo mask

So for the static tests (continuity and insulation) and the dynamic tests (signal integrity), the characteristics are very similar to AXOMACH family. The main difference, as expected, comes from crosstalk which is significantly poorer than that of the classic AXOMACH. Less than -30dB up to 5Ghz (-45dB for AXOMACH) and less than -20dB up to 10Ghz (-35dB for AXOMACH). These initial results on a 2M length, see table Figure 13, need to be confirmed on all the test

vehicles.V. FEASIBILITY STUDY INTO MINIATURE MULTI-CHANNEL SPACEFIBRE LINKS

A second remit of the EMITS was to explore the feasibility of creating Space grade miniature multi-channel links for short length applications.

Considerations. The links and were to be as small as possible, but using space grade components throughout, the key objective being the possibility of having multiple channels in the same connector.

Initial design proposal: multiple, sub-miniature space grade coaxial cables, style SM50, terminated into ESA evaluated Nano-D connectors, according to ESCC 3401-086

Principle: 4 x SM50 coaxials form one dual way channel of SpaceFibre. By terminating the central coax core to one contact and adjacent coaxial contact to the next-but-one contact (i.e. skipping one contact each time) we can achieve the necessary space to terminate the coaxials, and also help with the impedance objective. All the coaxial screens are terminated together to the outer shell of the connector.

In this way, 8 coaxials can be terminated to a standard 15 way Nano, achieving 4 SpaceFibre channels.

- 12 coaxials in a 25 way Nano, giving 6 channels
- Or 24 coaxials in a 51 way Nano providing 12 channels

An initial feasibility test was conducted on a prior manufactured sample with a typical SpaceWire (9 way Micro-D) connector at one end and a 15 way Nano-D connector at the other. (**Figure 13**)

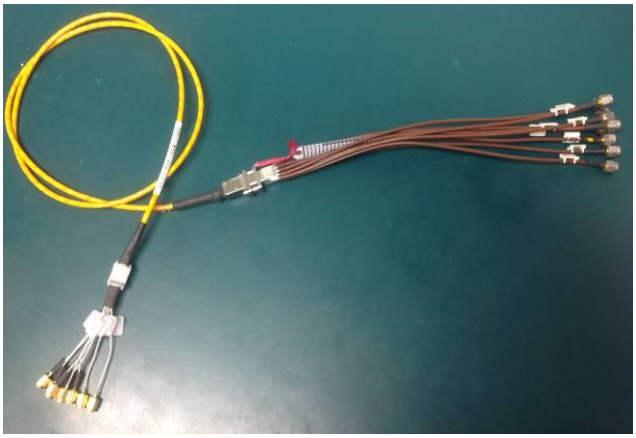

Fig. 13: Test vehicle with MDSA 9pins and NDSA 15p, test leads fitted

Conclusion. The initial test results were positive, with the Nano-D link capable of up to 3.4 Gb/s for a 1m length, and up to 1.5 Gb/s for a 2.3m length, see also **Figure 14**.

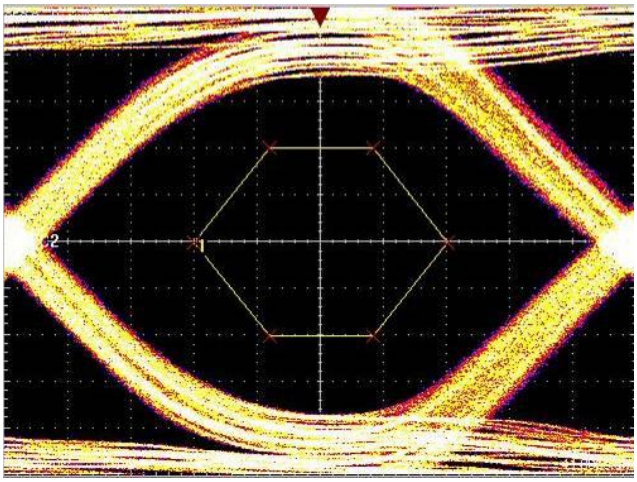

Fig. 14: Micro-D to Nano-D 1m link @ 2.5Gb/s

V. OVERALL CONCLUSIONS

A. **Primary objective**. *The dual channel SpaceFibre Demonstrator Crossover Link based on the modified AXOMACH family*. A preliminary conclusion is this can be a very effective space grade copper-based solution for transmission of 2.5Gb/s SpaceFibre signals up to 10m in length. (or indeed higher speeds) The crosstalk on the connector, while significantly poorer than that of the classic AXOMACH, is generally considered to be acceptable for data rates of up to 2.5Gb/s.

B. **Secondary objective**. *Feasibility of creating space grade miniature multi-channel SpaceFibre links*. Early indications are that it may indeed be possible to design such a link, using the ESA approved Nano-D style of connector. A short paper will be produced on this subject, within the scope of the project for the University of Dundee, to gauge interest in developing such products for future applications.

VI. THANKS AND ACKNOWLEDGEMENTS

Axon' wishes to thank the team at the University of Dundee, Scotland, for their help and input into these two projects.

Radiation tolerant heterogeneous Multicore "system on chip" with built-in multichannel SpaceFibre switch for the “intelligent” signals and images processing systems

Components, Short Paper

Tatiana Solokhina, Jaroslav Petrichkovich, Alexander Glushkov, Ilya Alekseev, Leonid Menshenin
ELVEES RnD Center,
Zelenograd, Russia,
tanya@elvees.com

Sheynin Yuriy, Suvorova Elena
University of Aerospace Instrumentation,
St. Petersburg, Russia
sheynin@aanet.ru

Abstract— **The article presents a Radiation tolerant heterogeneous Multi-core ASIC MC-30SF6 as the SoC (System–on-Chip) for the onboard “intelligent” signals and images processing systems. MC-30SF6 based on a CMOS 180nm Radiation tolerant library and consists of the five ELVEES IP – cores for the processing and compression data with extra performance more than 9 GFLOPs. The SoC design and architecture support fault tolerance against SEU errors. SoC has built-in multichannel multiprotocol SpaceFibre/GigaSpaceWire (SpaceWire-RUS standard)/SpaceWire embedded networking subsystem. The networking subsystem provides multiple ports for high-rate interconnection with combination of SpaceWire/GigaSpaceWire/SpaceFibre links. SoC support four ports GigaSpaceWire/two ports SpaceWire switch. Input and processed data streams transmitted via 1.25 Gbps two multiprotocol SpaceFibre/GigaSpaceWire and four GigaSpaceWire links. Two SpaceWire links (ECSS-E-50-12C) provide data transfer bandwidth from 2 up to 400 Mbps. The MC-30SF6 embedded networking subsystem on the base SpaceWire/GigaSpaceWire/SpaceFibre provide a balance between ASIC throughput and SoC performance especially for the multifunctional micro and nanosatellites systems.**

Index Terms **— Radiation tolerant heterogeneous Multicore ASIC, multiprotocol SpaceFibre/GigaSpaceWire based links**

I. INTRODUCTION

In the spacecraft signal/image processing on-board systems, for example, for the Earth Observation (EO) missions it is necessary to solve several important tasks, including:

1. The task of delivering large amounts of data at high speed from the sensors to the proper processing system;
2. The task of achieving the required performance in the onboard processing system, which in fact determines all the main qualitative characteristics of the optical/radar monitoring or "intelligent" on board processing of the synthesized images with “video analytics”;
3. The compression task of optical/radar image for subsequent storage or for the transmission;
4. The overall management space system task.

Therefore these missions have the highest performance needs for the signal/Image processing and analysis, data reduction and compression. Typical application data types demand both fixed-point and floating point processing capability.

Expected processing power and the key requirements needs for the EO Payloads have been analyzed in an ESA study [1] and can be summarized as follows.

Candidate Payload - MTG IR sounder:

- Peak sensor data rate - 2.2 Gbit/sec;
- Processing power - 10 GOPS, mixed fixed & floating point operations.

Candidate Payload - High Resolution Wide Swath SAR:

- Peak sensor data rate - 500 Gbit/sec;
- Peak processing power - 1000 GOPS.

Even the minimum requirements for the implementation of these tasks in the spacecraft show the global gap between these current (future) needs onboard processing capabilities and the implementation of modern silicon for space applications.

Thus, the limiting factor in the development of modern and advanced high-performance and a high bandwidth on-board signal/image processing systems is the absence of a large selection of space microprocessors for the high-performance space computing, that provided the highly throughput by the links (up to the gigabits) based on modern advanced standards such as SpaceWire and SpaceFiber and its modifications.

This article describes the experience in the creation of an actual high-performance the highly throughput "system-on-chip" MC-30SF6 qualified for space applications with balanced architecture of processing IP-cores and network subsystem of the data exchange between ASIC resources and between multiprocessor networks on the SpaceWire,

SpaceFiber [2] and Giga SpaceWire (SpaceWire-RUS standard) base.

II. THE MC-30SF6 ARHITECTURE

Radiation tolerant Multicore ASIC MS-30SF6 (Fig.1) was developed as the heterogeneous SoC (System–on-Chip) for the "intelligent" signals and images processing systems.

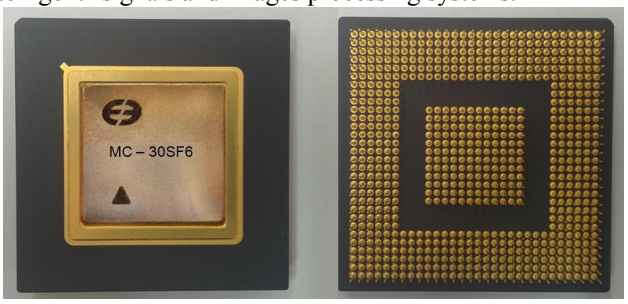

Figure 1. MC-30SF6 chip in the CPGA720 package.

The ASIC consists of five processing MULTICORE platform based IP – cores for processing and compression images with extra performance it is provided with the rich periphery also.

All ASIC processors and accelerators operate independently from each other (each on its own program) and, therefore, represent the three power core "system-on-chip" with MIMD - architecture (MIMD - Multiple Instructions Multiple Data).

The top system manager (CPU) is a standard RISC - processor (RISCore32) with 4-stage pipeline. CPU has EEE-754 compliant Floating Point Unit (FPU) supports single and double precision data types. With Multiply/Divide accelerator CPU provides the addition, multiplication and division operations with single/double precision in floating-point format (maximum issue rate of one 32x32 multiply per clock, 2 cycle multiply latency, 11 cycle divide latency). CPU also has a memory management unit (MMU) on the basis of fully associative address translation buffer (TLB) of 16 double cells, the instruction cache (I CACHE) of 32 Kbytes of data cache (D CACHE) of 32 Kbytes. The programmable MMU provides two operating modes: with TLB (Translation Lookaside Buffer) and FM (Fixed Mapped). On-chip JTAG IEEE 1149.1 Debug Unit support the single stepping and data address/value breakpoints.

MC-30SF6 ASIC was realized to support all architectural solutions, which increased its resistance to failure and fault tolerance. All ASIC memory blocks including the register files in CPU/DSP are protected by Hamming code with single errors correcting and two errors detecting. It was provided the Single event upset mitigation with Triple Modular Redundancy (TMR) for all triggers registers in CPU. The mode can be switched off in order to provide reduce power consumption for the all chip.

As the RISCore32 tools we used gcc 3.2.3., Gcc 4.3.2. To develop and debug programs we created the IDE MCStudio ®, which supports the configuration of our multicore chip based on RISCore32 IP - core and others IP - cores, for example, DSP.

MS-30SF6 contains high-performance Dual DSP cluster from IP – cores library developed by ELVEES. DSP cluster used for digital signal/image processing with fixed and floating point and provides data processing with variable data formats. The 7-stage pipeline and flexible address modes allow realizing algorithms of signal/image processing with high efficiency. Dual DSP IP-core main features:

- 8/16/32/64-bit fixed-point data types;
- 32/bit floating-point data types;
- 16/32/64/128-bit data formats;
- high effective instruction set density: 16 bits per instruction;
- configurable data/program memory size;
- X and Y data memory pointers;
- VLIW-type parallelism: up to 2 computational operations and up to 2 data transfers per cycle;
- SIMD-type parallelism: 128-bit data vectors for 8/16/32-bit data types.

In one cycle DSP cluster can perform (with 140MHz clock):

- 16 op/s with floating point data format (24E8, IEEE754) - 2.24 GFLOPs;
- 16 op/s with integer data format (int32) - 2.24 GOPs;
- 48 op/s with integer data format (int16) - 6,72 GOPs;

In addition to the named three processor cores (CPU and Dual DSP) the "system-on-chip" MS-30SF6 includes two hardware accelerators (FFT and JPEG), operating in parallel.

Additionally, system on chip includes two hardware accelerators: FFT IP - core and JPEG IP-core.

FFT (Fast Fourier Transform) Accelerator:

- Input-output data are carried out in real time, in parallel with processing;
- Entrance/output data for the user are in a direct order;
- For calculations and data storage in a direct order additional memory isn't required;
- The formats of real/imaginary component of the entrance and output data: 32-bit floating point format (IEEE-754 standard), 32-bit integer (additional code), 16-digit integer (additional code). Format of calculations: 32-bit floating point;
- The maximum amount of a directly carried out transformation – 8192, minimum – 16. The limit amount of increased transformation – 256K;
- Performance: for one step 40 arithmetic operations with a floating point (24 additions/subtraction and 16 multiplications) are carried out. That with a work frequency 160 MHz corresponds 6.4 GFLOPs.

The signal/image Compression accelerator (JPEG Encoder) according to the JPEG standard provides Input-output of images is carried out in real time, in parallel with processing. The productivity of the accelerators image compression is:

- one component (Y, by Cb or Cr) with a size of block of 8x8 pixels is processed with a speed of 2,46 pixels for a

step. With a frequency of 200 MHz productivity of compression is equal 490 megapixels/s.

- at three components of the same size of the YCbCr 4:4:4 format productivity of compression is equal to 164 megapixels/s or 75 fps Full HD.

Parameters of the real performance of the processor units are similar to the peak, as input and output data and the intermediate results of processor cores transmitted simultaneously with basic data processing.

RISC IP-core together with the signal and image processing IP-cores provides 9 gigaflops ASIC performance.

The new generation MC-30SF6 applies the ability to turn off unused processor IP cores and other resources such as unused high throughput links. The MC-30SF6 also supports a sleep mode in which it consumes minimum milliwatts of power.

The ASIC has two DDR2 memory ports (1600 MB/s), support DMA transfers between external I/O ports and external memory, have four Multifunctional Buffed Serial Ports (MFBSP) that can act as SPI, I2S, LPORT, GPIO interfaces, Ethernet MAC 10/100, six Space Wire family, USB port, External memory Port (MPORT):

- Data bus - 64-bit address bus - 26 bits;
- Integrated controller of the memory (SRAM, FLASH, ROM, SDRAM);
- Software Configuration for memory blocks and its size.

Input and processed data streams via through eight SpaceWire based family links (six up to the 1.25 Gbps and two up to 400 Mbps) provide a balance between its throughput and SoC performance.

MC-30SF6 also has a dedicated test and debug interface; run the Linux operating system; and have a C /C++ application software compiler for the CPU and DSP cores.

III. THE MC-30SF6 SOFTWARE PLATFORM

The MC-30SF6 software platform basic advantages are:

- The Complete tool set for the fast development and integration of the space signal/image processing applications, includes MCStudio IDE (Integrated Development Environment);
- Full software development kit: optimizing C compilers, advanced multi-core debugger, simulators, application profiler and industry's proven signal, video and image processing Software and Algorithms library;
- Full-featured simulation model allows to start application software developing early;
- Unified programmable DSP core allows to avoid software migration troubles;
- Instruction-level support for all C language data types, including floating-point and complex types, enables effective use of C compiler for different applications, improving time-to-market;
- Standard API – all software components (including signal/image processing and algorithms).

IV. THE MC-30SF6 ASIC EMBEDDED NETWORKING SUBSYSTEM

The MC-30SF6 embedded networking subsystem provides multiple ports for high-rate interconnection with combination of the SpaceWire/SpaceFibre /GigaSpaceWire (SpaceWire-RUS standard) links.

The combination of the SpaceWire based family links (SpaceWire, SpaceFibre and GigaSpaceWire with various speeds and opportunities) provides unprecedented flexibility and scalability for space on-board processing systems with the equal efficiency as for the large distributed digital signal/image processing applications and as stand-alone multifunctional chip based systems for micro and nanosatellites.

The eight MC-30SF6 SpaceWire based family serial high-rate links consist of:

1) Two multiprotocol ports such as SpaceFibre (4 VC, 1250Mbps)/GigaSpaceWire(SpaceWire-RUS); have rates up to 5,10,15 ... (with 5 Mbps increments) ... 125, 312.5, 625, 1250 Mbps);
2) Four ports GigaSpaceWire (SpaceWire-RUS); have rates up to 5, 10,15 ... (with 5 Mbps increments) ... 125, 312.5, 625, 1250Mbps), this ports efficiently combined using six ports internal switch with two SpaceWire ports;
3) Two SpaceWire ports (ECSS-E-50-12C) have rates up to 2-400 Mbps.

It should be noted that MC-30SF6 ASIC SpaceWire links implementation supports the extensions towards next SpaceWire standard revision such as Distributed interrupts and others.

It is also important to note that the GigaSpaceWire ports can provide bandwidth up to the 1250 Mbps, but can operate also in a range of lower data rates, down to 5 Mbit/s. Lower data rates could efficient for longer distances or using older types of cabling.

GigaSpaceWire is in fact a high-rate link for SpaceWire networks, and has the exactly the same Packet, Network layers and the same packet formats that makes the packets routing and switching between any combination SpaceWire and GigaSpaceWire ports straightforward and resource-efficient. The internal switch operates as a SpaceWire routing switch, with routing and switching SpaceWire/GigaSpaceWire packets between any combinations of its ports, in accordance with ports operation modes and the routing table.

Two SpaceFibre links [3] are provided by the multiprotocol network interface controller. The main SpaceFibre link rate in the MC-30SF6 ASIC is 1250 Mbit/s.

In the multiprotocol ports implementation another operation mode is to support the GigaSpaceWire protocol. Such combination of the different types of ASIC links (SpaceWire/SpaceFibre/GigaSpaceWire) and internal switches makes the MC-30SF6 very flexible in building ASIC network interconnection with external processors, nodes, and peripherals with any type of

SpaceWire/SpaceFibre/GigaSpaceWire networks, provide different types of network services.

While SpaceFibre links provide advanced QoS features (very important for the space onboard control systems), the SpaceWire/GigaSpaceWire combination links provide effective and cost-efficient networking for other on-board applications (for example, for the space signal/image processing systems). Such applications may not require SpaceFibre QoS features with an extra cost of the SpaceFibre implementation silicon area. The analysis of the complexity of SpaceFibre implementation is presented in the next section.

Thus MC-30SF6 ASIC is a new generation balanced data processing "system on a chip" of that supports a wide class of space on-board applications ranging from control systems to onboard signal /image processing high-end systems.

As the conclusion from the comparative analysis of the Stereo Computer Vision task application example follows that if DSP "Elcore30" will support clock ~ 140 MHz and ARM Cortex A9 will provide ~ 1 GHz then this task speedup for the ELVEES Single DSP core will be 0.58x and for the Dual DSP core - 1.0x. This means that the Radiation Tolerant MC-30SF6 multicore microprocessor will decide such application task not worse than powerful 1 GHz ARM Cortex A9 microprocessor.

So MC-30SF6 SoC is capable not only to form video streams, but also providing the on-board real-time analysis of multi-megapixel image data with video analytics approach which will provide autonomous on board situational analysis and real time the on-board mission decision-making.

V. THE MC-30SF6 ASIC PROOF ON THE SILICON

The space qualifiable (Radiation Tolerant) ASIC Microelectronics technologies for the area and timing parameters estimations was reviewed [4].

The virtual design of the SpaceWire-RT (SpaceFibre) ASIC IP-Core (digital controller and CML- transceiver) was developed based on the using IP-core Library from the ELVEES ASIC platform MULTICORE including "Soft" and "Hard" IP-cores (8B/1B CODEC, SERDES, PLLs, transceivers). The Netlist and layout of SpaceWire-RT ASIC IP - Core were synthesized on the various qualified for space microelectronic libraries also were used as an assessment for the implementation of the SpaceWire-RT ASIC IP – Core.

During the project, we analyzed the complexity and feasibility of six channels SpaceFiber switch built-in microprocessor with four virtual channels each.

The limitations of implementing (area cost) for the used Radiation Tolerant Libraries not allowed to unify and to implement the all SpaceFibre six high-speed links plus switch, and, moreover, as multiprotocol variant. The total silicon area cost of the six multiprotocol links is about 60.3 mm*2 (including 54, 06 mm*2 for the six SpaceFibre digital controllers, with the Broadcast controller and the mode/state registers block in each).

And, on the other hand, the size of the topology area of a 6-channel switch SpaceFiber with controllers SpaceFibre is 161,46 mm * 2 (excluding space for the block of mode/state registers and the Broadcast processing unit), which occupies more than half the area of the possible size of a microprocessor (306 mm * 2).

On the one hand the SpaceFibre benefits provided high cost area for virtual channels (VCh) and thereby to realize only a small number of SpaceFibre ports for a real space qualifiable ASIC.

And, on the other hand, using a little additional ASIC areas cost we can provide multiprotocol link and an inexpensive opportunity to transmit information through the GigaSpaceWire link supporting low cost data transmission networks, which do not require such properties as virtual channels.

In this MC-30SF6 ASIC project it was created a new innovative multiprotocol port IP - core (SpaceFibre/GigaSpaceWire IP – core) that provide a balanced solution between all advantages in QoS, FDIR and others from SpaceFibre and the simplicity and low cost of implementation from GigaSpaceWire. In this MC-30SF6 ASIC project it was created a new innovative multiprotocol port IP - core (SpaceFibre/GigaSpaceWire IP – core) that provide a balanced solution between all advantages in QoS, FDIR and others from SpaceFibre and the simplicity and low cost of implementation from GigaSpaceWire.

MC-30SF6 ASIC was developed and synthesized on the space qualifiable ASIC technologies base. The chip size is 17.5 mm x 17.5 mm (Fig.2). During the project, we analyzed the complexity and feasibility of 6 - channel SpaceFiber switch built-in microprocessor with four virtual channels each.

GigaSpaceWire (SpaceWire-RUS standard) digital controller IP - core is almost in the 19 times more economical in terms of the silicon area than the SpaceFibre digital controller IP - core (4 virtual channels). For a greater number of channels, this ratio is even more dramatic.

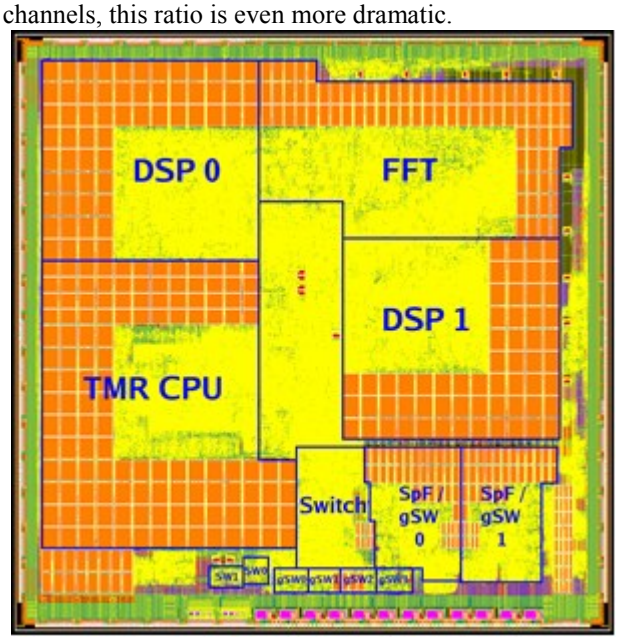

Figure 2. MC-30SF6 chip post-layout area. The chip size: 17.5 mm x 17.5 mm.

This is particularly evident in the SpaceFibre switches /routers ASIC implementations.

From the MC-30SF6 ASIC post-layout area analysis the size of the silicon area for some radiation tolerant MC-30SF6 IP - cores (real layout):

- TMR CPU: 57.00 mm*2;
- DSP (total): 126.11 mm*2, including:
  - One DSP-core: 40.96 mm*2;
  - FFT+JPEG - accelerators : 41.55 mm*2;
- multiprotocol port SpaceFibre (4VC) /GigaSpaceWire (controller): 10.05 mm*2
- SpaceFibre (4VC) part -9,01 mm*2;
- GigaSpaceWire (controller): 0.54 mm*2;
- SpaceWire (Controller): 0.33 mm*2;
- Switch (for the four GigaSpaceWire ports plus two SpaceWire ports): 8.64 mm*2.

The main parameters of the SpaceFibre/GigaSpaceWire CML based transceivers IP-cores, based on the space qualification Radiation Tolerant Libraries, are:

- A wide range of data rates 5, 10, 15… (with discrete of 5)...125, 312.5, 625, 1250 Mbps – for the GigaSpaceWire mode (including multiprotocol links) and - 1250Mbps for the SpaceFibre mode;
- The transmitter and receiver IP blocks dimensions are the same: RX = TX = 0.233 mm*2.

VI. CONCLUSION

Robust, high-performance MC-30SF6 space quality SoC provides power signal/image processing based on the software/hardware, programmable, intelligent Signal/Image processing platform proven on the silicon.

Open and modular MC-30SF6 innovative microprocessors architecture is supported by a set of hardware accelerators and special DSP instructions that can be used if necessary to obtain high performance and unprecedented flexibility in data formation and processing and their subsequent analysis.

MC-30SF6 SoC hardware and software are a unique, programmable, heterogeneous multi-core platform dedicated to addressing the computational needs of the most sophisticated on-board signal/image processing applications as for the single-chip standalone configurations as for the multichip parallel systems using balanced integrated embedded networking subsystem provides multiple ports for high-rate interconnection with combination of the SpaceWire/SpaceFibre/GigaSpaceWire (SpaceWire-RUS standard) links on the ASIC.

REFERENCES

[1]. Next Generation Processor for On-board Payload Data Processing Application ESA Round Table Synthesis, ESA, TEC-EDP/2007.35/RT, October 2007

[2]. S.M. Parkes, C. McClements, M. Dunstan and M. Suess, “SpaceFibre: Gbit/s Links For Use On board Spacecraft”, International Astronautically Congress, Daejeon, Korea, 2009, paper IAC-09-B2.5.8

[3] “D2.1 - SpaceWire-RT Outline Specification”, SPACEWIRE-RT Consortium, 06.09.2012.

[4] “D5.1 - SpaceWire-RT ASIC Implementation Feasibility Summary Report”, SPACEWIRE-RT Consortium, 09.05.2013

Experiences with a SpaceWire Backplane Connector

SpaceWire Components, Short Paper

John-Paul Coetzee, Alan Senior
Space Division
Thales Alenia Space UK (TASUK)
Bristol, United Kingdom
John-Paul.Coetzee@thalesaleniaspace.com,
Alan.Senior@thalesaleniaspace.com

Jørgen Ilstad
European Space Agency, ESTEC
Noordwijk ZH, Netherlands
jorgen.ilstad@esa.int

***Abstract* — Prototype connectors have been developed by Smiths Connectors (Hypertac) for backplane applications, and testing has been performed by TASUK under ESA contract.**

The connector is a modular type with different pin inserts for power, signal and high-speed data, making it suitable for SpaceWire applications.

This paper describes TASUK's experience designing a test setup and using the backplane connector.

***Index Terms*—SpaceWire, backplane, connector, layout**

I. Introduction

The SpaceWire Backplane connector has been designed to be modular so that it is suitable for a range of applications of differing complexity. Common card sizes in space applications are single and double Eurocard (and extended versions) and the SpaceWire Backplane Connector has been designed for these sizes.

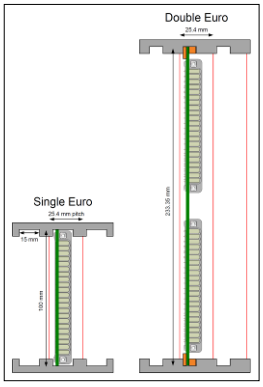

Fig. 1. Typical backplane card sizes

The connector used for TASUK's testing is 97.5mm long. It incorporates two types of data interconnect, one general-purpose and one high-speed coaxial for differential links. It also has two types of power connector, one with 7.5A pins and one with 5.75A pins. Fig. 2. shows the connector. The tightly-packed general-purpose signal pins have a pitch of 1.5mm (the "Signal 10.4 modules", ~1 Gbps). The co-axial differential pins are shielded ("Quadrax modules", ~3 Gbps) [1].

Fig. 2. SpaceWire backplane connector pins

Tests were required in the following areas:

- Physical - check dimension, mass, solderability, mating/de-mating.
- SpaceWire - check SpaceWire performance over both types of data interconnect.
- Eye diagram - capture some basic eye diagrams on data transfers over both types of data interconnect.
- Power - measure power transfer losses over both types of power interconnect.

TASUK have designed and tested a suitable system. This is the first time that such a backplane connector has been used in a real-world SpaceWire application.

This paper focuses on the user experience of the SpaceWire Backplane connector, in particular the layout of the backplane and daughterboards to achieve an extensible, high-speed system. Some rudimentary test measurements were performed but these were not rigorous and are provided here to demonstrate usability of the connector. Rigorous parametric analysis will be performed by Smith Connectors.

II. Test System

The test system consists of a small backplane board and two small daughterboards. The connections across the daughterboards and backplane allow end-to-end testing of the different interconnect types in a representative environment. All connectors and test pins are easily accessible.

Fig. 3. shows the test system with one daughterboard disconnected. In the figure the two micro-D SpaceWire

connectors on the daughterboard have black protective covers. One of these SpaceWire connectors is connected to co-axial differential Quadrax modules, the other to pins within a Signal 10.4 module. The SMA connectors are connected to Quadrax modules.

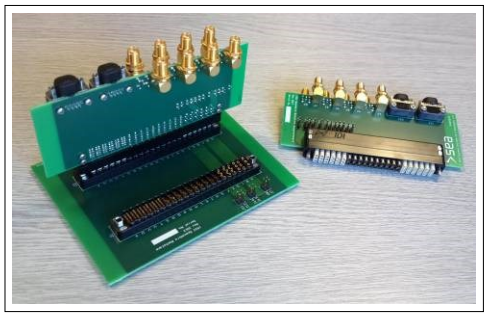

Fig. 3. Test system showing backplane and two daughterboards

For testing all the boards are mounted in a simple frame constructed from extruded aluminium. This provides a robust framework and holds the daughterboards perpendicular to the backplane to prevent any damage from flexing.

Fig. 4. shows the test system in the frame and connected to a Xilinx Virtex 6 evaluation board which was used to generate high-speed signals for eye diagram measurements. Fig. 5. shows the system connected to a laptop running Star-Dundee SpaceWire link verification software to measure data throughput, Fig. 6. shows the same system connected to power supplies and load resistors to test the performance of the power connectors.

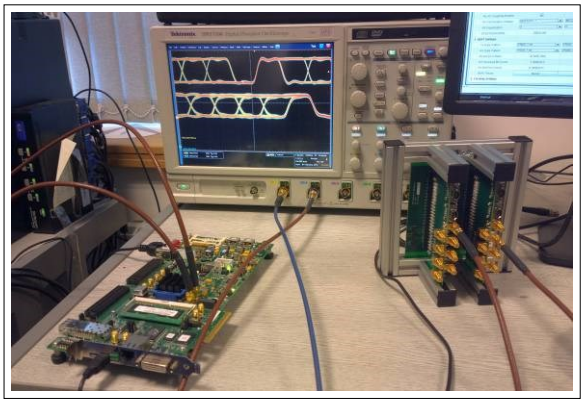

Fig. 4. Eye diagram testbench

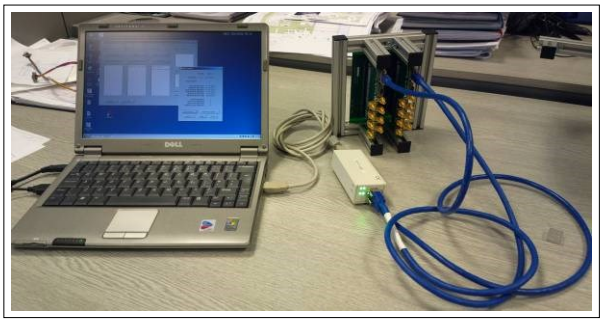

Fig. 5. SpaceWire throughput testbench

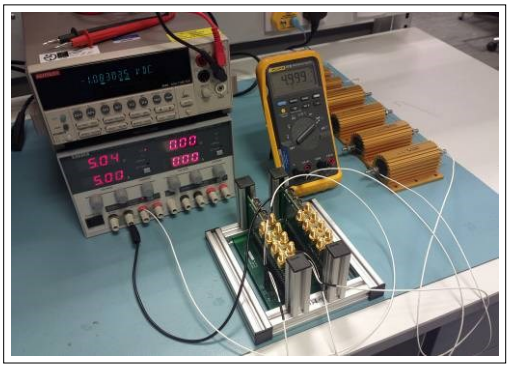

Fig. 6. Power measurement testbench

III. PCB Layout

IIIa. Backplane layout

For optimal performance the backplane design must facilitate a multi-slot backplane high-signal-integrity architecture with flow-through routing. This can be especially challenging in the close pin fields of dense connectors.

A 6-layer FR4 board is used, providing two power layers, two ground layers and two inner signal layers. This stack-up is sufficient to achieve an extensible flow-through architecture.

The layout of the backplane is illustrated in Fig. 7. Fig. 7. also shows how the flow-through architecture allows designs to be easily extended to multiple slots.

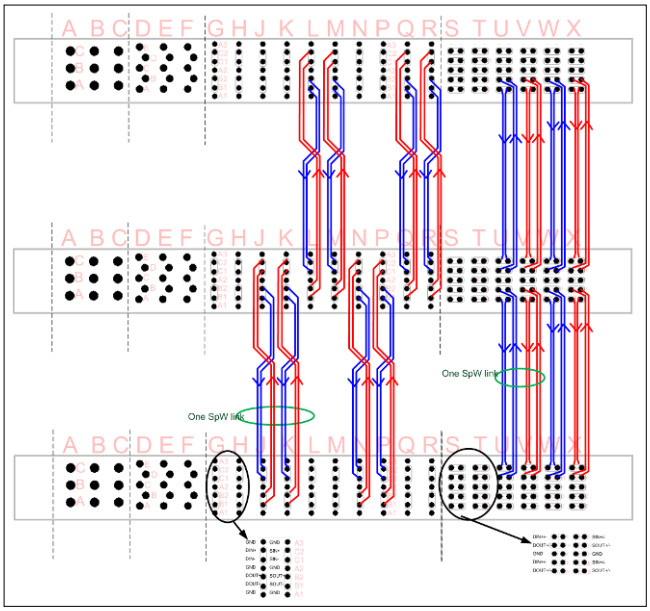

Fig. 7. Backplane layout showing multiple slots

Two important factors in routing differential pairs are (a) keeping the separation of the pairs constant (and matched to 100R differential) and (b) minimising skew between the two traces in a pair. The skew between the differential traces on a backplane adds to the skew on the daughterboards and this must be taken into account when designing high-speed systems which include a backplane.

In TASUK's backplane design all differential signals can be routed, even in the dense pin field of the Signal 10.4 modules (Fig. 8.). The path difference between the individual traces within these differential pairs is approximately 2.2mm. The crossover in the traces connecting the Quadrax modules keeps the path difference between the individual traces down to about 0.2mm.

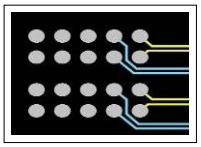

Fig. 8. Backplane layout showing detail in Signal 10.4 modules

IIIB. Daughterboard Layout

The daughterboard layout is shown in Fig. 9. It is also a 6-layer board, all signals are easily routed.

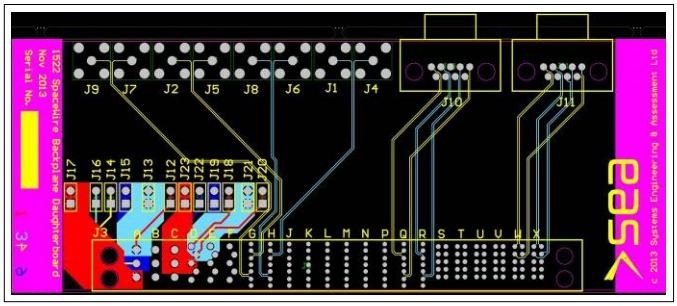

Fig. 9. Daughterboard layout

Fig. 9. also shows the thick traces from the power pins. These reduce resistance and hence both voltage drop along the trace and heat generated. Power is connected using two adjacent 0.1" pins to spread the current across pins. Power is measured using a 4-wire connection facilitated by the provision of voltage measurement pins adjacent to the pins carrying power.

IV. Test Results

The following tests were performed on the SpaceWire backplane assembly.

- Physical tests
- SpaceWire signal tests
- Eye diagram tests
- Power tests

Further tests will be performed by Smiths Connectors.

A. Physical tests

1) Visual inspection

The connectors were visually inspected and no defects were found. Considering these are prototypes the precision of manufacture and the quality of the build was high.

2) Dimensions

The PCB layout was performed directly from the drawings supplied by Smiths Connectors [1]. The prototype connectors fitted the PCBs exactly indicating the correctness of the drawings.

Other basic dimensional checks were performed that demonstrated compliance with the drawings.

3) Mass

The connector is modular and therefore the mass will vary according to the contact types fitted. The drawing [1] indicates a mass of ~53g for the mated pair. The measured masses of the prototype connector are:

- Daughter connector (socket): 35.97g
- Backplane (plug): 15.76g

Thus the mated pair weighs 51.73g.

It is anticipated that a connector fully populated with all Quadrax modules would have the highest mass, though the delta is likely to be small.

4) Solderability

The pins are gold plated, no soldering issues were found.

During the manufacture of a spacecraft unit the contact tails that enter the PCB would be de-golded to prevent solder embrittlement, this process could not be performed by a dip into a molten solder pot due the close proximity and shape of the connector body, however each pin could be de-golded manually.

Soldering on the prototype board proved straightforward. Inspection of the top joint is hampered by the low profile of the connector body but is possible. The pins have a shoulder that prevents a full solder fillet forming on the PCB top surface under the connector body however confirmation of solder flow up the Plated Through Hole (PTH) is possible and the joint meets the inspection criteria imposed by ESA [2].

5) Mating

The connectors have a high mating and de-mating force. Smiths Connectors have stated that the prototype may not have the same insertion and extraction force as the final design. The connector drawing [1] predicts a connector mating force of 100N (TBC). TASUK has not measured this force.

The test system does not have a designed-in method of extraction so de-mating is difficult without applying leverage. Extraction tools and methods may need to be developed when using this connector to avoid damage from non-linear forces.

TASUK have previously manufactured Spacecraft unit modules that use 2 off 144-way Hypertac KMC connectors, these connectors have an insertion/extraction force of 100N, so the module insertion/extraction force could be 200N. Insertion forces of this magnitude are not considered excessive since there are robust surfaces to push against; also the card guides and dedicated connector alignment pins align the card. Extraction is more difficult since a typical board has no features which allow a firm grip. To solve this TASUK designed a mechanical extractor which engages with the PCB and pushes against the front lip of the unit box, pulling the card smoothly from the unit in a controlled manner. For reference a picture of the TASUK designed extraction tool is shown in Fig. 10. .

Fig. 10. Extraction tool designed and used by TASUK on Spacecraft units

B. SpaceWire Signal Tests

1) Setup

Fig. 5. shows the backplane, STAR-Dundee SpaceWire brick (green LEDs showing traffic) and laptop with the test software. The SpaceWire brick was used to investigate the SpaceWire data rate capability of the two types of data interconnect. The blue SpaceWire cables were connected in loopback.

One of the daughterboard's two SpaceWire micro-D connectors is connected via the "Quadrax module" shielded differential co-axial connectors, the other via the "Signal 10.4 module" unshielded plain connectors. Both were tested, with the same results.

2) Results

STAR-Dundee provides two software test utilities, the "spacewire_usb_test" console application and the "SpaceWire Validation Software" Java application.

"Spacewire_usb_test" showed a slowest link speed of 131.07 Mbit/s and "SpaceWire Validation Software" transferred >187Gbytes with 0 errors at an average data rate of 100.648 Mbit/s, see Fig. 11.

These speeds were exactly the same as those measured with the SpaceWire brick in loopback i.e. the speeds are limited by PC software or SpaceWire brick, not by the SpaceWire backplane.

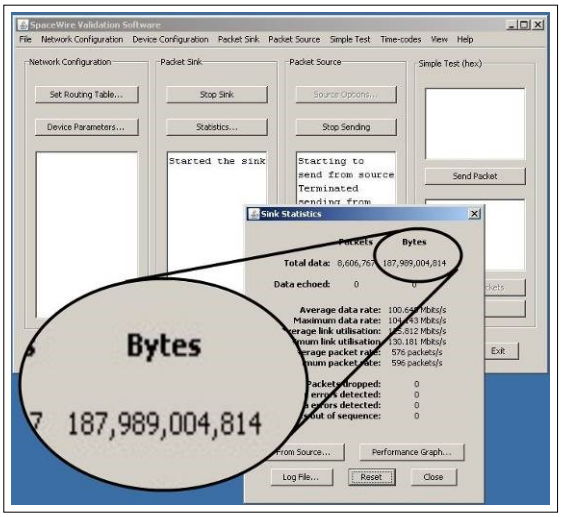

Fig. 11. SpaceWire backplane SpaceWire Validation Software results

C. Eye Diagram Tests

Only rudimentary eye diagram tests were possible owing to lack of suitable equipment; more comprehensive tests will be performed by Smiths Connectors. The oscilloscope was only capable of triggering on a rising edge and not on a recovered clock so these are not true eye diagrams; however the results are indicative of the low distortion levels.

1) Setup

A Xilinx Virtex 6 evaluation board was used to generate high-speed signals. The board has differential outputs with the Tx+ and Tx– signals available on SMAs. See Fig. 4. and Fig. 12. In Fig. 4. the blue cable into Ch1 is connected to the brown cable from the Xilinx board Tx+ using an SMA connector to keep the total cable lengths the same. The connector cannot be seen in the figure.

Bit-Error Rate test code was downloaded from the Xilinx website (UG811 *"ChipScope Pro Tutorial - Using an IBERT*

Core with ChipScope Pro Analyzer"). The code uses the GTX transceivers to perform bit-error rate tests, the bit rate can be set to 625Mbps, 1.25Gbp, 2.5Gbps or 5Gbps. The oscilloscope had a 1Gbps bandwidth so the only data rate which could be tested was 625Mbps.

Eye diagrams were generated for two signals (i) directly from the Tx+ SMA output of the evaluation board (i.e. without the test backplane in the signal path) and (ii) from Tx- via the "Quadrax module" shielded differential co-axial connectors in the SpaceWire backplane.

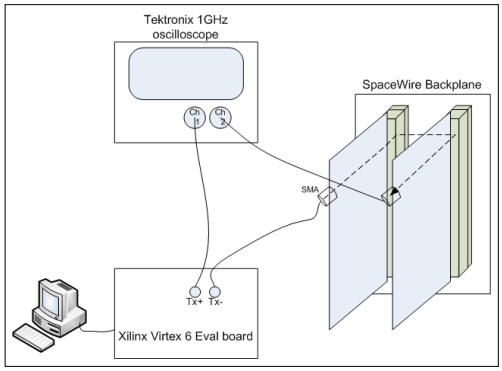

Fig. 12. SpaceWire backplane eye diagram test setup

2) Results

See Fig. 13. The top trace is the direct connection i.e. without the test backplane in the signal path, the bottom trace is via the "Quadrax module" shielded differential co-axial connectors. As may be expected the eye diagram of the signal via the SpaceWire backplane was different from the direct connection.

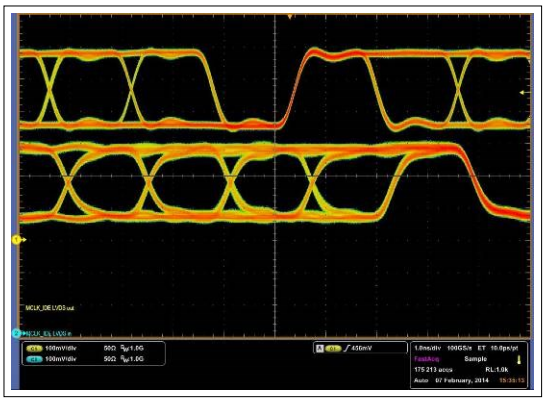

Fig. 13. SpaceWire backplane eye diagram results

D. Power Tests

1) Setup

To simulate power transfer across the backplane, power was applied at one end of the backplane and a low-resistance load connected across connectors on the daughter board. The voltage drop across the backplane was measured using a 4-wire measurement, Fig. 6. shows the testbench.

There are two different types of power connector modules in the backplane (a) "PWR5.75 module" with pins of 5A current-carrying capacity and (b) "PWR3.1 module" with pins of 7.5A capacity. Two measurements were made for each connector type, across different pairs of pins.

2) Results

a) PWR5.75 module pins (5A)

1. Voltage measured = 31.6mV, current = 4.99A
   => Resistance = 31.6/4.99 = 6.3 mΩ
2. Voltage measured = 27.6mV, current = 4.99A
   => Resistance = 27.6/4.99 = 5.5 mΩ

These pins have a specified resistance of 3mΩ [1].

b) PWR3.1 module pins (7.5A)

1. Voltage measured = 32.6mV, current = 7.49A
   => Resistance = 32.6/7.49 = 4.4 mΩ
2. Voltage measured = 32.0mV, current = 7.47A
   => Resistance = 32.0/7.47 = 4.3 mΩ

These pins have a specified resistance of 2mΩ [1].

V. Conclusion

The SpaceWire backplane and daughter board have proved the feasibility of the SpaceWire backplane concept, PCB layout and the use of the backplane components in a real-world application. The application incorporates a representative backplane, two daughter boards and the backplane connectors.

Testing shows that the SpaceWire links routed through two pairs of connectors will run at the maximum possible rate of the SpaceWire Brick.

Eye diagrams of the "Quadrax module" shielded differential co-axial connectors within the connectors at 625MHz show some expected attenuation, and that the link is not creating any unexpected discontinuity or distortion.

Power measurements show that the power pins are performing at their rated capability with minimal losses. It should be noted that the pins used in the connectors are prototype rather than production parts.

The layout of the backplane and daughter boards have shown that a multi-slot backplane with a flow-through architecture is easily achieved, even within the tight restrictions of the closer pin field of the densest connector module types.

A flexible architecture has been shown to be feasible using only 6 layers.

References

[1] Smiths Connectors / Hypertac drawing HYP-6890 (Provisional) dated 10.01.2014.

[2] ECSS-Q-ST-70-08C Space product assurance - Manual soldering of high-reliability electrical connections (6 March 2009)

Missions & Applications (Short)

Service oriented integration of SpaceWire and conventional protocols with reference to SOIS

SpaceWire Missions and Applications, Short Paper

Hiroki Hihara, Asako Terada, Satoko Kawakami, Muneyuki Iwanabe
NEC TOSHIBA Space Systems, Ltd.
10, Nisshin-cho 1-chome, Fuchu, Tokyo, 183-8551, Japan
h-hihara@bc.jp.nec.com, a-terada@vx.jp.nec.com, s-kawakami@bk.jp.nec.com, m-iwanabe@sx.jp.nec.com

Takeshi Takashima, Motohide Kokubun, Tadayuki Takahashi
Institute of Space and Astronautical Science, JAXA,
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan
ttakeshi@stp.isas.jaxa.jp, kokubun@astro.isas.jaxa.jp, takahasi@astro.isas.jaxa.jp

Takayuki Tohma, Takashi Kominato, Kazuyo Mizushima, Kenichi Baba
Space Systems Division,
NEC Corporation
10, Nisshin-cho 1-chome, Fuchu, Tokyo, 183-8501, Japan
t-tohma@bx.jp.nec.com, t-kominato@pd.jp.nec.com, k-mizushima@cb.jp.nec.com, k-baba@dg.jp.nec.com

Takayuki Yuasa
RIKEN The Institute for Physics and Chemical Research
2-1 Hirosawa, Wako, Saitama 351-0198, Japan
takayuki.yuasa@riken.jp

Masaharu Nomachi
Osaka University
1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan
nomachi@rcnp.osaka-u.ac.jp

***Abstract*—Conventional protocols have been integrated with SpaceWire through service oriented approach with reference to SPACECRAFT ONBOARD INTERFACE SERVICES (SOIS). The design framework is based on the definition of determinism provided by SpaceWire-D draft standard in order to keep established services inherited from previous satellite projects. The implementation result is under evaluation in order to establish the consistency with the draft standard of SpaceWire – Plug-and-play protocol. This paper describes the integration approach and the evaluation of implementation experience.**

***Index Terms*— SpaceWire, Networking, SpaceWire-D, Plug and play, SOIS.**

I. Introduction

Next generation spacecraft bus architecture has been established by JAXA/ISAS (Japan Aerospace Exploration Agency/Institute of Space and Astronautical Science) and NEC. Scalability with well-defined interface specification is the main issue for the architecture in order to apply the architecture on wide range of satellites with great flexibility. Various units are to be connected to spacecraft system bus in simple way as plugging in power plugs into outlets.

Service oriented approach was employed for the integration with reference to SOIS concept [1]. SpaceWire [2]/RMAP (Remote Memory Access Protocol) [3] is adopted in the architecture with SpaceWire-D draft standard [4], and conventional protocols have been integrated with SpaceWire exploiting the "New Concept" research project and scientific satellite projects. Almost all of onboard subsystems of ASTRO-H, such as the command/data handling subsystem, the attitude and orbit control subsystem, and four types of X-ray/gamma-ray telescope instruments, are connected to the SpaceWire network using a highly redundant topology [5], [6], [7], [8]. The number of physical SpaceWire links between onboard components exceeds 140 among 40 independent units, and there are more links in intra-component (intra-board) networks. A partial redundant SpaceWire networks with the electronics units developed in ASTRO-H project has been demonstrated in orbit by HISAKI successfully in 2013. Hybrid systems with conventional interfaces and SpaceWire interfaces are also under development in HAYABUSA2 [9] project and ASNARO project [10], [11].

In order to accommodate conventional interfaces, discrete command and telemetry interfaces have been integrated into SpaceWire network by dedicated attachments, and serial digital interfaces as MIL-STD-1553B, UART and dedicated serial transmission protocol for Japanese scientific satellites have been transformed through protocol bridges. The implementation methods for interface specification conversion were investigated through the new concept research, which encompassed software implementation, device implementation and equipment architecture.

The ground station operation scheme using CCSDS framework was also carefully investigated through the

preliminary design phase of ASTRO-H space X-ray observatory in order to maintain the operation manner of previous projects. This development activity had been resulted in service oriented approach by categorizing the network transmission scheme of prior projects with referenced to CCSDS SOIS concept. The flexibility of remote memory access protocol of SpaceWire and the determinism for system integration and test provided by SpaceWire-D draft standard were exploited to accommodate each service, and specific memory space has been assigned as the channel for each service.

The implementation result is under evaluation in order to establish the consistency with the draft standard of SpaceWire – Plug-and-play protocol [12].

II. Integration Approach

Four types of satellite systems have been developed using SpaceWire. They are single system, partial redundant system, fully redundant system, and hybrid system with legacy network as shown in Table 1. In order to encompass these types of satellite systems, four approaches have been carried out for establishing the scalability through the project. They are the establishment of design criteria, the establishment of the standard onboard network architecture based on SpaceWire, the development of standard units and devices, and the service oriented network definition.

Most technology development has been achieved in ASTRO-H project [13]. Design criteria have been established through the development phase of the project, and standard electronics units were also developed. A partial redundant system, which is called as "HISAKI", was developed with the design criteria and standard electronics units have already been demonstrated in orbit successfully. A data handling subsystem in single configuration and an attitude and orbit control subsystem in dual standby redundant configuration are employed for HISAKI. Therefore, single and standby redundant operation scheme of common onboard computer (Space Cube®2) have been demonstrate and validated. PIM (peripheral interface module) is an inherited legacy onboard network for JAXA scientific satellites, and it is used in HAYABUSA2 project with SpaceWire [14]. The preliminary integration test of HAYABUSA2 has been completed and the hybrid system has also been established. In consequence the scalability of the design criteria have been validated.

Standard onboard network architecture based on SpaceWire has been established by ISAS [15] in order to realize scalability. The architecture is based on the functional model of spacecraft, and the functional model of spacecraft is defined with functional objects [16].

TABLE I. Four types of SpaceWire networks

Redundancy	Project
Single system	ASNARO (planned to be launched in 2014)
Partial standby system	HISAKI (SPRINT-A) (launched on 14th September 2013)
Full redundant system	ASTRO-H (planned to be launched in 2015)
Hybrid system	HAYABUSA2 (planned to be launched in 2014)

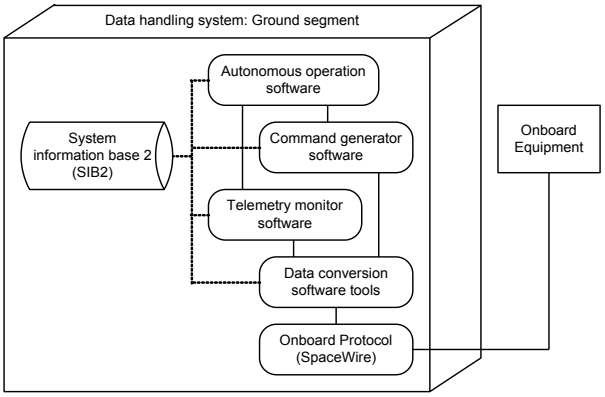

Fig. 1. ISAS functional model development framework

The attribute, operation, event, behavior, and diagnose rule of the functional model are specified in system information base 2 (SIB2). The architecture consists of three sub-architectures, which are physical architecture, functional architecture and protocol architecture. The development framework of the functional model is shown in Fig. 1.

The physical architecture specifies how to configure onboard network systems physically and defines basic physical elements. Any onboard network system will be constructed physically by connecting basic physical elements according to the characteristics and the complexity of the spacecraft. The functional architecture specifies how to configure onboard network functionally and defines basic functional elements. These functional elements are implemented in physical elements. The protocol architecture specifies how to connect physical and functional elements with communications protocols and defines a set of standard protocols to be used.

By using this architecture, the basic portion of onboard systems will be developed by selecting appropriate standard components and connecting them. The difference in the size of different spacecraft will be reflected in the number of units used in each spacecraft.

top: Space Cube2, bottom left: SpaceWire router, bottom right: data recorder

Fig. 2. Standard components for SpaceWire network

As for the development of standard components, Space Cube2 onboard computer, SpaceWire router and Data Recorder (mass storage unit) have been developed with SpaceWire based on the architecture. Figure 2 shows those three standard components.

The service oriented network definition refers to SOIS. Communication services are defined in reference to SOIS sub network services, and two design criteria are established. The criteria are independent on data link layer and physical layer, so both SpaceWire and legacy protocols refer to them.

TABLE II. TELEMETRY COLLECTION FORMAT FOR EACH COMMUNICATION SERVICE

Communication service	Collection format	
	Space Packet	Raw data through RMAP
Essential housekeeping data		X
Auxiliary housekeeping data		X
Housekeeping packet data	X	
Response value telemetry		X
Memory dump data	X	
Notification	X	
Acknowledge	X	
User request		X
Payload correction data		X

Fig. 3. MIL-STD-1553B interface attachment for Space Cube2

Fig. 4. Payload interface unit (PIU)

Prioritization, segmentation and blocking are defined in telemetry/command design criteria, and SpaceWire and legacy network use the same design criteria. Retry and redundancy are specified in network design criteria, which is dependent on each data link layer protocol such as SpaceWire.

Two types of data collection scheme are provided for communication services. One is to collect Space Packet directly from each target, and the other is to collect raw data in order to make Space Packet by initiator. The raw data is collected through multiple transaction of RMAP. Telemetry collection format for each communication service is shown in Table 2.

III. PROTOCOL BRIDGE DEVELOPMENT

In order to incorporate units with legacy communication interfaces into SpaceWire networks, two types of protocol bridges have been developed.

One is an attachment for an onboard computer. The attachment is connected to a CPU base module. Either SpaceWire or other protocols as PCIbus® [17] are used to connect the attachment to a CPU base module. SpaceWire active backplane is used in case that SpaceWire is used for inter-module communication. Figure 3 shows an attachment for Space Cube2, which adopts MIL-STD-1553B.

The other type of the protocol bridge is an independent unit, which is called as a payload interface unit (PIU). Figure 4 shows an example of PIU. The PIU is used for dedicated

interface as communication equipment as well as discrete interfaces for sensors and actuators.

IV. Design Framework

The protocol layer of onboard and space link communications shown in Fig. 5 [18], [19]. Three upper layers are common for both space link communication and onboard communication. As for onboard data bus, inherited network design from previous scientific satellites had the access protocol whose image is close to memory access image. A linear address space encompasses network-wide access, and communication services are tied to reserved address space.

The upper protocol layers [20], which include RMAP and SpaceWire-D, are independent on the data link layer and physical layer. The notion of RMAP is used both for SpaceWire network and legacy network. The characteristics of physical layer affects the transmission speed and latency. Therefor SpaceWire-D is essential for designing both SpaceWire network and legacy network. The time slot design criteria and latency definition scheme of SpaceWire-D are incorporated into the definition of our SpaceWire network design criteria [21] as well as inherited legacy network design criteria. In consequence, the design flow of all system shown in Table 1 is the same. The maximum transaction numbers in one time slot and the latency performance between an initiator and a target are essential to system performance, so the standardized design criteria for the time slot and latency specified in SpaceWire-D draft standard are adopted both for SpaceWire networks and legacy networks.

V. The Evaluation of Implementation Experience

Since our application layer protocol and segmentation/ blocking scheme is independent on data link protocol layer and physical layer [22], the functions are implemented as the upper layer on SpaceWire and RMAP.

In addition to that, we adopted two draft standards, which are SpaceWire-D and SpaceWire Plug and Play. SpaceWire-D had been almost established during the design phase of ASTRO-H project, and the draft B version was applied to our SpaceWire network design creteria. SpaceWire Plug and Play specification had been in the early stage of its establishment during the design phase of the projects shown in Table 1, and the notion of the specification has been reflected both on the SpaceWire netrwork design criteria and each electronics unit interface design specification.

In accordance with our experiencs, SpaceWire-D is useful for system design phase, because the transaction performance within a time slot is essential for system perforrnance. The common design flow for the network interface was established with the standardisation for transaction and latency definition which is quoted from the SpaceWire-D draft specification. The latency definition was also useful for assigning time out duration on each SpaceWire router.

The notion of SpaceWire Plug and Play was useful for component design phase and system test phase. Common access scheme for the status of each unit is useful for diagnosing the status and configuration information of each electronics unit during the system test. The ownership is tied to network region or network domain in our design, and SpaceWire time code and system time indicator [23] belong to the ownership [8].

Acknowledgment

Authors thank ASTRO-H project people for their precious suggestions for implementing real-time system for a wide range of scalable system design as well as hybrid system design with legacy interface specifications.

References

[1] The Consultative Committee for Space Data Systems, CCSDS 850.0-G-1, “SPACECRAFT ONBOARD INTERFACE SERVICES”, June 2007.

[2] European Space Agency, ECSS-E-ST-50-12C, "Space engineering, SpaceWire - Links, nodes, routers and networks", 31, July 2008.

[3] European Space Agency, ECSS-E-ST-50-52C, "Space engineering, SpaceWire – Remote memory access protocol", 5 February 2010.

[4] Space Technology Centre, School of Computing, University of Dundee, “SpaceWire-D, Deterministic Control and Data Delivery Over SpaceWire Networks“, April 2010.

[5] Tadayuki Takahashi, et al., ”The ASTRO-H Mission”, SPIE, 7732, 77320Z, 30 July 2010.

[6] Tadayuki Takahashi, et al., "The ASTRO-H X-ray Observatory", Proceedings of SPIE, 8443 (2012)

[7] Tadayuki Takahashi et al., "The ASTRO-H X-ray Astronomy Satellite'', Proceedings of SPIE, 9144 (2014)

[8] Takayuki Yuasa, Tadayuki Takahashi, Masanobu Ozaki, Motohide Kokubun, Masaharu Nomachi, Hiroki Hihara, Kazunori Masukawa, “A Deterministic SpaceWire Network Onboard the ASTRO-H Space X-ray Observatory", International SpaceWire Conference 2011, 8-10 November 2011, p.348-351.

[9] Hiroki Hihara, Kaori Iwase, Junpei Sano, Hisashi Otake, Tatsuaki Okada, Ryu Funase, Ryoichi Kashikawa, Isamu Higashino, Tetsuya Masuda, “SpaceWire-based thermal-infrared imager system for asteroid sample return mission HAYABUSA2,” J. Appl. Remote Sens. 8 (1), 084987 (April 28, 2014); doi: 10.1117/1.JRS.8.084987

[10] Toshiaki Ogawa, Yusuke Kobayashi, Shoichiro Mihara, Koichi Ijichi, and Hideyuki Hamada “Outline and Progress of ASNARO (Advanced Satellite with New System Architecture for Observation) Satellite System”, 8th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 04 – 08 April 2011.

[11] Hiroki Hihara, Toshiaki Ogawa and Kenji Kitade, “NEXTAR: Small Satellite Bus Based on SpaceWire Deterministic Implementation”, International SpaceWire Conference 2011, 8-10 November 2011, p.344-347.

[12] European Space Agency, ECSS-E-ST-50-54C draft, "Space engineering, SpaceWire – Plug-and-play protocol", 22 March 2013.

[13] Takayuki Yuasa, Tadayuki Takahashi, Masanobu Ozaki and Motohide Kokubun, “A Deterministic SpaceWire Network Onboard the ASTRO-H Space X-Ray Observatory”, International SpaceWire Conference 2011, 8-10 November 2011, p.348-351.

[14] Hiroki Hihara, Koutarou Moritani, Ryu Funase, Tetsuya Masuda, Hisashi Otake, Tatsuaki Okada, "Intelligent Navigation System with SpaceWire for Asteroid Sample Return Mission HAYABUSA2", International SpaceWire Conference 2013, 10-14 June 2013, pp.308-311.

[15] Takahiro Yamada, and Tadayuki Takahashi, “Standard Onboard Data Handling Architecture Based on SpaceWire”, International SpaceWire Conference 2008, 4-6 November 2008, p.253-256.

[16] Takahiro Yamada, GSTOS 201-0.7a, “Functional Model of Spacecraft (FMS)”, 15 September 2009.

[17] PCI Special Interest Group, “PCI Local Bus Specification2, Revision 2.2”, 18 December, 1998.

[18] NEC Corporation, ASTH-111/SP-111, “Telemetry/Command Design Critria”, 12 October 2010.

[19] Takahiro Yamada, GSTOS 200-0.10a, “Spacecraft Monitor & Control Protocol (SMCP)”, 15 September 2009.

[20] NEC Corporation, ASTH-112/SP-112, “SpaceWire Network Design Criteria”, 15 November 2010.

[21] SpaceWire User’s Group, Japan, “SpaceWire Network Design Guideline”, Version 1.0, 13 May 2010.

[22] Takahiro Yamada, “Proposal for Defining Standard Services Over SpaceWire –Revision A -”, The sixteenth SpaceWire working group meeting ESTEC, Netherlands, 22 March 2011.

[23] Aeroflex Gaisler AB, “High Accuracy Time synchronization over SpaceWire Networks - update”, April 2012.

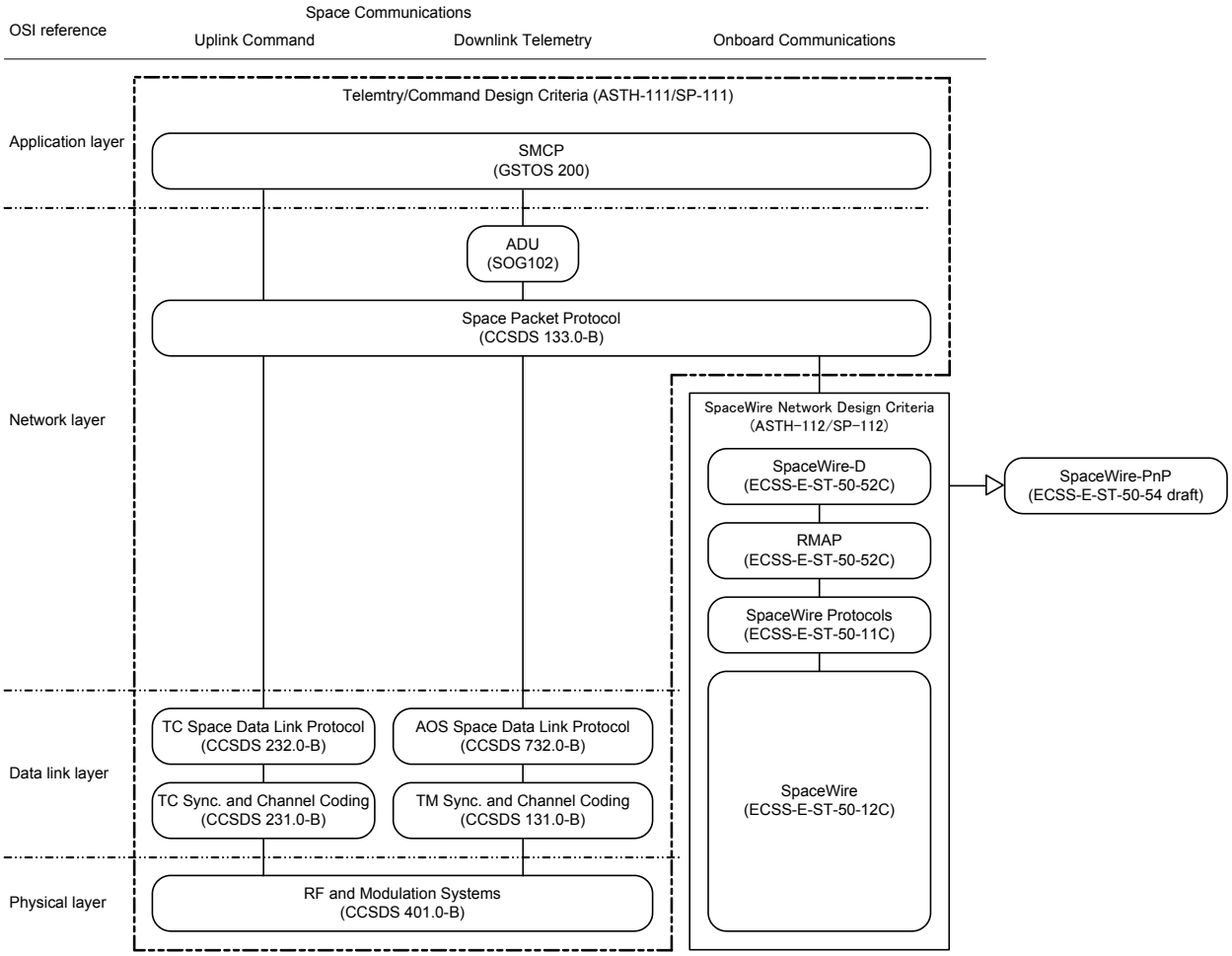

Fig. 5. Standard Platform Protocol Layer

Frequency Calibration of the SWI Instrument on-board of JUICE using SpaceWire Time-Codes

SpaceWire Missions and Applications, Short Paper

Martin Suess
ESTEC, European Space Agency
Noordwijk,
The Netherlands
martin.suess@esa.int

***Abstract*—The Sub-mm Wave Instrument (SWI) is one of the scientific instruments on board of the JUpiter ICy moons Explorer (JUICE) mission. It is a sub-millimeter wave spectrometer with a very high spectral resolution. In order to calibrate the location of the spectral bands the frequency of the instrument's ultra-stable oscillator (USO) has to be monitored to very high accuracy. The instruments on board of JUICE are connected through a SpaceWire network which is used for the transmission of the scientific data to the on-board memory as well as to control the instruments and for the transmission of instrument housekeeping data. The objective of this paper is to investigate if SpaceWire Time-Codes can be used to calibrate the USO frequency of the SWI to the required accuracy. It presents an end-to-end budget of the elements contributing to the error of the frequency calibration with the reference on ground.**

***Index Terms*—SpaceWire, Time-Codes, frequency calibration, SWI, JUICE**

I. Introduction

The JUpiter ICy moons Explorer (JUICE) mission has been selected as the first large mission to be implemented in ESA's Cosmic Vision Program. The objective of JUICE is to investigate the Jovian system focusing on its ice covered moons Ganymede, Europa and Calypso [1].

Amongst the 11 scientific experiments and instruments on board there is the Sub-mm Wave Instrument (SWI). Its development is led by the Max Planck Institute for Solar System Research in Germany. The SWI is a spectrometer and radiometer working in the frequency range of 530 to 601 GHz with a spectral resolution of up to ~10^7. One objective of the SWI is to measure a wind speed and temperature profile in the Jupiter stratosphere and troposphere.

The measurement principle used is to sense the Doppler shift and shape of specific absorption lines of molecules in the Jupiter stratosphere. For this the location of the measured spectral bands has to be known with a very high accuracy. An Ultra-stable Oscillator (USO) is used as the frequency reference in SWI from which all other frequencies in the instrument are derived. In order to meet overall accuracy requirements the frequency of this USO has to be known with an accuracy better than $1.7 \cdot 10^{-8}$.

II. USO Frequency knowledge Requirement

The measurement of Doppler shifts of the absorption lines caused by atmospheric motions relies on a precise absolute frequency calibration of the observed spectra. SWI's frequency scale is derived from the USO frequency.

The required frequency knowledge is derived from the Doppler shift accuracy measurement requirement. To allow atmospheric wind measurements from observations of Doppler shifted spectral lines with a systematic error of less than 5 m/s due to a drift in the absolute frequency scale the relative error must be less than $\Delta v/v$ = (5 m/s)/c = 1.7×10^{-8} where c is the speed of light. This relative error corresponds to a 10 kHz frequency error at 600 GHz.

The USO used for SWI must provide a frequency signal with a very low phase noise as its frequency is multiplied by a large factor to mix down the 600 GHz signal received. The long term stability of this type of low phase noise oscillators is somewhat limited and therefore the frequency drift of the USO needs to be monitored on a regular basis of about once a month.

Ultimately this frequency has to be calibrated against an absolute atomic clock frequency reference on ground. The regular monitoring can be done by comparing the SWI USO frequency to a second, more long term (or sufficiently known) stable oscillator on board of the spacecraft or to measure the SWI USO frequency using on the Mission Elapsed Time (MET) which is the time reference held in the On-Board Computer (OBC). At the same time this MET must be correlated with an highly accurate time reference on the ground.

The accuracy of this second option is investigated in this paper. It is further investigated if the SpaceWire links using SpaceWire Time-Codes can be used to perform this frequency measurement or if a special time signal transferred between the OBC and SWI over a dedicated line is needed.

In paragraph III the accuracy of a frequency counter to measure the USO frequency is derived and compared with the time jitter introduced by SpaceWire Time-Codes. In paragraph IV the process used for time synchronization between spacecraft and ground is explained. In paragraph V the

achievable time correlation accuracy is compared with the time correlation performance which has been achieved in the Gaia mission as example.

III. Analysis of the SWI Frequency Monitoring Accuracy Requirement

The frequency of the oscillator has to be measured with an accuracy better at least an order of magnitude better than the required frequency accuracy of 1.7×10^{-8}. One way of measuring the frequency of an oscillator is to use a frequency counter. This is to count the number of cycles in a fixed and well known period of time T_G. This period of time T_G is called the gate time. For a given gate time T_G and the counter value N the frequency f is calculated by $f=N/T_G$.

Due to the statistical uncertainty of how the counted events are falling relative to the beginning and the end of the gate time window, the counter value N will differ by one count in 50% of the measurements:

$$2\Delta f = f_2 - f_1 = (N+1)/T_G - N/T_G = 1/T_G \quad (1)$$

This corresponds to a principle frequency uncertainty of a frequency counter relative to the mean frequency $(f_2+f_1)/2$ of half a clock period: $\Delta f=1/(2T_G)$. On the other side this means that by extending the gate time the frequency measurement accuracy can be greatly improved to match that accuracy requirement.

The requirement for SWI is on the frequency knowledge $\Delta f/f=1/(2T_G f) < 1.7\cdot10^{-8}$.

With an USO frequency of 10 MHz and if T_G is perfectly known this would require a time gate of at least 2.94 seconds. Any error on the knowledge of the gate time increases the principle uncertainty of the frequency counter measurement.

If the possible error of the gate time T_G is $\pm\Delta T_G$ the relative frequency error $\Delta f/f$ can be calculated as follows:

$$2\Delta f = f_2 - f_1 = \frac{N}{T_G - \Delta T_G} - \frac{N}{T_G + \Delta T_G} = \frac{2\Delta T_G N}{(T_G - \Delta T_G)(T_G + \Delta T_G)} \quad (2)$$

$$\frac{\Delta f}{f} = \frac{\Delta T_G N}{(T_G - \Delta T_G)(T_G + \Delta T_G)} \frac{T_G}{N} = \frac{\Delta T_G}{T_G - \frac{\Delta T_G^2}{T_G}} \cong \frac{\Delta T_G}{T_G} \quad (3)$$

The time distribution mechanism using SpaceWire time-codes introduces a certain mean latency and a jitter. If the begin and the end of the gate time is signaled using SpaceWire time-codes only the jitter needs to be considered as the mean latency stays the same. This jitter which has been investigated in [3] is dependent on a number of parameters. As one example for a link speed of 60 Mbps the jitter introduced by a single link has a standard deviation of 39 ns. Consequently the standard deviation of the gate time will be 55 ns which is the RSS summation to the standard deviation of the two statistically independent time-code arrival events. Just looking at this error contribution the necessary gate time to achieve relative frequency knowledge $\Delta f/f= \Delta T_G/T_G< 1.7\cdot10^{-8}$ is at least 3.24 seconds. In order to get the final figure for the required gate time length all contributors to the gate time error have to be summed up in the RSS sense. As the frequency counter gate time uncertainty of half a period at a USO frequency of 10 MHz corresponds to 50 ns, the combination of this error with the SpaceWire time-code jitter of 55 ns results in a standard deviation of 74.3 ns. Consequently a gate time T_G of at least 4.37 seconds shall be used.

In order to improve the statistical certainty the 3σ error value should be used to calculate the actual gate time.

IV. Time Synchronisation between Spacecraft and Ground

The On-board Computer (OBC) has a counter derived from free running clock. This counter represents the Mission Elapsed Time (MET). The MET is the basis for the execution of the on-board time lines, time tacking of telemetry, generation of SpaceWire time-codes and other time based functions. In order to guarantee the proper spacecraft operation the MET has to be related to the UTC time reference on ground. This is done by means of correlation. Commonly the correlation between MET and Universal Coordinate Time (UTC) is performed as follows:

The MET counter is read out and sampled every time a telemetry frame is sent to the ground. This sampling is triggered by the telemetry frame generator whenever the first bit of the Attached Synch Marker (ASM) of the TLM frame is generated. The sampled time value is then transmitted together with other information in the following TLM frame to the ground.

On the ground the digital processor demodulates the signal and time stamps every frame received based on the local UTC. This time stamp is generated upon the arrival of the leading edge of the first data bit after the ASM. The time information is memorized and “attached” to the frame for post-processing evaluation together with ranging data.

In order to correlate the MET with UTC the time delay between the sampling of the MET on board and time stamping of the received frame on the ground needs to be compensated. There are a number of contributors which have to be take into account. Some are static, others change dynamically.

V. Time Correlation Process of the GAIA Mission

In order to illustrate what level of time correlation can be achieved in scientific missions the Gaia mission [4] is presented as reference. For the highly accurate star position measurements performed by Gaia the precise correlation between MET which is called Spacecraft Elapsed Time (SCET) in the Gaia case and UTC is of outmost importance [5].

There are actually two time scales which are defined for used on-board Gaia:

- the On-Board Time (OBT) for the science packets time stamping and
- the Spacecraft Elapsed Time (SCET) for the platform generated data time stamping.

Gaia follows the a one-way time correlation procedure which has been described in paragraph IV. In regular time intervals a time report is generated on-board and transmitted to ground. Once the event is generated on-board, it is used to sample both the OBT and SCET (actually sampled exactly on the rising edge of the first bit of the related frame Attached Synch Marker (ASM)). The arrival date of this event on ground is time tagged in UTC.

When the first leading edge of the transfer frame ASM is determined by ground a time stamp (UTC) is applied at the ground station. The GPS linked ground station Maser is therefore time stamping the frame after the following delays:

- delays between the ASM event occurrence and the on-board time taken to record and transmit;
- the delay between the sending of the event and its arrival on ground (in propagation time);
- delays between the arrival on-ground of the signal and the on-ground recorded time;
- the relationship between the station time reference and a known time standard.

To meet the timing accuracy requirements of the Gaia mission the on-board time is generated from a single highly accurate Rubidium atomic master clock in the Clock Distribution Unit (CDU). The main clock generates the OBT which is used directly to time stamp the scientific data generated in the Gaia Focal plane. Additionally, the SCET, generated within the OBC called Central Data Management Unit (CDMU), is kept in synchronisation with the master clock by means of a pulse per second reference and a Numerically Controlled Oscillator (NCO).

Before this synchronisation both, the SCET and OBT are free running counters. Even after the synchronisation is achieved the OBT and the SCET are not aligned, only synchronised, so there is still a fixed offset between the two time lines.

The final OBT-SCET-UTC correlation accuracy depends on the precise determination of the free space transmission delay between the satellite and ground. The residual error in this delay calculation depends heavily on the knowledge accuracy of the Gaia orbit.

The mission requirement for the end to end time correlation accuracy is 1.7µsec with an on-board contribution of less than 1µsec.

For the GAIA mission the end-to-end OBT-UTC time correlation budget as shown in TABLE I has been established.

The expected performance is clearly much better the end-to-end time correlation requirement of 1.7 µs. In order to achieve this time correlation accuracy a number of measures and tests had to be put in place calibrate internal delays and to control their variation. The effort made in this respect for the Gaia mission goes beyond what is normally applied for missions with a not so demanding time correlation accuracy requirement.

TABLE I GAIA ERROR CONTRIBUTIONS FOR TIME CORRELATION ACCURACY

	Description of error contributions	*Error size*
1	On-board delays	300 ns
2	Propagation delays (75m error, restituted orbit)	250 ns
3	Tropospheric correction	1 ns
4	Station delay error (daily range calibrated)	1 ns
5	Demodulator jitter/quantization	57 ns
6	Correction to Attached Synch Marker (ASM) bit 0 (inc. puncture code variations)	100 ns
7	Intermediate Frequency Modem System (IFMS) sync to Inter-Range Instrumentation Group (IRIG) B	100 ns
8	Station time sync to GPS (corrected maximum)	200 ns
9	Station GPS to GPS master clock sync	80 ns
	Total RSS Error	**414 ns**

VI. IMPLICATIONS OF THE GAIA RESULTS FOR THE JUICE MISSION DESIGN

It has been demonstrated by GAIA that a very accurate level time correlation between the MET used on board and UTC on ground can be achieved. This requires a very tight control of the absolute time delays on board but also in the ground station during the duration of the mission.

For the calibration of the USO frequency of the SWI instrument absolute time delays are not relevant to obtain a good performance. It is only the change in time delay between the beginning and the end of the calibration measurement, the Gate Time, which contributes to the error. Some of the important error contributors listed in

TABLE I can be assumed to be stable during the calibration measurement period. Other more noise like terms may still contribute to the calibration error.

The error contributors in lines, 1, 5, 6 and 7 may still play a role for the short term time delay error. A realistic requirement could be that this short term time error should be less than 500 nsec.

In order to meet the frequency monitoring requirement of the SWI instrument the gate time has to be chosen long enough relative to the end-to-end time delay variation. In this particular case in accordance with equation (3) the gate time should be longer than 29.7 sec to achieve a frequency calibration accuracy better than $1.7 \cdot 10^{-8}$

The important result here is that the timing jitter in measurement of the MET interval trough the space to ground link is approximately 10 times higher than the gate time error introduced through the frequency counter and the SpaceWire time-code jitter. In this case the option to use a dedicated line using a dedicated line between the OBC and the SWI for the gate time signal distribution will reduce the required calibration measurement time by 0.17 sec which is insignificant. The use of SpaceWire time codes instead will help to reduce the number of interfaces and the required harness mass.

It should be further noticed that an end to end time correlation accuracy of less than 2 µsec like demonstrated by the Gaia mission can only be achieved with significant effort not only on board but also in the ground system. The SpaceWire time code jitter size as reported in [3] has to be set in relation time correlation accuracy. In most cases these errors will only be minor contributor to the overall error.

In some cases there may be a benefit to synchronize the different on-board times and to measure or remove the offset between them as explained in [6].

For some special type of instruments the synchronization requirement between on board clocks may be much higher than the achievable space to ground correlation accuracy. Even if this is not the case the synchronization between the clocks may be a significant system simplification as the time correlation with only the one reference on board clock is needed. In other cases the measurement of the clock offset due to the time code latency may be one important measurement to perform the time correlation.

VII. CONCLUSIONS

This paper investigated the possibility to use SpaceWire time codes on-board of the JUICE mission to calibrate the USO frequency of the SWI instrument. When measuring a frequency with a frequency counter any uncertainty in the knowledge of the measurement period, which is called gate time, is directly contributing to the frequency measurement error. It has been shown that the gate time uncertainty introduced by the time-code jitter at a link speed of 60 Mbps is of the same order of magnitude as the inherent uncertainty of a frequency counter measuring the frequency of a 10 MHz USO. In comparison the gate time uncertainty introduced by the space to ground link is approximately 10 times higher. Fortunately all the errors sources contributing to the gate time uncertainty can be compensated by increasing the gate time length until the required frequency measurement accuracy is met. In the investigated case the required gate time length is determined by the time jitter in the space to ground link.

It is concluded that SpaceWire time-codes can be used without problem to indicate the start and the stop of the gate time. A dedicated line for a time signal between the on-board computer and the SWI instrument does not improve the measurement accuracy or reduce the required measurement time significantly.

The analysis of the time correlation accuracy achieved in the Gaia mission allows further to draw even some wider conclusions for the use of SpaceWire time codes for the time distribution on board of spacecraft. It has been shown that the achievable time correlation accuracy is dominated by the time jitter of the space to ground link as well. In comparison the jitter introduced by the SpaceWire time codes contributes only little to the overall time correlation error budget. This result advocates the use of the SpaceWire network and SpaceWire time codes for the on-board time distribution which will reduce the number of on-board interfaces and required harness.

[1] JUpiter ICy moons Explorer (JUICE): An ESA mission to orbit Ganymede and to characterise the Jupiter system, Grasset, O. et. al, Planetary and Space Science, Volume 78, April 2013, Pages 1–21.

[2] The Submillimetre Wave Instrument on JUICE, P. Hartogh et. al., European Planetary Science Congress 2013.

[3] SpaceWire Time Code Latency and Jitter, M. Suess, F. Siegle, Proceedings SpaceWire Conference 2013, 11-13 June 2013, Gothenburg, Sweden.

[4] The Gaia mission science, organization and present status, L. Lindegren et. al., Proceedings IAU Symposium No. 248, 2007.

[5] GAIA.EST.TN.28135, Gaia System Time Correlation Budget.

[6] SpaceWire time distribution protocol implementation and results, A. Sakthivel et. al., SpaceWire Conference 2014, Athens, Greece.

How RMAP improves in-flight update of on-board software via SpaceWire

SpaceWire Missions and Applications, Short Paper

Holger Michel, Adrian Belger, Björn Fiethe, Tobias Lange, Harald Michalik
Institute of Computer and Network Engineering
Technische Universität Braunschweig
Braunschweig, Germany
michel@ida.ing.tu-bs.de

Martin Kolleck
Max Planck Institute for Solar System Research
Göttingen, Germany

***Abstract*—Modern space probes such as Solar Orbiter employ a SpaceWire network to connect to on-board computer (OBC), solid state mass memory (SSMM), and scientific instruments. Management of SpaceWire links within scientific instruments is typically performed by a data processing module (DPM) featuring a space qualified processor that is executing on-board software. To adapt to changing mission requirements, account for failures and fix possible software bugs, the ability of uploading and patching instrument software is mandatory. However uploading and over-writing of the software's boot image cannot securely be performed by the software itself. If over-writing the boot image fails, the remaining image might be corrupted. So the processor may not be able to reboot successfully and no further upload would be possible. Therefore reception of uploaded patches must be performed by an independent entity. Currently, this is accomplished by a dedicated boot loader in separate memory area, to be qualified according to ECSS criticality category B. This boot loader processes uploading of patches and copies them to the second boot area, where the actual software including the operating system is stored. Due to the opportunity of modern processors to handle SpaceWire RMAP accesses (e.g. SpW-RTC, UT699, GR712RC [1], or upcoming NGMP [2]), it would be possible to perform uploading and patching of the instrument software independent of software execution using RMAP. This would dramatically simplify the development, eliminate the need for a class-B qualified boot loader, and will inherently improve reliability, as reception of patches would entirely be performed by hardware. This paper presents a possible update and patch process for boot images using hardware based RMAP features. Furthermore implications of the standard ECSS services affecting such patching routines are discussed.**

***Index Terms*—SpaceWire, RMAP, in-flight update, boot loader, CCSDS PUS.**

I. Introduction

In modern scientific instruments a data processing module (DPM) handles processing of science data, instrument control, and telecommand (TC), telemetry (TM) and housekeeping (HK) communication. To receive TC and send TM and HK to the on-board computer (OBC) and solid state mass memory (SSMM) modern space probes such as Solar Orbiter or BepiColombo are equipped with a SpaceWire network. The DPM typically features a processor running instrument software to process TC, generate TM and HK, and perform instrument control. The instrument software is stored in a boot memory contained in the DPM and is booted automatically on power up. Due to changing scientific mission requirements or handling of unexpected difficulties with the instrument, this software may need to be exchanged or updated. Furthermore, if in disregard of instrument changing scientific requirements, the instrument was equipped with software that cannot be fixed and updated it would be required to completely qualify the software to almost highest ECSS criticality category. This causes additional effort and limits possibilities like dynamic memory allocation. Dynamic memory allocation in turn is essential for fast external interfaces using direct memory access (DMA).

II. Telecommand (TC) and Housekeeping (HK) Standards and Structure

The standard for data structures in TC, TM, and HK packets in ESA spacecrafts is ECSS-E-70-41A [3], which is based on guidelines agreed on in the Consultative Committee for Space Data Systems (CCSDS) such as reference document [4]. The standard ECSS-E-70-41A [3] is a packet utilization standard (PUS), defining the packet structure and a set of standard services. For TC packets arriving at a scientific instrument, the defined structure is depicted in Figure 1. Also this PUS [4] defines a set of standard services. These services are functions of the commanded entity e.g. the instruments DPM consisting of a command, an action, and if applicable a reply. The standard services are listed in Table I.

For a particular spacecraft the contractor building the spacecraft platform performs a tailoring of this standard and selects mandatory and optional services, the payload instruments must be able to perform.

Packet Header (48 Bits)							Packet Data Field (Variable)			
Packet ID				Packet Sequence Control		Packet Length	Data Field Header (most services)	Source Data	Spare (Optional)	Packet Error Control (Optional)
Version Number (=0)	Type (=0)	Data Field Header Flag	Applica-tion Process ID	Grouping Flags	Source Sequence Count					
3	1	1	11	2	14					
16				16		16	Variable	Variable	Variable	Not CCSDS, but ECSS

Figure 1 : CCSDS packet structure as refined by ECSS-E-70-41A [3]

Table I : Table 1 standard service defined by ECSS-E-70-41A [2]

Service Type	Service Name
1	Telecommand verification service
2	Device command distribution service
3	Housekeeping & diagnostic data reporting service
4	Parameter statistics reporting service
5	Event reporting service
6	Memory management service
7	Not used
8	Function management service
9	Time management service
10	Not used
11	On-board operations scheduling service
12	On-board monitoring service
13	Large data transfer service
14	Packet forwarding control service
15	On-board storage and retrieval service
16	Not used
17	Test service
18	On-board operations procedure service
19	Event-action service

Within that standard set of services, the service that can be used to update the instrument software is service 6 subtype 2 “Load Memory using Absolute Addresses service”. After an upload and storing of a new software version has finished, the DPM would simply have to be rebooted. There is no standard service for rebooting a payload instrument; one possibility is to use the service 8 subtype 1 “Perform function” or to use a set of private services, if they are allocated for the mission, to implement the reboot function. It is not sufficient if a boot loader supports only these two services, instead the boot loader must also implement a set of minimal standard PUS services, so that spacecraft requirements for nominal operation are fulfilled and the spacecraft allows further operation and does not power down the instrument, Table II lists an exemplary set of services.

Table II : Exemplary collection of a set of services

Minimal services that need to be supported			
Service 1: TC Verification Service			
TM	1	1	TC acceptance success report
TM	1	2	TC acceptance failure report
TM	1	7	TC execution success report
TM	1	8	TC execution failure report
Service 6: Memory Management Service			
TC	6	2	Load data into memory area using absolute address
TC	6	5	Dump memory area using absolute address
TM	6	6	Memory dump using absolute address Report
TC	6	9	Check memory area using absolute address
TM	6	10	Memory check using absolute address Report
Services for which the boot loader may need to generate a reply that avoids tripping error detection by the OBC			
Service 3: Housekeeping and Diagnostic Data Reporting Service			
TM	3	25	Housekeeping Parameter Report
Service 5: Event Reporting Service			
TM	5	1	Normal / Progress Report
TM	5	2	Error / Anomaly Report - Low Severity –Warning
TM	5	3	Error / Anomaly Report - Medium Severity - Ground Action
TM	5	4	Error / Anomaly Report - High Severity - On-board Action
Service 9: Time Management Service			
Service 17: Test Service			
TM	17	1	Connection Test Response
TM	17	2	Connection Test Response Report
Service 19: Event-Action Service			
TC	19	1	Add an Event to the Detection List
TC	19	4	Enable Actions
TC	19	5	Disable Actions

III. CCSDS / PUS SERVICES IN SPACEWIRE PACKETS

In SpaceWire packets a protocol identifier defines the packet type [6]. RMAP is assigned to the protocol identifier value 0x01 and the CCSDS packet transfer protocol is assigned to the protocol identifier value 0x02. [7] defines how packets of the CCSDS packet transfer protocol are transmitted through a SpaceWire network by appending addressing, protocol identifier, a reserved and user application byte at the start of the packet and an EOP marker at the end of the packet, see Figure 2.

	Target Spw Address	Target Spw Address	Target Spw Address
Target Logical Address	Protocol Identifier	Reserved = 0x00	User Application
CCSDS Packet (First Byte)	CCSDS Packet	CCSDS Packet	CCSDS Packet
Target Spw Address	...	...	CCSDS Packet
CCSDS Packet	CCSDS Packet (Last Byte)	EOP	

Figure 2 : SpaceWire packet transporting a CCSDS packet as defined by [6] ECSS-E-ST-50-53C

IV. BOOT MEMORY OPTIONS AND ARCHITECTURE

In order to avoid the effort of qualifying the entire instrument software to highest ECSS criticality level and allow for updates during space flight, current DPMs (such as for the

Polarimetric and Helioseismic Imager (PHI) instrument on Solar Orbiter) have a two stage boot process, as depicted in the system in Figure 3. The default boot memory address range (0x0000 0000-0x0FFF FFFC) of the employed LEON processor connects to a non-volatile memory containing a minimal boot loader plus an additional boot memory which is larger in storage and the content of which can be exchanged. In the case of Solar Orbiter PHI DPM a minimal PROM memory could be implemented within the Microsemi RTAX2000 system FPGA and a redundant NOR-flash is used to store the second boot image, which includes an RTEMS operating system and the complete instrument software. The basic boot loader will need to initialize processor registers and the SpaceWire interface and implement a basic driver for the SpaceWire interface. As the SpaceWire interface in processors such as the GR712 RC [1] uses direct memory access a substantial amount of software complexity and therefore boot loader size is required. Subsequently the boot loader needs to perform basic TC and TM handling and check if an update of the boot software needs to be performed. As NOR-flash cannot be written directly like a simple SRAM device, a driver performing defined program sequences also needs to be integrated in the boot loader.

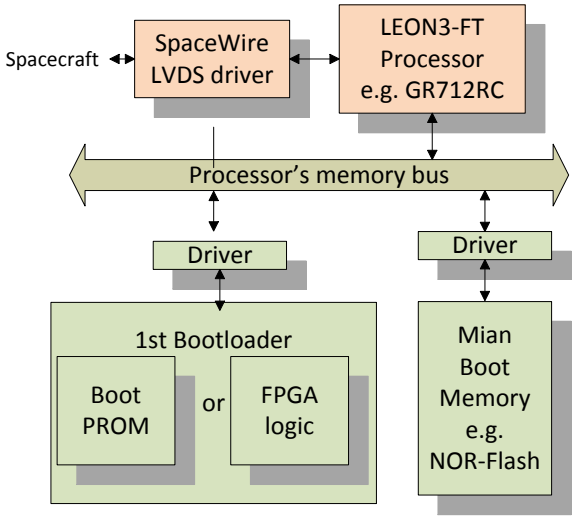

Figure 3 : Instrument data processing module boot memory set-up

V. Remote Memory Acces Protocol (RMAP)

The SpaceWire Remote Memory Access Protocol (RMAP) is a protocol that works over SpaceWire. This Protocol allows reading and writing memory remotely in a SpaceWire node. RMAP is defined in ECSS-E-ST-50-52C [5]. A memory write transaction is depicted in Figure 4 and it consists of SpaceWire addressing, protocol identifier, instruction, key, reply address, initiator logical, transaction identifier, address, data length, data and CRC-byte. In many radiation hard processors, such as e.g. the Aeroflex Gasiler GR712RC [1] the Aeroflex UT700 etc. the RMAP protocol is supported directly in hardware. In these devices even the complete address space from the processor bus can be read and written by RMAP. Therefore all the cores on the processor's AMBA bus including the debug support unit (DSU) can be reached. On ground the software debugger (GRMON) can connect to the processor through the SpaceWire interface without requiring an additional debug connector. Also on ground the second boot memory in the NOR-flash is written by uploading the boot image and a small program to the processor's working memory and then starting the small program that copies the boot image from working memory to the NOR-flash.

	Target Spw Address	Target Spw Address	Target Spw Address
Target Logical Address	Protocol Identifier	Instruction	Key
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Initiator Logical Addess	Transaction Identifier (MS)	Transaction Identifier (LS)	Extended Address
Address (MS)	Address	Address	Address (LS)
Data Length (MS)	Data Length	Data Length (LS)	Header CRC
Data	Data	Data	Data
Data	...	...	Data
Data	Data CRC	EOP	

Figure 4 : SpaceWire packet containing an RMAP write command (as given by ECSS-E-ST-50-52C [4])

VI. Using RMAP for Update of Instrument Software

As the boot memory update procedure on ground only uses the SpaceWire interface, which is also available via the space craft's OBC in flight, it seems logical to also use this procedure to update the boot software during flight operation. The only changes that would be necessary are, to create a separate RMAP command for the processor initialization that GRMON performs in the update process on ground and use RMAP's safety and error checking capabilities. However as RMAP provides access to all registers including debug support unit (DSU) this is not a problem. Thus a possible update procedure via RMAP for e.g. the GR712RC could consist of:

1) Stopping the software execution of the processor, by simply writing to the DSU register "break now"

2) Initializing processor register via DSU including program counter and Ancillary State Registers (ASRs)

3) Initializing the interrupt controller, memory configuration registers, and the GPIO controller

4) Disabling breakpoints and the debug mode by writing to the DSU control register and disabling the DSU Debug Mode Mask register

5) Uploading a program that writes boot memory content (see step 6) from working memory into the NOR-flash memory; alternatively this program could already be stored in another memory area and just copied to the working memory to minimize upload data volume, but still being replaceable by a newer version if needed

6) Uploading the new boot image via RMAP to the working memory. This could also be optimized by only uploading addresses and chunks of data where a difference to the current boot image occurs (patch)

7) Finally starting the copy to NOR-flash process by writing to the DSU break and single step register, when copying the data has finished the program can cause the processor to boot from the new boot image or cause a processor reset via a register in the supervisor FPGA.

As RMAP features a key byte and can be used with target logical address and RMAP cores are able to check these two bytes, they can be used as a security check to prevent any accidental triggering of the steps mentioned above, like e.g. stopping software execution.

VII. RMAP Access Problems

Despite appearing pretty straight forward an RMAP based software update procedure has some problems. Firstly in the many cases such as the GR712 processor the hardware RMAP support can be disabled by software, whereby a corrupted software image could potentially block any further accesses and disable software updates. Also software can set the SpaceWire logical address in the SpaceWire core of the GR712, which results in the SpaceWire core discarding any data that does not start with this logical address. Both of these problems could be somewhat alleviated by ensuring correct core settings through the first boot loader and halting the processor execution or including a wait before the second boot loader is started for a sufficient amount of time. As this is only a register access it will not increase the size of the first boot loader by a lot.

VIII. Conclusion

In the case of Solar Orbiter the ground operation team and particularly the OBC only offers CCSDS PUS services for payload instruments. Reasoning this with safety checks performed on the OBC and SSMM, which require a match between a SpaceWire packet's logical address and the application ID (APID) inside the CCSDS data packet, see Figure 1. However an RMAP write packet, which is depicted in Figure 4, starts with instruction, key and, reply address bytes and therefore cannot contain a CCSDS type header including APID which could be used for this safety check. Furthermore the Spacecraft would need to allow sending a packet with the protocol ID of RMAP (0x01). As the reply address in RMAP has 12 Bytes and logical addressing of a single byte is used it would be feasible to use the remaining 11 bytes for such header information. However this only applies to a reply, as in a request the position of an APID is the first byte of the reply address which is used for routing the reply packet.

Despite these difficulties an RMAP based update procedure would have several advantages. Such an update procedure would completely eliminate the need of any instrument software in the update process and thereby be inherently more reliable. Furthermore, if no software is required, this software does not need to be stored, which frees valuable FPGA resources or eliminates the need of an additional PROM, which would simplify system architecture and processor bus load as depicted in Figure 3. Additionally it would reduce development effort and costs, because no boot loader would need to be developed and qualified.

In conclusion RMAP would be an elegant, effective and more reliable mean of updating the instrument software, but there is a lack of harmonization between the two standard protocols of SpaceWire RMAP and the CCSDS PUS services and missing support by the OBC platform at least in the exemplary case of Solar Orbiter.

References

[1] GR712RC – User's Manual, Aeroflex Gaisler AB, http://www.gaisler.com, Issue 2.3, May 2014

[2] LEON4-N2X - Data Sheet and User's Manual, Aeroflex Gaisler AB, http://www.gaisler.com, Issue 2.3, May 2014

[3] CCSDS 102.0-B-5 Packet Telemetry, Blue Book, Issue 5, November 2000

[4] ECSS-E-70-41A "Ground systems and operations - Telemetry and telecommand packet utilization", European Cooperation for Space Standardization http://www.ecss.nl/, January 2003

[5] ECSS-E-ST-50-52C "SpaceWire - Remote memory access protocol", European Cooperation for Space Standardization http://www.ecss.nl/, February 2010

[6] ECSS-E-ST-50-51C "SpaceWire protocol identification", European Cooperation for Space Standardization http://www.ecss.nl/, February 2010

[7] ECSS-E-ST-50-53C "SpaceWire - CCSDS packet transfer protocol", European Cooperation for Space Standardization http://www.ecss.nl/, February 2010

Standardisation (Long)

Standardisation of the Network Management Service Suite (N-MaSS) for Fault Detection, Isolation and Recovery for SpaceWire

NOT PERMITTED TO PUBLISH PAPER

Standardized SpaceWire Solutions for Next Generation Systems

SpaceWire Standardization, Long Paper

Joseph R Marshall
BAE Systems
Manassas, Virginia, USA
joe.marshall@BAESystems.com

***Abstract*[1]—SpaceWire is leveraged as one of three main fabrics in SpaceVPX, a new system physical interconnect and form factor standard nearing completion at VITA. RapidIO and I2C along with heritage CompactPCI enable space systems to be built with full single point fault tolerance yet leverage the existing OpenVPX infrastructure for prototyping and test. SpaceVPX is described along with new network components under development that together may be applied across a large range of spaceborne electronics mission needs yet providing interoperability, scalability and future upgrade savings.**

***Index Terms*— Relevant indexing terms: SpaceWire, Networking, Spacecraft Electronics, SpaceVPX, OpenVPX, RapidIO, I2C, CompactPCI, NGSIS, Processor, Endpoint, Packet Switch, Crosspoint Switch, fault tolerance, DSP, RCC.**

I. Introduction

SpaceWire continues to see extensive usage throughout the space community. As a medium speed serial fabric, SpaceWire provides a low cost alternative to bussed systems and is easily scaled to meet performance requirements from less than 1 Mbps up to a couple of Gbps, after which a SERDES-based interface becomes appropriate. Supported by a large body of users, SpaceWire continues to evolve. This support and flexibility has caused SpaceWire to be written in as the control plane of VITA 78 (SpaceVPX) document, one of the Next Generation Spacecraft Interconnect Standards (NGSIS). The document specifies an interoperable form factor for space electronics boards within a backplane-based system needing multiple levels of fabric based interconnects for future space systems under AIAA, Serial RapidIO and VITA standards umbrellas.

This paper describes the history and development of the SpaceVPX standard and how the standard grew out of an industry consensus for a high performance internal form factor to support future high performance space payloads. The capabilities and extensions of the standard that are built on top of the extensive infrastructure developed for OpenVPX® – VITA 65 are highlighted. SpaceWire is leveraged in SpaceVPX as the control plane across defined slots and backplanes and as an intricate part of the fabric families supported (RapidIO, SpaceWire and I2C). The paper will describe how SpaceWire and SpaceVPX may be applied to space systems, both large and small. The application of BAE Systems' family of processing and network products with SpaceWire capabilities to SpaceVPX illustrates how systems may be built from a small set of network building blocks. Such building blocks may be utilized to create various interoperable SpaceVPX modules and networks, which range from remote endpoints to high performance processing payloads. The paper concludes with a view of future efforts in the NGSIS and SpaceVPX realms.

[1] Approved for Public Release #ES-MVA-072814-0388

II. SpaceVPX Development

A. NGSIS Formation and Goals

In 2011, a group was formed at the GOMACTech conference to look at the future in interconnect standards. Both government and industry people realized that they were on the verge of moving spaceborne onboard processing solutions from bussed based systems to fabric based systems. SpaceWire was the first successful foray into these new topologies though it did not offer enough scalability for upcoming mission data movement needs. With high speed SERDES elements supporting multiple commercial / protocol standards beginning to appear in advanced space technologies, there was a risk of multiple development options fracturing the space market. Rather than everyone invent their own way to do this, the group was formed to develop or adopt common standards. The Next Generation Spacecraft Interconnect Standards group was formed under the leadership of AFRL and JPL. [1]

The NGSIS group spent the first year capturing requirements and defining scope according to standard system engineering practices. With this in hand, the group determined the best interfaces to focus on. Four levels of interfaces were explored – a high speed SERDES fabric for data, a medium speed LVDS fabric for control and data, some form of lower speed interface such as 1553 for telemetry and a bussed interface such as PCI or VME. Comparing different trade studies for the best SERDES interface showed most organizations had selected Serial RapidIO as the most

promising because of its efficient data and error handling and power needs [2]. RapidIO also had sufficient commercial industry usage to provide an infrastructure to build on. SpaceWire was selected as the best choice for the medium speed control and data handling interface due to widespread usage, scalable performance, ease of implementation, existing standards and, though unique to spacecraft applications - sufficient existing test equipment infrastructure. 1553 and PCI were both identified as heritage interfaces to consider though 1553 was more oriented to box to box than internal boxes.

This led to two major focuses of the NGSIS effort. One group focused on creating a space version of RapidIO with extensions to that standard that are beneficial for using RapidIO in future spaceborne applications.[3] Originally named Part S, this has since been moved to the mainstream of the standard and will be incorporated in upcoming releases. The other group focused on a common form factor for NGSIS applications as otherwise there would be little interoperability. Once again, the consensus of the members was that OpenVPX or VITA 65 was the best match to the types of systems and modules that would be created in the future. This led to the formation of the SpaceVPX study group assigned to VITA 78 by the VITA Standards Organization.[4]

B. SpaceVPX History

The study group set a number of goals for their standard. SpaceVPX was to be as close to OpenVPX as reasonable to make sure users could leverage the OpenVPX infrastructure of modules, chasses and backplane for their systems. The standard would extend OpenVPX so that fully single point fault tolerant systems could be built by following the standard. RapidIO and SpaceWire would form the major interfaces in slot and backplane profiles defined to support interoperable modules that could support both mesh and star topologies. Heritage modules from previous systems would be accommodated.

One of the major early trades was how to extend the fault tolerance of modules beyond that of OpenVPX. Analysis showed that there were several single points of failure in the power and utility signal areas. SpaceVPX decided not to extend modules or place active circuitry on the backplane. Instead one new type of module, a Space Utility Management (SpaceUM) module was added. The SpaceUM module contains switching circuits and logic necessary to present one set of power and utility signals to each module selected from at least two sets of power and utility signal feeds from redundant elements. With proper fault containment, this provides single fault tolerance yet allows reuse of OpenVPX modules and backplanes for prototyping.

III. SpaceVPX Capabilities

SpaceVPX is mainly defined as overlapping profiles as shown in Fig 1. Chassis profiles define the structures that SpaceVPX backplanes and modules plug into. Power-keying profiles define the mix of voltages and currents that may feed the SpaceUM modules for selection to the logic modules. Backplanes describe the combinations of slot profiles that are used to build a system. Slot profiles defined the module pinouts and Module profiles define the protocols that are mapped onto the slots. The slot and module profiles provide the maximum opportunities for interoperability between vendors as flight backplanes will likely use SpaceVPX backplanes as guides – final flight backplanes will exactly match their needs to minimize size, weight and power.

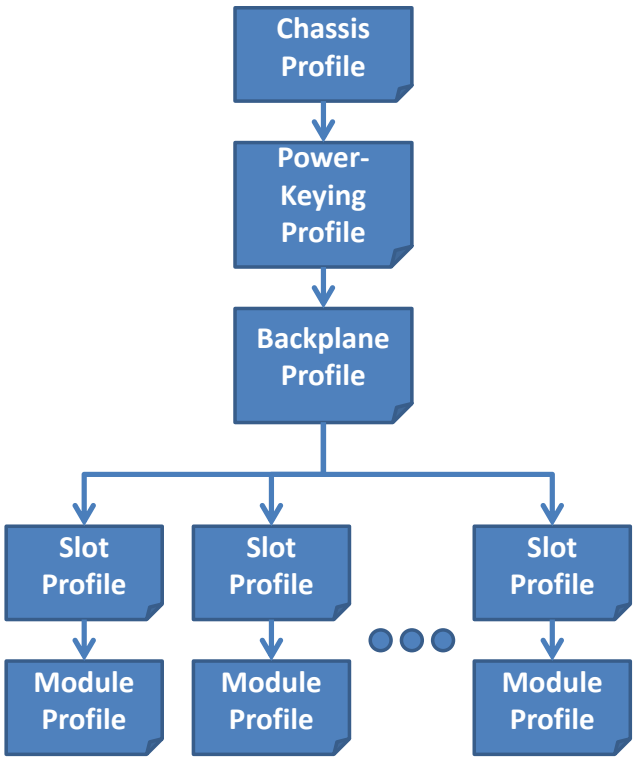

Figure 1 – SpaceVPX Profile Map

A. Interconnections and Planes

SpaceVPX, like OpenVPX, is all about connections between circuit card assemblies or modules. SpaceVPX provides a defined set of interconnection planes to span a large set of applications. These start with the System Management Interface (SMI) which is defined as I2C supplemented with a reset and a status line. This may operate up to 400 KHz. The SMI is designed to enable the system controller to handle chassis management, low speed commands, configuration, telemetry and status limited by the speed of the interface. The SMI is supplemented with a system reset and up to four broadcast clock or strobe signals to make up the signal portion of the utility plane in SpaceVPX.

SpaceWire is defined as the control plane for SpaceVPX. This provides a medium speed (up to 400 MHz) interface for both higher speed control, status, testing and telemetry as well as data rates that can make use of the speed of one or more SpaceWire links. SpaceVPX only defines the control plane in backplanes as a switch/router topology. However, SpaceWire switch/routers with embedded payload functions could be used in a mesh topology especially if no data plane was required.

The highest speed data needs are met by SpaceVPX's Data Plane. This is defined as RapidIO in configurations of one to four lanes each running up to 6.25 GHz per lane. SpaceVPX

defines both switch and mesh topologies to enable system integrators to build exactly what is needed.

SpaceVPX also defines an expansion plane that may be used to route high speed interfaces in a slot to slot fashion, forming rings or subnets independent of the main data plane. RapidIO is one of the interfaces that may use this additional plane, but the expansion plane may also be used for user defined interfaces between modules such as additional SpaceWire, XAUI, PCI Express or a unique protocol.

All these planes are connected in point to point networks, even the SMI. This is one the enablers for the advanced fault tolerance possible in SpaceVPX systems as compared to OpenVPX. Figure 2 shows a maximum slot profile with all the various planes and user defined pins identified.

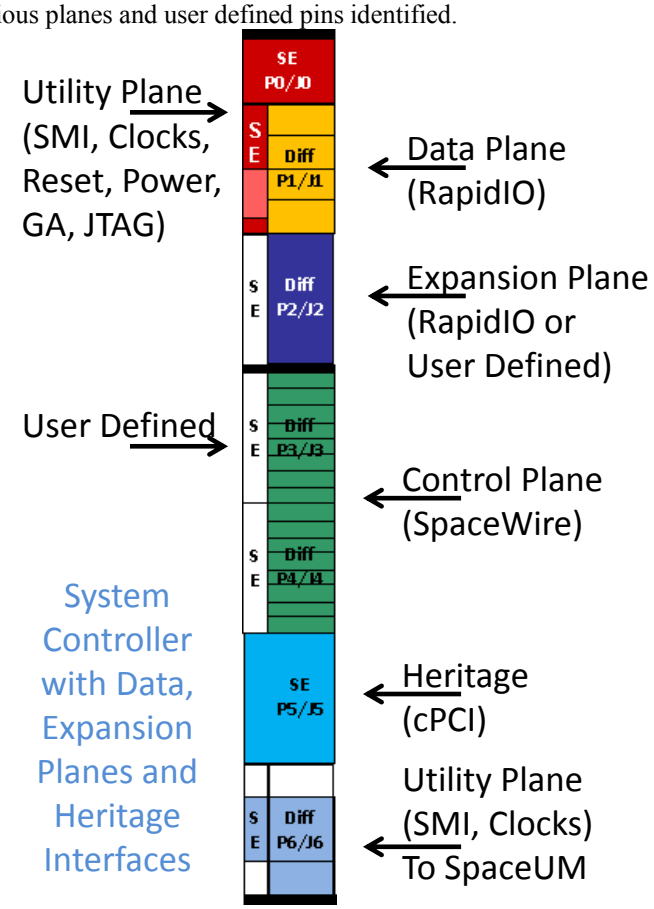

Figure 2 – Slot Profile Planes

B. Form Factors

SpaceVPX builds upon the metric form factors of OpenVPX, VITA 48.2[5], defining 3U and 6U form factors in lengths of 160, 220, 280 and 340 mm. Module pitch may be 0.8", 1.0" or 1.2". Larger slots may be created by ganging any of these. 6U Slot connector and pin definitions are identical for any length or pitch form factor. This is also true for 3U slot definitions within the standard.

The form factor uses the front edge of the top printed wiring board as the datum for all measures with standardized envelopes defined for each size. The 1.2" module pitch was added to OpenVPX's so larger components could be placed on both the front and the back of the modules. All SpaceVPX modules are assumed to be conduction cooled and wedgelocks are allowed on either side as long as the overall dimensions and connector placement are maintained. Module dimensions were chosen to allow OpenVPX modules to fit in SpaceVPX backplanes and chassis. Daughter cards are allowed within the envelope.

OpenVPX has defined three connectors which each may be used on OpenVPX modules. These connectors fit in the same space on a printed wiring board and thus are interchangeable to the design. The three require different backplane connectors and are thus not inter-matable. A system designer must pick a backplane connector for each slot and then that will determine which connector needs to be on the pluggable module. The working group could not differentiate sufficiently between the three connector types and thus also passed along this choice in SpaceVPX. The working group expects initial users will choose one connector and that may become the defacto standard for SpaceVPX modules. Or the ease of changing connectors on a module without other changes may encourage the use of two or all three.

Large 6U OpenVPX modules with multiple graphic engines can use up to 500 W per module. In the conduction cooled space environment, this is way beyond today's practical space cooling limits. SpaceVPX limits any 6U module to 100W and expects most initial modules to be under 50W. OpenVPX allows modules to take most of their power from 12V (>40A) and remaining amounts on 5V (>20A). 3U OpenVPX modules allow 20A on 12V, 5V and 3.3V. Since all power goes through the SpaceUM module with pin and isolation imposed limits on inputs and distribution, SpaceVPX defined several power profiles with varying amounts from either one to all three of these voltages. Keying is used to allow either 6U or 3U power schemes to be used on any module. This allows 6U heritage module designs that are 3.3V powered to be easily mapped to a SpaceVPX form factor without major redesign. Like the connector, power profiles are expected to consolidate around a few popular choices and others may be eliminated in a future version of the standard.

C. Fault Tolerance

The largest enhancement in SpaceVPX is the extension for fault tolerance. SpaceVPX defines the requirements to construct a single point fault tolerant system. This led to the following improvements:

At least two of every key element in the system is required. There are two power supplies, two system controllers and at least two of each payload, switch or peripheral module. For items like specific payloads or peripherals, M of N sparing may be employed to produce a more efficient sparing than multiple pairs of devices. All interfaces are cross-strapped between primary and redundant elements. Each redundant element must be designed for error containment to make sure an error is not able to propagate primary and redundant copies.

All utility and power signals in OpenVPX were analyzed. Any that have usage in SpaceVPX are duplicated by each system controller (e.g. CLOCKS, SMI) or contain a parity bit (e.g. the SYS_CON and SYS_CONP signals) to guard against single bit errors.

The newly defined SpaceUM module is a logical extension of the power supplies, the system controller and each logic slot in the system. It receives power from two power supplies and then uses selection logic to provide power to its logic and switches and to one or two of the system controller modules based on discretes received from external sources. The system controllers (or external commands to the controllers) direct the chassis powering, testing, status and operation of all other logic modules in a SpaceVPX system using the SMI. This includes isolating problems and powering up spare modules to replace failed ones. The SpaceUM module is the vessel for the power and signal switches from two power supplies and two system controllers to the single power feed and control signals that are radially routed to each logic module.

Each logic module contains a module manager that responds to the SMI and is able to provide basic module status and diagnostics. A minimum mandatory set is specified in SpaceVPX and additional access capability is defined for both generically specified elements and user defined extensions. In systems with control requirements that can operate within the bounds of the slow SMI, additional operational configuration, status and commands may also use the SMI from the System Controller.

The SMI managers must use and respond to either the complex publish –- subscribe protocol defined in VITA 46.11[6] or the simpler direct access protocol (DAP) defined in SpaceVPX. The latter was created to support simpler hardware implementations then possible with VITA 46.11. A simple state machine driven SMI is possible using the DAP that may be easily implemented as part of an ASIC or FPGA on the logic module.

Figure 3 shows the System Controller's central position of managing the internals of a SpaceVPX system. Note that compared to SMI, SpaceWire presents significant scalability and flexibility for increasing amounts of command and data handling throughout the system.

Taken together, the fault tolerant extensions introduced in SpaceVPX enable systems to be built that are single point fault tolerant. Of course for small applications, single string SpaceVPX systems may be built without the need for the duplicate elements, power and signal source switching or cross-strapping.

D. Heritage

In the past decade many space backplane systems relied on a PCI Bus using the CompactPCI® form factor and standard.[7] Thus, moving to the pure fabric approach in SpaceVPX could require all new modules. OpenVPX attempted to create a PCI backplane definition (VITA 46.2), but most military and commercial applications had already moved to fabrics on backplanes. Thus only PMC (IEEE 1386.1[8]) was defined as an on board daughter card standard for OpenVPX. SpaceVPX updated the OpenVPX work for the typical space usage (32-bit) of CompactPCI. The layouts and orientations of CompactPCI (cPCI) and VPX modules are opposite [9]. Thus in order to enable backplanes to route the cPCI bus from VPX modules to cPCI modules, the connector section P5/J5 was defined for the 32 bit mapping of PCI. This routed directly across to the P1/J1 connector on a cPCI module. Surveys of space users found limited interest in a 64 bit bus so only the 32 bit version was standardized. SpaceVPX did publish a suggested pinout for the additional pins on user defined pins in P4/J4, but the committee did not expect anyone to use that since the higher speed fabrics would be much more scalable than a 64 bit cPCI bus.

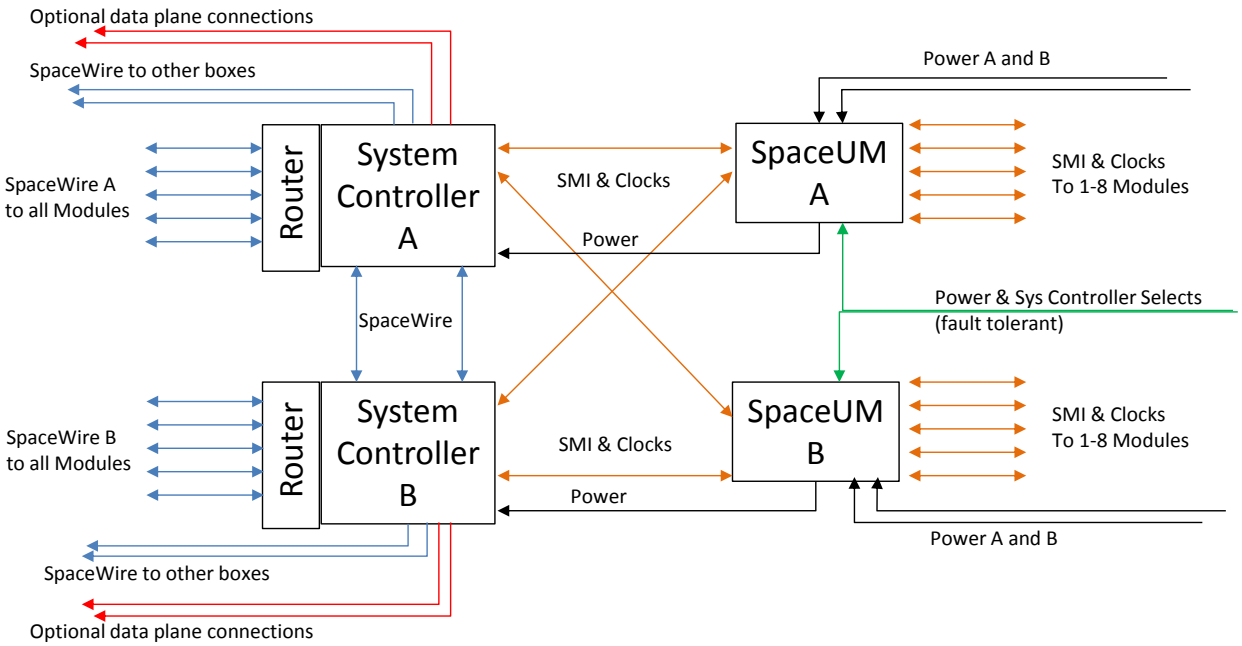

Figure 3 – System Controller Location in SpaceVPX

With the 32 bit PCI definition on P5, any payload, system controller or peripheral may serve as a PCI Bridge or participate on the PCI Bus. This will encourage heritage cPCI modules to first be inserted as a heritage module and then in subsequent systems be re-released in the SpaceVPX form factor with changes required only to add the utility and control plane signals. CompactPCI® is a bus and fully compliant modules do not support cold sparing. Thus any group of modules with a PCI Bus needs to be treated as a group for sparing. Redundancy requires that a separate set of modules have a second PCI Bus to be single point fault tolerant. The working group believes this will eventually phase out the use of cPCI in SpaceVPX; however the longevity of MIL-STD-1553 illustrates how long this may take in space systems.

E. OpenVPX Infrastructure

OpenVPX enjoys multiple suppliers and many applications. Modules, backplanes and chassis are available from multiple vendors that may be used for prototyping, debugging, stimulating and/or testing SpaceVPX modules and systems. The working group went to great lengths to make sure that SpaceVPX developers could leverage OpenVPX infrastructure [10] to reduce non-recurring costs in the development of systems. For example, the SYS_CON and SYS_CONP signals of SpaceVPX were carefully defined to work with the OpenVPX standard definitions and any known usage of these OpenVPX signal positions.

Mappings between SpaceVPX and OpenVPX profiles have been created and studied throughout the effort to develop SpaceVPX modules with the maximum cross-use between modules and backplanes between the two standards.

IV. SpaceWire in SpaceVPX

As described above, SpaceWire is the medium speed performance fabric in SpaceVPX systems. Using an extension such as RMAP, SpaceWire is well-suited for control plane operations as well as basic data handling and data streams. VPX connectors are rated commercially for up to 10 GHz and should support up to 6.25 GHz depending on backplane length. This should envelope the needs of the SpaceWire running up to 600 MHz on these connectors as well as the standard external 9 pin connectors off the top of a module.

A. System Controller

Refer back to Figure 3. Only slots with SYS_CON and SYS_CONP set to the proper states may be a system controller. The slot definitions are defined so that a single card design can function as a System Controller, Payload or Peripheral slot depending on what is active. A Payload or Peripheral function may use and route additional Control Plane ports for transferring data through the Control Plane to supplement the data plane connections.

The System Controller will depend on SpaceWire for connections to all modules after they are powered and active (via the Utility plane) without interfering with data transfers on the data plane. This can be used to configure modules by moving large blocks of code or tables of parameters or gather larger amounts of telemetry in real time than is possible on the 400 KHz SMI or to transfer medium amounts of data between the System Controller and specific modules. The control plane router/switch is defined on the System Controller in SpaceVPX but this can be split into two modules if needed as long as appropriate control plane cross-strapping between controller and control plane switch modules is employed.

B. Control Plane

The System Controller will have a control plane (SpaceWire link) to each SpaceVPX module in the box. Two links are provided between the SpaceVPX controllers and two external connections are included for extending the control functions between SpaceVPX boxes. Each System Controller slot has defined sufficient SpaceWire ports to meet the above in a typical system. User Defined signals may be used to add additional ports. Unused control plane ports are typically reserved on a module for maximum interoperability.

V. SpaceVPX Systems

SpaceVPX has been designed to apply across a large group of spacecraft applications. Anywhere a backplane is useful, SpaceVPX should be able to provide the interconnecting form factor for a spaceborne box, using the needed fabric subset for processing performance.

A. Large Payloads/Systems

The focus of the working group has been on boxes with 6U modules. These naturally may result in larger payload systems than the limited connectivity a 3U module can provide. Figure 4 shows a typical large system.

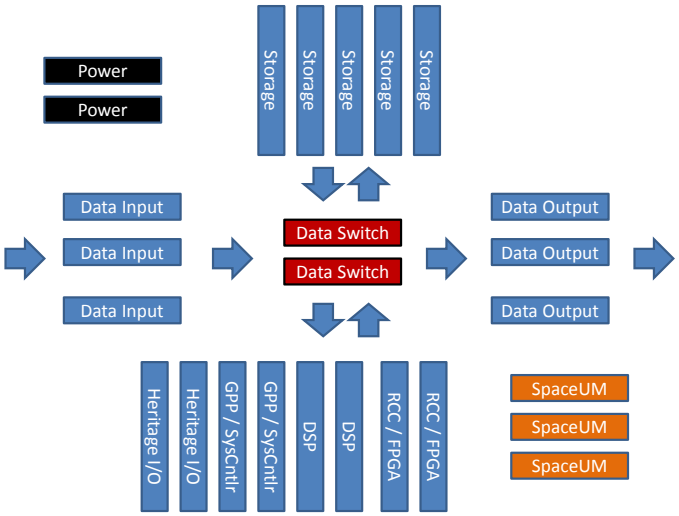

Figure 4 – Large System Module Block Diagram

Data (indicated by blue arrows) arrives from the analog world through input modules, may be stored in and retrieved from memory modules, processed on board in various types of processing modules and then exits the system on output modules. Each of these is typically a payload or a peripheral module. Each module type typically has at least one spare for fault tolerance as determined by reliability calculations. Two of the processing modules will be designated System

Controllers in a SpaceVPX system and one of these at a time is in control of the system as explained earlier. Data connections between modules are typically through a pair of redundant Data Plane Switches or a peer to peer mesh or some combination of the two. (Switches are shown). Dual power supplies and one to four SpaceUM modules (each supporting up to 8 logic modules) round out the makeup of a typical SpaceVPX system.

B. Small Payloads/Systems

Smaller payloads and systems typically combine some of the payload functions onto a smaller set of cards. For instance a reconfigurable computer (RCC) module may also have input RF or optical data functions or a storage module may also have a communications link to an external sync. Data Planes are typically greatly reduced which lowers the ports needed on switches or even eliminates data plane switches in favor of meshes. At the extreme, or to handle some of the input or output data, the SpaceWire links in the control plane may be used. Figure 5 shows a typical smaller system. The small system may be created out of either 6U or 3U modules depending on the function density on each module. Depending on data needs, either RapidIO or SpaceWire will be used for data movement.

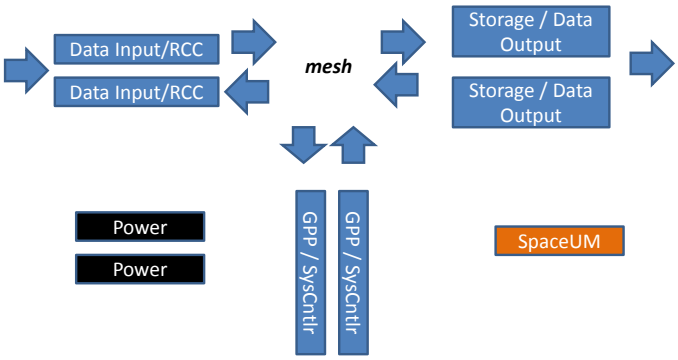

Figure 5 – Small System Module Block Diagram

VI. SpaceVPX Building Blocks

BAE Systems created one of the first radiation-hardened SpaceWire ASICs in 2004 [11]. This ASIC provided a bridge between two PCI busses and a SpaceWire router with four SpaceWire ports. This ASIC was used in multiple locations within the NASA Lunar Reconnaissance Orbiter (LRO) [12]. Based on this part, BAE Systems created a dual PHY single port SpaceWire Endpoint (now named the RADNETSPW-EP™) in 2011 [13] and the Golden Gate ASIC (now designated the RADNETSPW-BR4™) in 2012 [14]. This latter device combined into a single 150nm ASIC the original SpaceWire ASIC with the RAD750 bridge ASIC and the PCI Peripheral Interface ASIC containing 1553 and FIFO interfaces. Each of these two newer devices contain RMAP support in their connections from SpaceWire to the rest of each ASIC and an embedded microcontroller making remote load and remote access possible without initialization.

A. Latest Network ASICs

NGSIS standards were started because multiple organizations were on the cusp of moving to new fabric networks and wanted to standardize. BAE Systems is currently one of those and in the process of designing several new network ASICs.

The RAD55xx™ family of processors[15] feature from one to four 1.3 GOP CPU cores supported by four RapidIO ports of four 5 Gbaud lanes, sixteen SpaceWire links with a router, four I2C interfaces, two DDR3 ports and a host of other System on a chip (SOC) functions and interfaces.

The RADNETSRIO-EP™ bridges a dual PHY single port 5 Gbaud 4 lane RapidIO port to four SpaceWire links with a router, four I2C interfaces, a redundant MIL-STD-1553 interface, two dual PHY XAUI ports, two DDR3 memory ports and other SOC functions and interfaces. Five embedded microcontrollers supported by 256 KB of embedded memory are available to move the data between interfaces.

The RADNET1848-PS™ implements a 12 to 18 port RapidIO non-blocking crossbar switch across 48 lanes of 5 Gbaud RapidIO and significant network diagnostic registers.

The RADNET1616-XP™ provides a SERDES cross point switch that may be used for port sniffing, redundant ports or a small switch using its independent 16 input and 16 output SERDES lanes.

All of these ASICs are being implemented in RH45™, BAE Systems' 45 nm radiation hardened by design SOI technology.

B. Example Systems

Figures 6 and 7 show the systems in Fig 4 and Fig 5 annotated for where these components may be used to create SpaceVPX modules. This is representative of how new network ASICs or high performance FPGAs may be applied to creating SpaceVPX modules and systems.

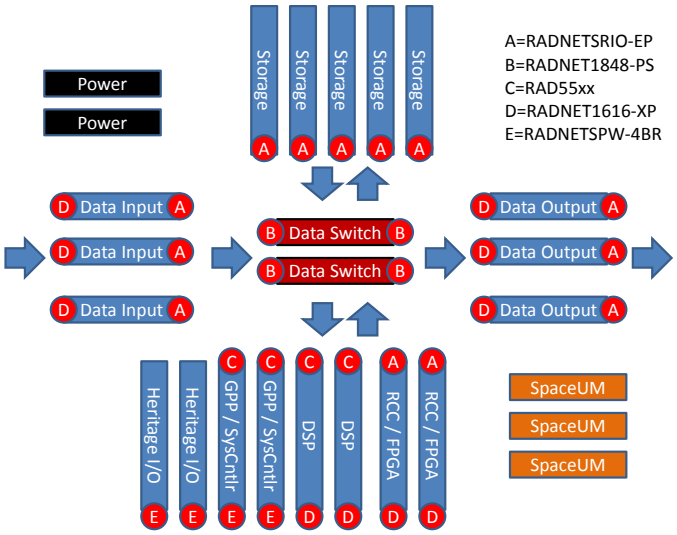

Figure 6 – Example Large Payload utilizing BAE Systems ASICs as building blocks

RapidIO endpoints (A) provide the SpaceVPX fabric connections to any of the typical payload functions – processing, input, output and storage. Packet switches (B) make up the data switch modules. RAD55xx (C) processors provide a scalable high performance processing module and system controller as well as SpaceWire control plane switch. Crosspoint switches (D) enable redundant ports to be added. Existing bridge ASICs (E) connect heritage cards to the SpaceVPX system.

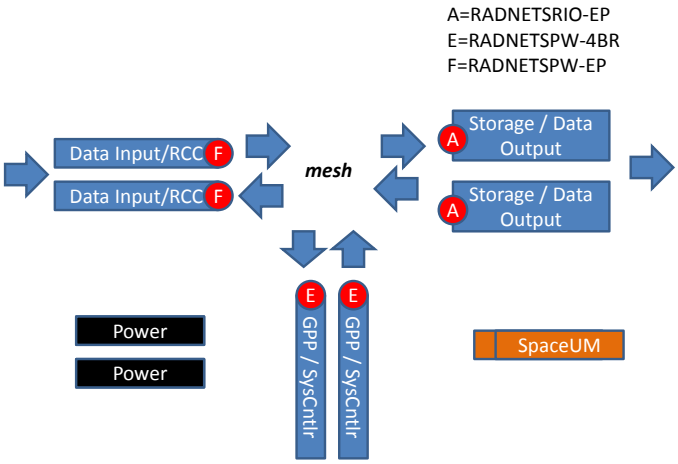

Figure 7 – Example Small Payload utilizing BAE Systems ASICs as building blocks.

The small payload is illustrated with SpaceWire interfaces being used for both control and data planes. Existing endpoints (F) and bridges (E) are used to provide the SpaceWire interfaces for the payload functions. If additional performance or interfaces are needed, the RapidIO Endpoint (A) can also be used.

VII. Future Direction

A. SpaceVPX

SpaceVPX or VITA 78 just completed a second working group ballot with only one no vote out of 17 and that was mainly to make sure the many broken links in that version were closed. The next step will be either a trial use standard or move toward full ANSI standard status. Several organizations are known to be developing modules that follow the current standard which will help validate many capabilities. Future focus of the group will be towards the 3U scenarios, power converters, optical in conjunction with VITA 79 and integrating some features back into VITA65 for OpenVPX users interested in more fault tolerant systems.

B. NGSIS

In 2014, NGSIS will be wrapping up initial SpaceVPX and RapidIO extensions with potential follow-ons. During the past year, there have been several other complementary standards efforts (e.g. SUMO, ESA)[16] that are attempting to define other areas of spacecraft to standardize. NGSIS members are active in defining the proper roles for NGSIS standards in conjunction with these efforts. The upcoming standards changes for SpaceWire will need to be evaluated for possible compatibility changes needed in NGSIS standards to maximize common solutions and interoperable networks.

C. Payoff

As SpaceVPX modules begin to appear in 2015 that utilize the new network ASICs being developed, scalable solutions will be possible across the three interconnect fabrics defined in SpaceVPX along with heritage interfaces such as PCI. Interoperability will be important both between different module types (switches and payloads) and the same modules (different payloads). As these products become available from multiple space vendors, best of breed systems can be constructed. Prototypes, test cards and generic backplanes and chassis may be purchased or adapted from OpenVPX modules, backplanes and chassis, cutting the NRE needed and leveraging the larger infrastructure of the OpenVPX marketplace.

The real payoff will come when systems that use the SpaceVPX modules are ready for an upgrade and modules that meet the SpaceVPX standards for existing modules or spare module slots may be inserted to create more powerful systems without having to go through a full interface definition effort again. This is attractive to both integrators who define the system needs and suppliers who will be able to apply their solutions developed for one system to other systems without the typical NRE required for insertion. BAE Systems is and will leverage its varied experience in spaceborne electronics standards along with its new and existing network ASICs to be part of this new paradigm for spacecraft systems.

References

[1] Charles Patrick Collier, Joseph Marshall, Richard Berger, Michael Enoch, Scott Goedeke, Next Generation Space Interconnect Standard (NGSIS): A Modular Open Standards Approach for High Performance Interconnects for Space, AIAA Reinventing Space 2013 Conference, Los Angeles, CA, September 2013.

[2] Marshall, J. R., Berger, R. W. and Bear, M. J., "Space Data Bus Technologies Evolve to Network Fabrics", GOMACTech 2012, Las Vegas, Nevada, March, 2012.

[3] RapidIO Trade Association, Austin, TX, www.rapidio.org.

[4] VITA Standards Organization, Fountain Hills, AZ, www.vita.com

[5] *Mechanical Specifications for Microcomputers Using REDI Conduction Cooling Applied to VITA VPX*, ANSI/VITA 48.2, VITA Standards Organization, Fountain Hills, AZ, July 2010.

[6] *System Management on VPX*, VITA 46.11 Trial Use Version, VITA Standards Organization, Fountain Hills, AZ, September, 2013.

[7] Marshall, J. R., Stanley, D and Robertson, J. E., "Matching Processor Performance to Mission Application Needs", 2011, Infotech@Aerospace Conference, St. Louis, MO, 2011.

[8] *IEEE Standard Physical and Environmental Layers for PCI Mezzanine Cards (PMC)*, IEEE 1386.1, IEEE Computer Society, New York, NY, June 2001.

[9] CompactPCI Specification, PICMG 2.0 Release 3.0, PCI Industrial Manufacturing Group, Wakefield, MA, October, 1999.

[10] VPX Marketing Alliance, www.vita.com/vpx.

[11] Marshall, J. R., Berger, R. W. and Rakow, G. P., "A One-Chip Hardened Solution for High Speed SpaceWire System Implementations", 1st International SpaceWire Conference, Dundee, Scotland, September,

[12] Berger, R. W., et. al., "RAD750 SpaceWire-Enabled Flight Computer for Lunar Reconnaissance Orbiter", Proceedings of 1st International SpaceWire Conference, Dundee, Scotland, September, 2007.

[13] Marshall, J., et. al., "Leveraging SpaceWire Network Prototyping to Create Flexible SpaceWire Components and Support Software", Proceedings of the 4th International SpaceWire Conference, San Antonio, TX, November*, 2011.*

[14] Marshall, J. R., "Evolution and Applications of System on a Chip SpaceWire Components for Spaceborne Missions", 2nd International SpaceWire Conference, Nara, Japan, 2008.

[15] Richard Berger, Richard Ferguson, Jane Gilliam, Michael Graziano, Mary Hanley, Marla Lassa, Joe Marshall, Hugh Miller, Dave Moser, Dale Rickard, Dan Stanley, Joe Stevenson, Standards Development of a radiation-hardened system-on-chip multicore Power Architecture processor , GOMACTech 2014, Charleston, SC, April 2014.

Test & Verification (Long)

iSAFT-PVS: Recording, Simulation & Traffic Generation at Full Network Load

Test & Verification, Long Paper

Antonis Tavoularis, Vassilis Vlagkoulis, Nikos Pogkas and Vangelis Kollias
TELETEL S.A.
Athens, Greece
A.Tavoularis@teletel.eu, V.Vlagkoulis@teletel.eu, N.Pogkas@teletel.eu, V.Kollias@teletel.eu

Kostas Marinis
On-board Computers and Data Handling Section
European Space Agency/ESTEC
Noordwijk, The Netherlands
Kostas.Marinis@esa.int

***Abstract* — Over the past years several test tools have been developed for verification and validation activities for on-board components and networks. Up until now there is no tool that can be used for all on-board networks and EGSE providers have to combine from different COTS providers and custom developments for the fulfillment of the testing requirements.**

TELETEL under ESA's and ASTRIUM Toulouse's consultancy developed the iSAFT-PVS which is an integrated powerful HW/SW environment for the simulation, validation & monitoring of satellite/spacecraft on-board data networks supporting simultaneously a wide range of protocols (RMAP, CPTP, TM/TC, CANopen, etc.) and network interfaces (SpaceWire, ECSS-1553, ECSS-CAN) offering the reliability features required for space test benches and IRIG throughout all interfaces for common and accurate time-stamping.

The paper presents the SpaceWire instances of PVS, Recording and Simulation, its reliability features and performances. The paper presents an overview of the iSAFT system, the iSAFT Recorder which provides the user with the capability to record traffic on multiple networks/links, set triggers and filters etc., and the iSAFT Simulator, with Traffic Generation capabilities, which allows triggered transmission and programmable link saturation under local or CCS remote control. The performances of the two instances, which are presented herein, have been taken during long-run stress tests (full link traffic over several SpaceWire links) and reveal that iSAFT can be used at full link utilization for either Recording or Simulation. From the measurements it is evident that iSAFT can not only be used for Recording and Simulation of C&C flaws which present infrequent data bursts, but also for payload flaws with rates which are significantly higher and also for scenarios in which time-accurate transmissions are required in order to accurately simulate the instruments' behaviours.

***Index Terms*— Relevant indexing terms: SpaceWire, 1553, CAN, IRIG, FMECA, Recorder, Simulation, Traffic Generation**

I. Introduction

iSAFT is an integrated powerful HW/SW environment for the simulation, validation & monitoring of satellite/spacecraft on-board data networks supporting simultaneously a wide range of protocols (RMAP, CPTP, ECSS-1553, CANopen, etc.) and network interfaces (SpaceWire, MIL-STD-1553, CAN). It is based on over 20 years of TELETEL's experience in the area of protocol validation in the telecommunications and aeronautical sectors, and it has been fully re-engineered in cooperation of TELETEL with ESA & ASTRIUM Toulouse, to comply with space on-board validation requirements (ECSS, EGSE, AIT, AIV, etc.). iSAFT has already been used in several ESA studies to validate devices (e.g. SCoC3) or prototype, validate and assess new developments (ECSS-1553, SpW-T, SpW-D, SpW Interrupts Distribution, N-MaSS). iSAFT is highly modular and expandable to support new network interfaces & protocols and it is based on the powerful iSAFT graphical tool chain (Protocol Analyser / Recorder, TestRunner, Device Simulator, Traffic Generator, etc.).

iSAFT can be used for the validation of units used in specific scientific missions which generate large volumes of data, like the GAIA Video Processing Unit, and for which validation can become very demanding. For such missions the requirements for both recording and the simulation may exceed the performances of many systems and it may be required to parallelize test equipment thus creating complex EGSE architectures and generating SW synchronization issues. This paper presents the functional and performance characteristics of two instances of the iSAFT system, the iSAFT Recorder and iSAFT Simulator including its Traffic Generation engine. The main objective of the work presented in this paper was carried out in the frame of ESTEC Contract no. 4000105444/12/NL/CBI [titled "Protocol Validation System (PVS) activity"] and the results prove that, for both recording and simulation, iSAFT can be trusted even in missions with very high performance requirements.

II. iSAFT Overview

iSAFT is an advanced, integrated, high performing, modern platform for the simulation, validation & monitoring of a wide range of satellite/spacecraft on-board communication protocols and data networks. Its plethora of features makes it suitable for use in many different areas such as:

- Rapid Prototyping/Evaluation: Implementation of new protocols, experimentation with various protocol features (parameterization of protocol variables, exclusion/inclusion of protocol optional functions, combination of multiple protocols, etc.)
- Device Simulation: Economic & portable replacement of a device in the testbed (SSMM, RTU, RMAP responder, etc.)
- Functional/Conformance Testing: Execution of nominal tests to ensure that a device (System Under Test) is operating in compliance with the applicable ECSS standards. Error injection at various protocol layers to validate the response of the devices/networks in erroneous conditions
- Traffic Generation: Periodic and Bulk traffic injection at higher & lower protocol layers for performance evaluation and network dimensioning
- Protocol Analysis/Recording: Message decoding & recording, filters, start-stop triggers, intelligent error detection, export of results, real-time statistics

Fig. 1. The iSAFT Server based platform (2U)

iSAFT comes in different configurations which cover different performance/reliability/cost requirements. An example configuration is the one shown in Fig. 1. , which is a 2U rack mount system with Xeon E5-2403 processors. The system can be configured with single or dual processors, SSD of at least 256 GBytes expandable to 2 TB, archive disk of at least 2 TB, 4 GbE ports for connection to EGSE CCS, SCOE LANs and 6 PCIe slots in which the network interfaces are installed. Several configurations are possible, customized to the requirements of the system under test i.e. it can be configured with SpaceWire, 1553, CAN interfaces (shown in Fig. 2.) or pure SpaceWire with support for up to 20 SpaceWire ports etc. The system integrates CCSDS remote interface for remote commanding by the Central Checkout System (CCS), whereas support of the EDEN protocol is under development.

Each of the SpaceWire boards has eight SpW ports, each of the 1553 boards support up to four 1553 channels and each of the CAN boards supports up to four CAN channels. Internal failures are blocked and are not propagated to the flight equipment, thus constituting iSAFT safe for connection to flight equipment. The SpaceWire and 1553 have already passed through FMEA analysis and installed in primes testbeds, whereas for the CAN interface FMEA is under progress.

Common time-stamping across all interfaces/boards is supported through the iSAFT IRIG-B port. iSAFT uses a single IRIG connector for connection to external IRIG sources and propagates the IRIG signal in all installed cards internally with nanoseconds delay/skew. In addition, it can also be configured as an IRIG source for cases in which an external source is not available. When an external IRIG source is used, the system regenerates the IRIG stream with a few nanoseconds delay and provides it to the IRIG connector allowing cascaded iSAFT systems, or third-party external equipment to be connected to the IRIG chain in daisy-chain thus eliminating the need for IRIG splitters/distributors.

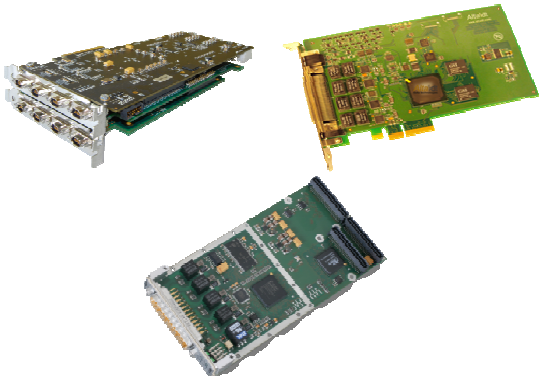

Fig. 2. The iSAFT SpaceWire, 1553 and CAN interfaces

The SpaceWire board supports up to eight SpaceWire ports, with independently programmable Link Speeds up to 400 Mbps (200 Mbps for recording in the current version) with a resolution of 30 Kbps and has Monitoring, Simulation, Traffic Generation and error injection capabilities. An option for RMAP Target Simulation is available, with programmable memory map and programmable response time down to less than one microsecond, used for emulation of devices implementing RMAP in Hardware. The board integrates an IRIG generator and an IRIG receiver with resolution and accuracy down to 8 nanoseconds.

The 1553 board supports up to four 1553 channels, with transformer bus coupling, and offers Monitoring, Simulation, Traffic generation and error injection capabilities. Different versions of the board exist supporting dual functionality (BC or RT with simultaneous BM functionality), full functionality (BC, RT and BM simultaneously), options for variable bus voltage and an extension supporting SAE tests is also available. The board integrates an IRIG receiver with down to 20 nanoseconds accuracy.

The CAN board supports up to four electrically isolated CAN channels with independently programmable baud rates from 10 Kbps to 1 Mbps and has Monitoring, Simulation and Traffic Generation capabilities. The board has a hardware scheduled transmission queue used for transmission of messages requiring minimized jitter (e.g. ECSS-CAN SYNC object), supports auto-queued answers in hardware and supports error injection at CAN level. It integrates an IRIG receiver with down to 63 nanoseconds accuracy.

A. iSAFT Recorder

iSAFT Protocol Analyser/Recorder is based on the iSAFT graphical tool chain (Runtime engine, iSAFT Console, offline analysis with the Wireshark Protocol Analyzer, recordings management). It captures and records large volumes of traffic

from multiple SpaceWire links, MIL-STD-1553 and/or CAN buses and offers off-line analysis of multi-gigabyte traffic logs. iSAFT supports chronological merging of recorded traffic (e.g. from both SpaceWire and 1553), event-trace trigger & selective tracing (filtering) support and offers plug-ins and real-time statistics for various protocols. It integrates a set of graphical tools for local/remote control, data recording, managing, searching and filtering the recordings and also interfaces with EGSE Central Checkout Systems. Export of traffic recordings to XML, PostScript®, CSV, or plain text and user selected protocol fields per packet are supported. In addition, an open APIs is available for 3rd-party applications to support customizations/adaptations to user needs.

For the case of SpaceWire the iSAFT recorder supports either packet level or character level monitoring per port allowing troubleshooting at system or at protocol level.

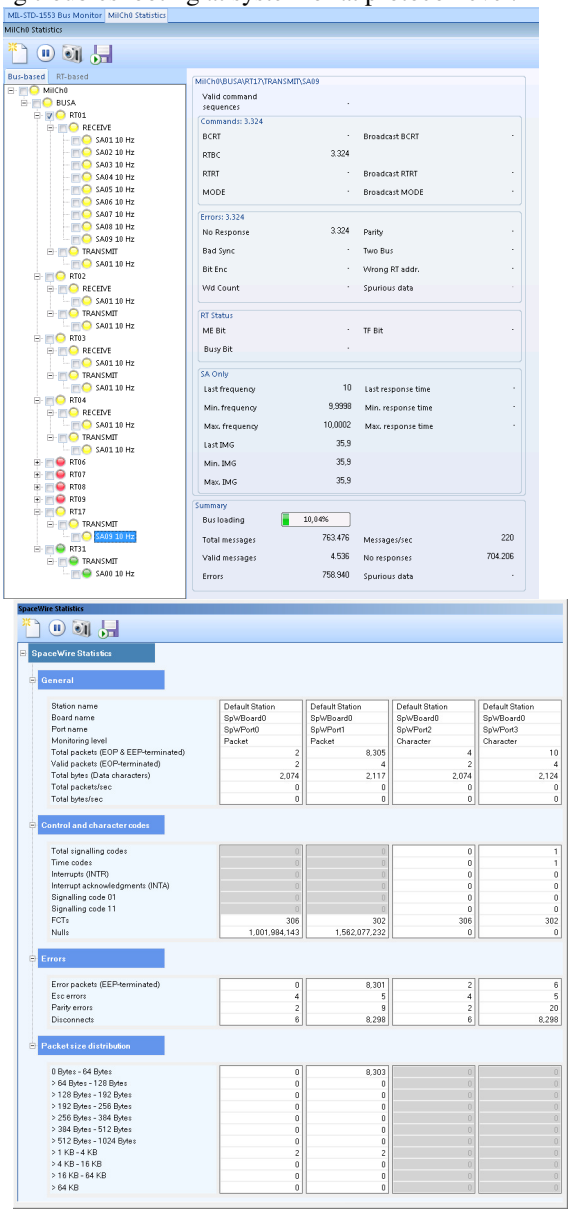

Fig. 3. iSAFT recorder MILBUS and SpaceWire statistics views

In order to support advanced recording functionalities, recording for each interface can either begin on the press of the start button by the user or, per port independent, start and stop trigger conditions can be set-up. Specifically, the following start/stop trigger conditions are supported:

- SpaceWire:
  - Packet Level monitoring: absolute/relative IRIG time, any/programmable Signaling Codes (Time-Codes, Interrupt Requests, Interrupt Acknowledgments, etc.), programmable NCHAR, programmable packet pattern, parity error
  - Character Level Monitoring: absolute/relative IRIG time, NULL, FCT, any/programmable NCHAR, EoP/EEP, any/programmable Signaling Code, parity error
- MIL-BUS: specific word (with mask), parity, bit encoding, SYNC, word count, gap between data words, no response, wrong status, spurious data errors and external trigger
- CAN: Date/Remote/Error frames, COB-ID (with mask)

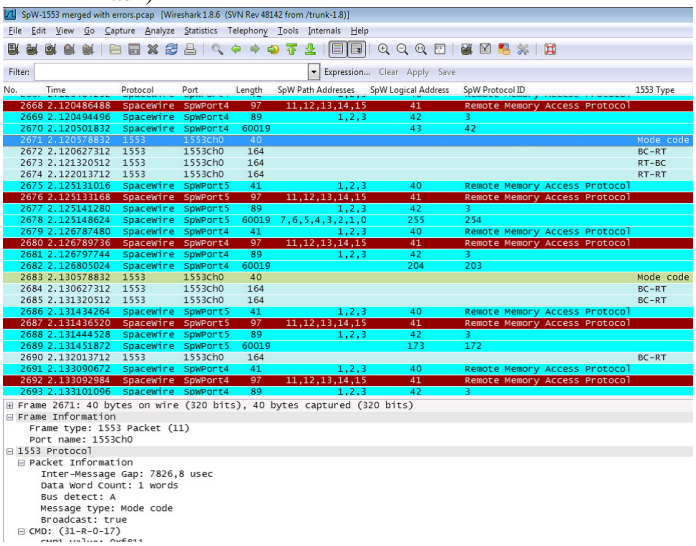

Fig. 4. The iSAFT protocol analyzer GUI

Similarly, the system allows filtering of traffic to be captured in order to decrease the amount of captured data and extend the recording time. The following filters are supported:

- SpaceWire:
  - Packet Level monitoring: traffic between Signaling-Codes, programmable packet pattern (normal or inverted) with the capability to ignore path address bytes in order to support traffic capture on links between switches, valid/error packets, programmable logical address/Protocol ID (extended PID also supported)
  - Character Level Monitoring: Traffic between Signaling Codes, NULL, FCT, NCHAR, Signaling Codes, valid/error characters
- MIL-BUS: Remote terminals, Sub-addresses

- CAN: Date/Remote/Error frames, COB-ID (with mask)

Captured traffic from multiple interfaces can be merged and displayed in a single protocol analyzer chronologically ordered, as shown in Figure 3, allowing common view of multiple interfaces thus enabling analysis of device performances. Standard or custom protocol decoders can be installed in the system allowing decoding of protocol fields such as RMAP, CCSDS, ECSS-1553, ECSS CAN etc.

B. iSAFT Simulator

iSAFT provides the ability for prototyping on-board data network devices allowing simulation of a network element thus enabling S/C integration tests before the availability of Flight models.

The iSAFT simulator allows for rapid prototyping of new functionalities allowing for experimentation with various device features and variations including parameterization of variables, exclusion/inclusion of device optional functions, combination of multiple protocols, etc. It enables simulation of specific satellite/spacecraft platform interfaces (as power, command, telemetry and communication) to different Payload Instruments. It provides a local interface as well as an open API for easy integration with 3rd party simulation software and a TCP/IP based remote control interface for integration to LAN-based environments.

Regarding SpaceWire, transmissions can either be asynchronous or on user-programmable trigger conditions per packet. The supported trigger conditions in the current version are full or partial IRIG time, programmable Signaling Code with programmable offset, simultaneous over selectable ports, external trigger signal with programmable offset (e.g. PPS), programmable delay from previous packet and disconnect on another port. Exploitation of these capabilities allows the user to reproduce previously captured traffic in time-accurate fashion with sub-microsecond accuracy, thus reproducing scenarios that lead to the appearance of failures.

In addition, iSAFT simulator provides a Traffic Generation engine supporting multiple periodic channels or bulk traffic injection for link saturation allowing performance evaluation at device or network levels.

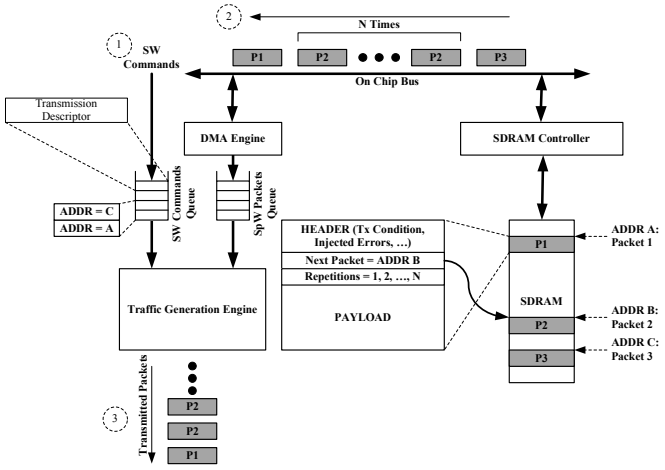

Fig. 5. The iSAFT Traffic Generation Engine operation

The bulk Traffic Generation engine, is based on transmission of linked lists of packets or packet sequences, allows the creation of single or repetitive sequences as shown in Fig. 5. The SW downloads the packets to be transmitted in the on-board SDRAM, having already programmed the packet headers accordingly to point to the next packet in the sequence and the number of repetitions for each packet/packet sequence. In the example shown in Fig. 5. the SW downloads a transmission descriptor used to fetch the first packet of the linked list sequence in the transmission queue (step 1), which in this case is Packet P1. As soon as the packet is fetched, the pointer to the next packet is examined (if any) and the next packet is automatically fetched for transmission, which in this example is packet P2 (step 2), at the same time at which Packet P1 is being transmitted (step 3). Packet P2 shall be transmitted N times, so it is fetched from the memory N times in the transmission queue. If no other packets exist in the link list transmission stops if there is no transmission descriptor in the SW commands queue. Each packet descriptor has additional control information such as the transmission trigger condition, error to be injected etc. For example, the user can select to start the transmission of a packet sequence upon the detection of a PPS and configure the subsequent packets to be transmitted on specific time-codes or have specific and different time-gaps among them. The combination of per packet independent triggers with Traffic Generation provides time-accurate device simulation capability.

III. iSAFT PERFORMANCES

All results presented herein were performed on a iSAFT 2U platform (shown in Fig. 1.) with a single Xeon E5-2403 processor, 256 GB SSD and 2TB archive disk. As the 1553 and CAN interfaces are low speed interfaces, their impact on the performance of the system is minimal and therefore it is only the SpaceWire performance which determines the overall system performance for both monitoring and simulation. To this respect in order to discover the performance limits of the system, tests with the SpaceWire interface were performed. In all the tests a single eight ports SpaceWire board was used and thus the system was capable of capturing traffic on four SpaceWire links or simulating up to eight SpaceWire devices.

A. iSAFT Recorder Performances

The first test was performed on the iSAFT Recorder. An external SpaceWire Traffic Generator was configured in traffic generation mode, continuously transmitting SpaceWire packets at 100 and 200 Mbps, without NULLs between the packets. As each captured packet is appended control information (packet length, start/end IRIG time-stamps etc.), for presentation to the user through the WireShark analyser, this constitutes the packet overhead which becomes more significant as the packet size decreases. It is therefore expected that with decreasing packet size, the required throughput to store the captured packets along with the control information on the platform memory will exceed systems performance. At this point the capture memory becomes full and recording stops in order not to overwrite already captured traffic. This is the rationale of the

tests presented here in, in which the packets starting from 1 Kbytes were continuously decreased until a "Buffer Full" condition appeared on the iSAFT GUI. The test results are shown in Fig. 6. from which we see a minimum packet size of 78 Bytes at 100 Mbps and 184 for 200 Mbps. This corresponds to 971463 and 830737 packets per second at 100 and 200 Mbps respectively.

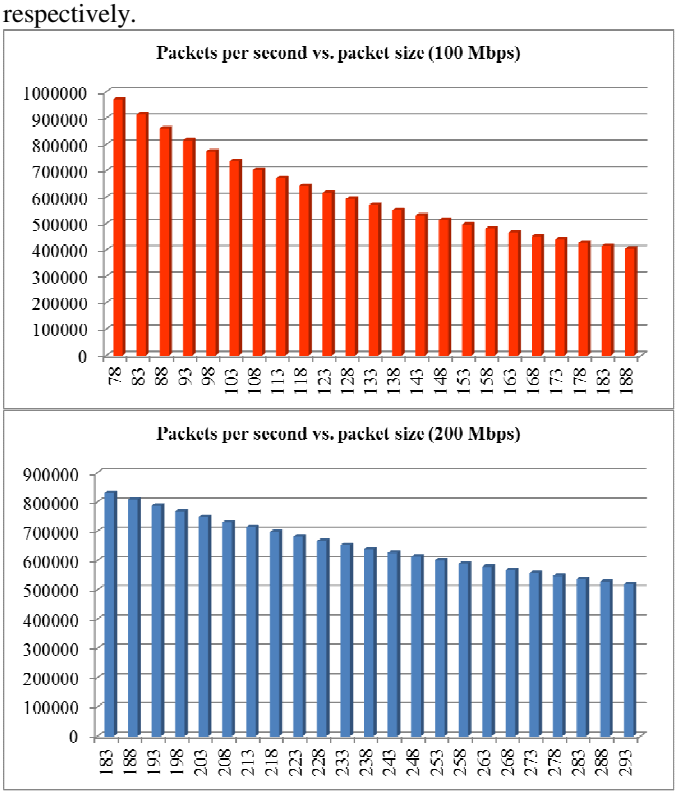

Fig. 6. iSAFT recorder performance. Packets per second vs. packet size on 8 SpW ports

The slight difference in packets per second between the two measurements can be explained by taking into account the payload of each measurement. Although the overhead for the iSAFT SW is same for both link speeds, since only the packet headers are processed by the SW, the performance difference can be explained by the fact that at 200 Mbps the throughput required to transfer the packet payload doubles, thus doubling the system bandwidth requirements.

The system limits, correspond to non-realistic scenarios, since small packets are used for C&C and are never transmitted in bulk mode. Bulk data corresponding to mission payload (e.g. images) are transmitted in large SpaceWire packets for which the overall iSAFT performance is more than adequate and therefore the reader can safely assume that iSAFT recorder is capable of capturing any mixture of realistic traffic over eight SpaceWire ports.

Ongoing tests that are being performed on a 16-ports Recorder have shown that up to 15 fully utilized ports can be captured at 200 Mbps with the current iSAFT version, resulting in an overall data throughput of more than 2,2 Gbps, whereas for 100 Mbps link speeds traffic from 16 fully utilized ports can be captured.

B. iSAFT Simulator Performance

The second set of tests was performed on the iSAFT Simulator. The purpose of the tests was to assess the performance of asynchronous transmission and reception. Specifically, the following measurements were made:

- Transmission Latency: The time from the point the transmitting application on iSAFT calls the transmit function to the time, the first packet NCHAR appears on the SpW link
- Reception Latency: The time from the point the entire packet reaches the destination port to the point this is available at the receiving user buffer

Long run tests were performed with test applications transmitting millions of packets in order to assess system stability. The scenario involved synchronization of all equipment through IRIG in order to provide common time-stamping and ensure the measurements accuracy. An IRIG source was connected to the PCs hosting the transmitting and receiving applications.

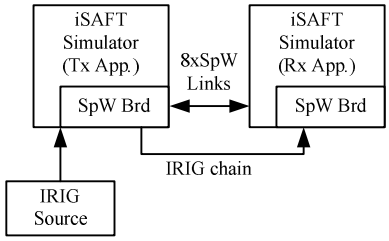

Fig. 7. Remote control measurements test set-up

The transmitting application was retrieving the IRIG time right before packet transmission. Upon arrival of the first packet byte at the receiving system the packet was assigned a "start time-stamp" by the HW. The difference between the two time-stamps constituted the Tx transmission latency.

Similarly, in order to measure the Rx latency the receiving application was invoking the receive call and upon the call return it was reading the IRIG time from its local IRIG receiver. Subtracting the packet's "end time-stamp" from this time corresponded to the Rx latency.

TABLE I. THE ISAFT SIMULATOR LOCAL OPERATION PERFORMANCE

Packet Size	Tx Latency (us)	Rx Latency (us)
1K	10.73	9.15
2K	12.79	9.15
4K	14.77	9.27
64K	73.65	9.88

The results shown in TABLE I. show that the latencies are reasonable with a transmission latency of less than 100 microseconds for 64Kbytes packet. For the receive path, the latency is minimal and independent of the packet's size due to iSAFT's architecture.

C. iSAFT Traffic Generation Performances

The last test run was performed in order to assess the performance of the Traffic Generation engine of the iSAFT Simulator. Two different test-sets were performed:

- In the first one, the same packet was being transmitted continuously in repetitive mode
- In the second, the traffic generation engine of the iSAFT Simulator was configured to transmit linked-list packet sequences

In both test-sets the receiver was configured in two different modes:

- HW sinking, in which packets are not uploaded to the platform memory but are immediately discarded by the HW
- Normal operation in which packets are uploaded to the system memory through a simple test application.

The first mode reveals the performance of the Traffic Generation engine, whereas the second mode reveals the overall iSAFT Simulator performance under stress. Stand-alone tests of the receiver are to follow.

All tests were performed at three different link speeds, 100, 200 and 300 Mbps and the purpose was to find the engine's maximum performance by decreasing the packet size down to the point at which NULLs are inserted in the link. The links were captured by the iSAFT Recorder in order to observe the presence of NULLs through the Recorder's real time statistics.

The results of all tests are shown in TABLE II. Fig. 8. shows only the results of the HW sinking mode tests of the receiver, i.e. the performance of the traffic generation engine.

TABLE II. THE ISAFT SIMULATOR TRAFFIC GENERATION PERFORMANCE

	SpW ports	**Minimum Packet length (Single packet in each sequence)**			**Minimum Packet length (Two-Packets sequence)**		
		100 Mbps	***200 Mbps***	***300 Mbps***	***100 Mbps***	***200 Mbps***	***300 Mbps***
Hardware Sinking of received packets	1	8 bytes	16 bytes	25 bytes	4 bytes	9 bytes	15 bytes
	2	16 bytes	31 bytes	48 bytes	8 bytes	17 bytes	26 bytes
	4	31 bytes	66 bytes	104 bytes	17 bytes	35 bytes	56 bytes
	8	66 bytes	140 bytes	256 bytes	35 bytes	82 bytes	149 bytes
Normal packet reception	1	30 bytes	60 bytes	94 bytes	26 bytes	52 bytes	83 bytes
	2	57 bytes	121 bytes	197 bytes	50 bytes	113 bytes	173 bytes
	4	121 bytes	290 bytes	1184 bytes	107 bytes	287 bytes	913 bytes
	8	289 bytes	-	-	287 bytes	-	-

Three observations can be made on the graph of Fig. 8. The first observation is that for repetitive transmissions of a single packet the performance is significantly lower than the respective performance of a two packets sequence. The second observation is that for a given SpW link speed, the packets per second does not have strong dependence on the number of SpW ports. The third observation is that as the number of ports are increased the total number of packets drops.

The first observation can easily be explained by taking into account that each DMA transaction has an overhead for bus arbitration and fetch of control information. As a two packets sequence is fetched in a single DMA transaction this overhead becomes less significant as it consumes a smaller amount of time per packet and therefore the performance of the system is increased.

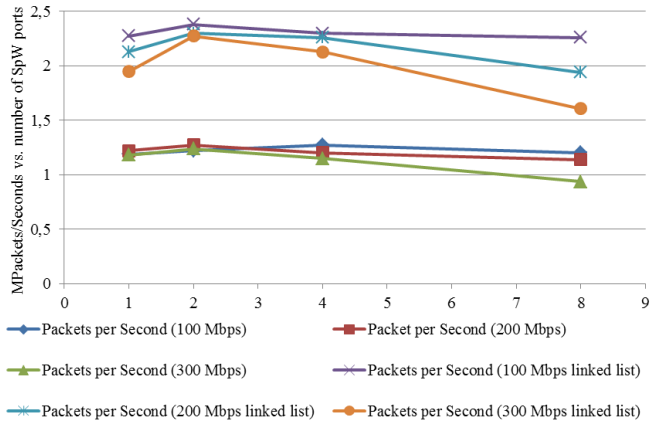

Fig. 8. Packets per Second vs. number of ports

As the link rate is increased the time to serve successive requests of the same SpW port decreases accordingly. Given that the DMA was programmed for round-robin operation, less time was left for successive requests of the same SpW port as the link rate increased almost linearly. This resulted in a packet size per port which increased almost linearly as the number of active ports increased (as shown in TABLE II. , thus resulting in an almost constant "packets per second" for all active ports thus explaining the second observation.

Nevertheless we observe a deviation from a straight line of constant overall performance. This happens because the DMA is serving more channels and time is lost in arbitration and serving other channels before fetching the next packet from the memory for the same port which explains the last observation.

From the diagram it becomes obvious that the Traffic Generation engine covers the requirements of all known systems since it is capable of obtaining a maximum throughput of more than 2 million packets per second. Saturation occurs at points which do not correspond to realistic scenarios, since high data rates are associated with science data which use large packets only.

IV. CONCLUSIONS

Both recording and simulation of flight devices can be very demanding for specific cases, like for example performance validation of Mass Memory Units or validation of mission equipment related to sky images acquisition and SAR. From the results presented herein it becomes obvious that the iSAFT is capable of simulating multiple flight devices that transmit data at very high throughputs and can also support time-accurate traffic shaping thus enabling microsecond-accurate simulation of a device's behaviour. The system also supports full-throughput 24/7 recording over multiple SpaceWire ports fulfilling the performance requirements for demanding scientific missions like the GAIA Video Processing Unit or the EUCLID Fine Guidance Sensor in which the throughput becomes challenging for many existing systems.

REFERENCES

[1] http://teletel.eu/isaft-protocol-validation-platform/

[2] http://teletel.eu/isaft-spacewire-mil-std-1553-can-recorder/

[3] Supporting development testbed – Protocol Validation System, 7th ESA Workshop on Avionics, Data, Control and Software Systems – ADCSS 2013.

[4] Evaluation, Assessment and Hardware prototyping of the SpaceWire-D protocol, DASIA 2013, 14- 16 May 2013, Porto Portugal.

[5] PVS Project Overview, 19th SpaceWire Working Group, 2-3 October 2012, Paris France.

Wednesday 24 September

Networks & Protocols (Short)

The evaluation of SpaceWire-R draft specification through the connectivity test using SpaceCube2

SpaceWire Networks and Protocols, Short Paper

Kaori Iwase, Hiroki Hihara
NEC TOSHIBA Space Systems, Ltd.
10, Nisshin-cho 1-chome, Fuchu, Tokyo, 183-8551, Japan
k-iwase@wr.jp.nec.com, h-hihara@bc.jp.nec.com

Takahiro Yamada
Institute of Space and Astronautical Science (ISAS),
Japan Aerospace Exploration Agency (JAXA),
3-1-1 Yoshinodai, Sagamihara, Chuo-ku, Kanagawa
229-8510, Japan
tyamada@pub.isas.jaxa.jp

Osamu Watanabe, Takahiko Tanaka
Space systems division,
NEC Corporation
10, Nisshin-cho 1-chome, Fuchu, Tokyo, 183-8501, Japan
o-watanabe@ak.jp.nec.com, t-tanaka@dy.jp.nec.com

Takayuki Yuasa
RIKEN The Institute for Physics and Chemical Research
2-1 Hirosawa, Wako, Saitama 351-0198, Japan
Takayuki_yuasa@riken.jp

Takayuki Tozawa, Toru Tamura
Embedded Systems Division, NEC Solution Innovators, Ltd.
2-2-41 Eki-mae, Kashiwazaki, Niigata 945-0055, Japan
tozawa-takayuki@mxv.nes.nec.co.jp,
tamura@mxm.nes.nec.co.jp

***Abstract*— SpaceWire-R is a protocol that provides onboard applications with reliable high-speed data transfer services over SpaceWire links especially for mission data transmission between sensors and data recorders. Independent implementations on different hardware with reference to the draft specification have been succeeded in interoperability test, which resulted in consolidating the protocol. The final specification document has been under preparation by JAXA. The present paper describes a result of interoperability test and evaluation of SpaceWire-R performed from 2013 to 2014.**

***Index Terms*— SpaceWire, Networking, Point-to-point link, SpaceWire-R.**

I. Introduction

Japan Aerospace Exploration Agency/Institute of Space and Astronautical Science (JAXA/ISAS) is consolidating the final specification of SpaceWire-R, which is intended to be used in mission data transmission between sensors and data recorders that require high-speed and reliable data transfer over SpaceWire links, where the transmission is carried out independently with SpaceWire network in the satellite bus. The protocol is also expected to be used between high speed optical sensors and solid state data recorders on upcoming scientific satellite projects.

The present paper describes a result of interoperability test and evaluation of SpaceWire-R performed from 2013 to 2014. SpaceWire-R is a protocol that provides onboard applications with reliable data transfer services over SpaceWire networks. The primary objective of SpaceWire-R is to transfer data reliably from a sending node to a receiving node over a SpaceWire network. To ahieve this, SpaceWire-R provides specifications on multiplexing of multiple communication channels, segmentation of packets, retransmission in case of packet loss, flow control, heartbeat, and redundancy control.

The protocol specification has been available among the SpaceWire Working Group members since 2012, and discussion of protocol ID assignment has been talked over in 21st Working Group meeting.

Through the interoperability test, basic functions of SpaceWire-R has been confirmed to work expectedly, and the two implementations succeeded to continue transferring data even with error injection owing to the retry mechanism in the protocol. Flow control and heart beat mechanisms were also confirmed to work. Based on the test result, some modifications have been incorporated to the protocol specification, and the final specification document has been under preparation by JAXA.

II. The purpose of interoperability test

The test was the second interoperability test between JAXA and NEC/NTSpace, and it was supposed to be the last evaluation step in order to consolidate the SpaceWire-R specification. The purpose of the test is to evaluate new functions added to the draft specification as a result of the

previous evaluation, and to confirm SpaceWire-R draft specification through the implementation on real hardware with SpaceWire interface

The new functions added to the previous draft specification are heart beat and flow control. They are categorized as optional mechanisms on SpaceWire-R.

Heart beat is a confirmation mechanism for a sending node and a receiving node when there is no data to send or receive whether the link between two nodes is kept and the both nodes are still alive. A timer is to be provided on a node in order to specify how long the node will be able to wait after sending a heart beat packet until it receives the heart beat acknowledgment packet from the other node. When the sender of the heart beat packet doesn't receive the acknowledgment corresponds to the heart beat within the specified limitation for the counter, the sender detects time out and cease the transaction cycle.

Flow control is a mechanism for a receiving node to tell a sending node how much the buffer capacity remains for receiving data units in order to suppress the sending node to send excessive data unit beyond the capacity of the buffer in the receiving node. The number of data units that receiving node can receive is shown as Maximum Acceptable Sequence Number (MASN), which is contained in a data acknowledgment packet or a flow control packet.

We evaluated the specification through three steps. The first step is to confirm the basic function described in the specification. The second step is to confirm continuous transmission between the sending node and the receiving node with error injection. And thirdly, the new functions have been confirmed, which are heart beat and flow control mechanism as described above.

III. TEST CONFIGURATION AND CONTENTS

Configuration of the test is shown in Fig.1. In order to perform the interoperability test, JAXA/ISAS and NEC/NEC Toshiba Space Systems (NTSpace) implemented the SpaceWire-R protocol as two independent software stacks on different hardware based on the draft specification. One was software for an ordinary Personal Computer (PC) with UNIX-based OS and Intel CPU. For convenience, we call this SpaceWire-GigabitEther(SpW-GbE) which is a conversion interface used between SpaceWire and GigabitEther. The other one was a SpaceCube2 (SpC2) on which a TRON based real-time OS is running. Its central processing unit (CPU) was commercial level with devices which was the same type as a flight qualified one. SpC2 has one 64bit microprocessor and one SpaceWire router in itself. The processing cycle of the microprocessor is 33MHz. The router has 6 external SpaceWire ports and the link rate of each port is 50MHz. The software stack for SpC2 is shown in Fig2.

SpW-GbE and SpC2 are connected each other with a SpaceWire cable.

Three test procedures were performed in other to consolidate the final draft version of SpaceWire-R specification.

A. Test case 1: Sending/Receiving data

Test case 1 was to confirm the basic functions. A sending node sends the data packet and then checks whether a receiving node receives the packet and responds with an acknowledgment packet. If the receiving node sends the acknowledgment packet properly, a sending node is to be checked if it receives the acknowledgment packet properly.

The test was performed for both directions between the two nodes in order to check two types of data transfer A and B as follows,

A. a sending node is the SpW-GbE, and a receiving node is the SpC2.

B. a sending node is the SpC2, and a receiving node is the SpW-GbE. These two data transfer tests are shown in Fig.3 and Fig.4, respectively.

The result of test case 1 is shown in Table.1 No.1 and No.2. The test was successfully done. Each packet which was expected to be received on each node was received properly.

B. Test case 2 :Error injection

Test case 2 was error injection. We made a receiving node stops to send an acknowledgment packet one time out of ten during data packet transactions intentionally. The purpose of this test was to confirm continuous transmission with correct responses between the sending node and the receiving node even in this situation.

In order to simulate the error case, the receiving node was programmed to suppress acknowledge response sometimes intentionally. The receiving node was programmed to suppress sending acknowledge response once per ten transactions in this case. And then it is checked whether a sending node resends the data packet. When a sending node resends the data packet, a receiving node is checked whether it receives the data packet and sends the data acknowledgment packet properly. When a receiving node sends a data acknowledgment packet properly, it is checked whether a sending node receives the data acknowledgment packet. The test was performed in both directions between the two nodes in order to check two types of data transfer A and B as follows.

A. a sending node is the SpW-GbE, and a receiving node is the SpC2.

B. a sending node is the SpC2, and a receiving node is the SpW-GbE.

Test configuration for case A and case B of test case 2 are shown in Fig.5 and Fig.6, respectively.

The result of test case 2 is shown in Tab.1 No.3 and No.4. The test was successfully done, and it was confirmed that the transmission between two nodes returned to normal sequence through the retry mechanism after the error injection. And both nodes acted how we expected and each packet was received as expected on each node.

C. Test case 3 : Two new functions

Test case 3 was to confirm heart beat mechanism and flow control mechanism.

1) Test case 3-1 : Heart beat mechanism

Two patterns of tests were performed in order to evaluate the heart beat mechanism. They were the normal pattern and the off nominal pattern.

As for the normal pattern, a sending node sends heart beat packets normally. The sending node sends a heart beat packet to a receiving node, receiving node is checked whether it receives the heart beat packet and sends a heart beat acknowledgment packet. If the receiving node sends the heart beat acknowledgment packet properly, the sending node is checked whether it receives the heart beat acknowledgment packet properly. We used a SpW-GbE as a sending node, and a SpC2 as a receiving node, for the test of normal pattern as shown in Fig.7 .

The result of the normal heart beat test is shown in Table.1 No.5. It was successfully done with each packet received as expected on each node, and it was confirmed to be received properly.

Off nominal pattern is that a receiving node is not set to send a heart beat acknowledgment packet. The sending node sends a heart beat packet and the receiving node receives the heart beat packet, whereas the receiving node does not send a heart beat acknowledgment packet this time. In consequence the sending node is checked whether it moves into terminate transmission state after the sending node sends several heart beat packets. We used a SpC2 as a sending node, and a SpW-GbE as a receiving node, for the test of off nominal pattern as shown in Fig.8.

The result of off nominal heart beat test is shown in Table.1 No.6. It was successfully done with each node acted as we expected. We set the timer to 5seconds at that time, and the heart beat packet time interval was set to 1second. After the sending node sent four heart beat packets, it judged timeout and terminated the transmission transaciton.

2) Flow control mechanism

The final test was to confirm flow control mechanism. A sending node sends a data packet with a smaller sequence number than the MASN held by a receiving node. Then sequence number of a next sending data becomes equal to the MASN of a receiving node, the sending node stops to send a data packet. After the sending node stops sending a data packet, the receiving node sends a flow control packet which contains larger MASN number. A sending node is checked whether it receives the flow control packet and sends a flow control acknowledgment packet. If the sending node sends the flow control acknowledgment packet properly, then the receiving node is checked whether it receives the flow control acknowledgment packet properly. The sending node is also checked whether it restarts sending data packets. If the sending node restarts to send data packet successfully, then the receiving node is checked whether it receives the data packet and sends the data acknowledgment packet. In the end the sending node is checked whether it receives the data acknowledgment packet or not. We used a SpW-GbE as a sending node and a SpC2 as a receiving node, for the test of flow control as shown in Fig.9.

The result of the flow control test is shown in Table.1 No.7. It was successfully done with each node acted as we expected, and each packet which expected to be received at each node was confirmed to be received properly.

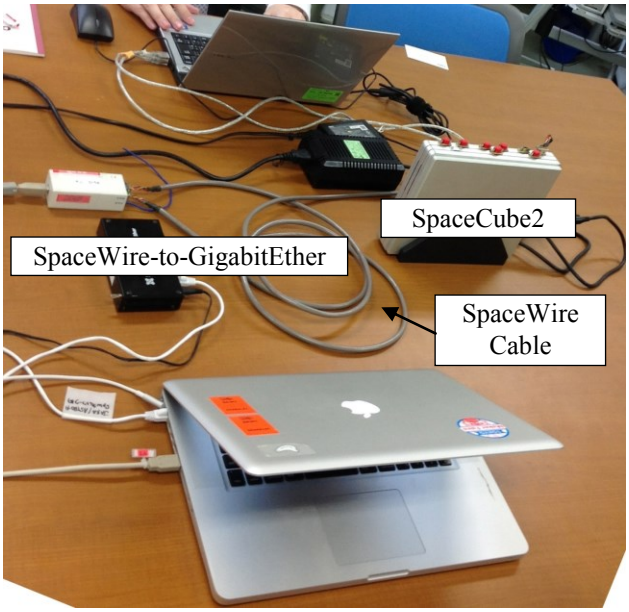

Figure1. The configuration of the evaluation test

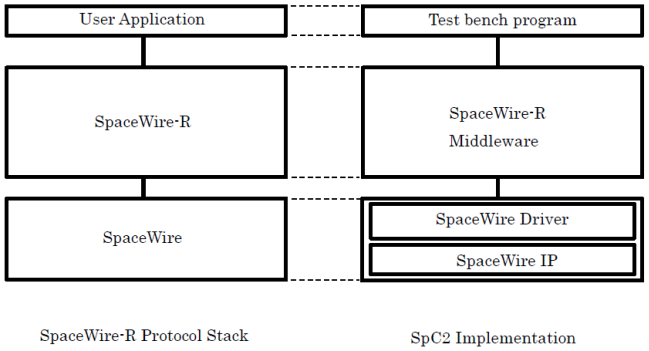

Figure2. Implementation of SpC2 for SpaceWire-R

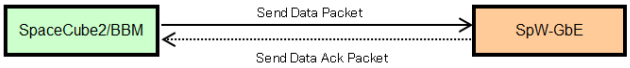

Figure3. Test case 1 – Case A

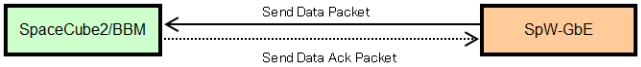

Figure4. Test case 1 – Case B

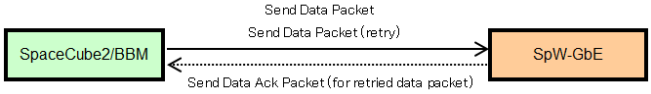

Figure5. Test case 2 – Case A

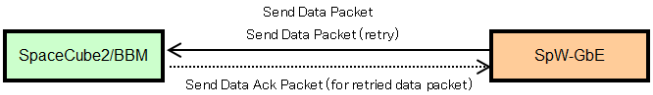

Figure6.Test case 2 – Case B

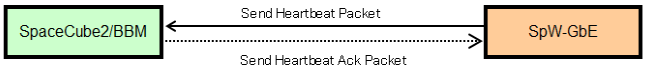

Figure7. Test case 3-1 – Heart beat normal case

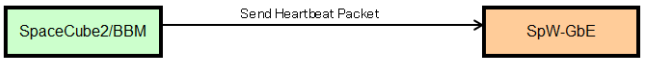

Figure8. Test case 3-1 – Heart beat off nominal case

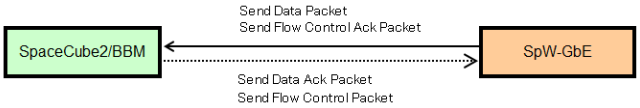

Figure9. Test case 3-2 – Flow control

Table1. Result of interoperability test

	Test items		result
No.	Packet direction	*Expected action*	
1	A)SpC2→SpW-GbE	SpC2: Data packet send	Pass
		SpW-GbE: Data packet receive	Pass
		SpW-GbE: Data Ack packet send	Pass
		SpC2: Data Ack packet receive	Pass
2	B)SpW-GbE→SpC2	SpW-GbE: Data packet send	Pass
		SpC2: Data packet receive	Pass
		SpC2: Data Ack packet send	Pass
		SpW-GbE: Data Ack packet receive	Pass
3	A)SpC2→SpW-GbE	SpC2: Data packet send	Pass
		SpW-GbE: Data packet receive	Pass
		SpW-GbE: don't send Data Ack packet	Pass
		SpC2: Data packet resend	Pass
		SpW-GbE: Data packet receive	Pass
		SpW-GbE: Data Ack packet send	Pass
		SpC2: Data Ack packet receive	Pass
4	B)SpW-GbE→SpC2	SpW-GbE: Data packet send	Pass
		SpC2: Data packet receive	Pass
		SpC2: don't send Data Ack packet	Pass
		SpW-GbE: Data packet resend	Pass
		SpC2: Data packet receive	Pass
		SpC2: Data Ack packet send	Pass
		SpW-GbE: Data Ack packet receive	Pass
5	normal) SpW-GbE →SpC2	SpW-GbE: Heart beat packet send	Pass
		SpC2: Heart beat packet receive	Pass
		SpC2: Heart beat Ack packet send	Pass
		SpW-GbE: Heart beat Ack packet receive	Pass
6	unusual) SpC2→ SpW-GbE	SpC2: Heart beat packet send	Pass
		SpW-GbE: don't send Heart beat Ack packet	Pass
		SpC2: Heart beat packet send	Pass
		SpW-GbE: don't send Heart beat Ack packet	Pass
		SpC2: Heart beat packet send	Pass
		SpW-GbE: don't send Heart beat Ack packet	Pass
		SpC2: Heart beat packet send	Pass
		SpW-GbE: don't send Heart beat Ack packet	Pass
		SpC2: Close	Pass
7	Data packet: SpW-GbE →SpC2	SpW-GbE: Data packet send	Pass
		SpC2: Data packet receive	Pass
		SpC2: Data Ack packet send	Pass
		SpW-GbE: don't send data packet	Pass
	Flow control	SpC2: Flow control packet send	Pass
		SpW-GbE: Flow control packet receive	Pass
	packet: SpC2→ SpW-GbE	SpW-GbE: Flow control Ack packet send	Pass
		SpC2: Flow control Ack packet receive	Pass
		SpW-GbE: Data packet send	Pass
		SpC2:Data packet receive	Pass
		SpC2: Data Ack packet send	Pass
		SpW-GbE: Data Ack packet receive	Pass

IV. TEST RESULT

All test cases have been completed successfully. The transmission rate for test case 1 was 400kbps at 50MHz link rate. At test case 2, the test was held in 10MHz link rate and the transmission rate was 8kbps, where it was 80kbps at same link rate with no error injection. Error rate injected in the test case 2 was 10%, and the transmission rate decreased in 90%. The result was corresponds to the analysis performed in advance.

We confirmed all of the test procedures and results as expected. No modification was identified with the draft version of SpaceWire-R specification. In consequence we confirmed that the specification of SpaceWire-R had been consolidated.

V. CONCLUSION

We evaluated the SpaceWire-R draft specification through the test using real hardware on which SpaceWire-R protocol is implemented. The evaluation tests of SpaceWire-R draft specification using a SpW-GbE and a SpC2 were completed successfully, and the specification has been confirmed through the three test procedures as follows. First, the basic function has been confirmed. Second, continuous data transmission against error injection has been confirmed. Third, the new functions, which are heart beat mechanism and flow control mechanism, have been confirmed.

In accordance with the result of the evaluation, no more modification was identified as required. As a result, SpaceWire-R specification has been establised.

ACKNOWLEDGMENT

We would like to thank Dr. Richard D. Hunt of Sandia National Laboratories and Dr. William H. Anderson of NASA/GSFC for their RDDP related advice.

REFERENCES

T. Yamada,, "SpaceWire-R SCDHA 151-0.3", Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency.

Automated SpaceWire Network Administration

SpaceWire Networks and Protocols, Short Paper

Khramenkova Ksenia
SUAI
Saint-Petersburg, Russia
ksenia.khramenkova@guap.ru

***Abstract*— Creation and launching a spacecraft are very expensive measure; it needs a great accuracy and attention to details. A spacecraft board network includes: system of the navigation, orientation and stabilization, special target devices giving a different payload. Success of work whole spacecraft and the realization its tasks and functions is depending on speed, quality and correctness. It is necessary not only to initialize the huge number of devices during the its interaction. The generation the output log-files and reconfiguration the each device also are the very important tasks. The adjustment of every network device is necessary and very significant because the correctness of all network functioning depends on that. In addition, the error situations can occur in network, then we need to have the opportunity to process them, recover the system efficiency with minimal loss of information.**

The special software created for the configuration, administration and monitoring the device status is the essential component of a creation every network. This software provides for proper setting and supporting of each network device an ability to work, decreases the error appearance and falling out an network element or part of the network and, of course, essentially reduces time of tuning of all network.

At this paper the software adjusting SpaceWire network by required type without human participation and granting results as output files is described.

Index Terms — SpaceWire network, Plug-and-Play, administration, configuration.

I. Introduction

Generally the actual networks consist of a huge number of devices (nodes and switches) located on different distance of each other. For organization its interaction it is necessary to execute the primary configuration in compliance with a structure of physical links and logical channels between applications, which owe exist in the system.

During the time of network functioning it is necessary to have the capability to monitor the state of network components and system work modes, support the network work correctness and accuracy its settings.

Even if the network consists of a little number of devices, the manual executing primary configuration also is not recommended, because human-operator mistakes can lead to unpredictable results. For huge networks, to which the modern networks belong, the manual configuration is impossible because it will be executed unacceptably long.

During the system functioning some devices or links between them can fail, as a result of that failures the network settings can be distorted. The monitoring of network states and correctness of network settings is necessary for detection the failures and correction of consequences of failures throughout all time of functioning. These actions have to execute automatically.

Based on the explained ideas the software allowing to execute discovering and network setup was created. The responsibility for correct adjustment the main registers of switch and node is settled on this software. Also the notification the network operator about a network setup results.

This technique is intended for administration of distributed SpaceWire network [1]. For execution of process of administration and setup network system components the RMAP protocol is used [2]. This protocol represents one of transport layer protocols which can function on the SpaceWire network. This protocol is intended for access to address space of nodes of a network and here is used for remote configuring of devices with SpaceWire interfaces.

II. The Generalized Algorithm

First of all it is necessary to select network connection point for the manager of a network.

In order that time of configuring and status inquiry of a network was minimum, and also there was minimum the volume of traffic generated in case of status inquiry of a network, expediently that a node - the network administrator was located at some "center" of a network. Node-Administrator in a network is connected to communications system through special official port, it doesn't require use the main ports on the basis of which the network is built.

The choice of network connection point of a node of a network on which functions of administration of a network will be executed, shall be carried out by the engineer performing tuning of a network, taking into account structure of a network and taking into account types of network points (a node – the network administrator can be connected only to nodes of those types for which in library possibility of such connection is set). After the network connection point the engineer performing tuning of a network is selected, shall connect to it a node on which beforehand it shall be set by an administration software.

It is supposed that on all switches in network a software is functioning, which execute the following functions:

- connection establishment an all required ports;
- setup the registers of the adaptive routing group;
- support of path addressing;
- processing the RMAP request packets and formation a responses on them.

The process of configuring includes two stages:

- check that all communication links required in system are connected, set a transmission rate on them, check after setup rates, that connection on all required links is still set;
- if the first stage was executed successfully (on all required links connection is set, transmission rate corresponds required), filling the adaptive routing registers an routing table of all switches.

Configuration is carried out by RMAP commands packets which the network manager delivers to switches and terminal nodes. Distribution command packets is realized in ascending order of length of the path address. At first the configuring of the switches/nodes directly connected to the network manager is executed, further – configuring of the switches/nodes connected to them, and so until the configuring of all available devices of the network are executed.

The administration process following it includes two stages:

- check that on all required communication links in system the connection is established and required transmission rate is set;
- check the contents of registers of the adaptive routing registers and routing tables.

In the course of network administration during its state processing the check of a status of devices is executed in the same order as in train of initial setup of system configuration.

III. The description of original setup of a network

The primary network setup provides installation all parameters of configuring of terminal nodes and switches in SpaceWire network according to a required network configuration.

The process performing the original setup consists of the following stages:

- check that all communication links required in system are connected, set a transmission rate on them, check after setup rates, that connection on all required links is still set;
- if the first stage was executed successfully (on all required links connection was set, transmission rate corresponds required), filling the adaptive routing registers an routing table of all switches;
- if the first stage is not executed successfully, these is a rearrangement of settings and scheme of a network structure.

All process of administration is carried out by RMAP packets. Distribution command packets is realized in ascending order of length of the path address.

After completion of operation of setup process the log file will be created which will give to the human operator opportunity to look at results of primary network setup. If errors were revealed, the operator should take necessary measures, for example, check communication links on which connection was not set, or test devices.

IV. The description of software operation in case of standard network functioning

After execution of network configuring at our disposal are list of paths to devices and list of found devices.

The program creates a packet for reading the register of connections. In the course of administration the device status checking is executes in the same order as in the course of configuring. If data accepted from device correspond required, the program creates a packet for reading transiting speed value on the specified ports. The derived data also are compared to what are specified in the input file. If any connection is gone, the software makes attempt to recover it and to set required speed, the number of attempts is restricted. If it works well, in case of switch administration there is a transition on second administration step, otherwise the device is admitted as inoperable and deleted from both lists by software of network administration.

The described step is necessary only for switches. On this step the program executes check a correctness of setup a routing table and adaptive group registers. At first the packet for reading a routing table is created. From an answer packet we accept data and we compare to what shall be written. If data don't match, a defined number of attempts to recover data, which were written by a software on the configuration stage, is executed. Further the software creates a packet for reading the adaptive group registers and process check retrieved data. If data don't match again, a defined number of attempts to recover data, which were written by a software on the configuration stage, is executed. Results are written to the output file.

V. Basic data for network setup

Because the network structure is known in advance, but is not considerate that some devices or communication links can be invalid, so input data for setup is concerning to each network element.

All ports within each terminal node and switch have unique number (these numbers match physical numbers of ports on devices). These numbers are used for identification of communications links in the table of communications and for formation the path addresses.

Owing to that in the initial status nodes of a network have not logical/regional addresses, during configuring and administration path addressing in the SpaceWire network is used. In case of path addressing the destination address represents sequence of numbers of output ports of switches through which shall pass this packet. When using path addressing the main part of formation of a way of a packet in a network lays down on administration node, switches need to direct only a packet with input on the output port, thus any

registers of setup of modes of routing, a routing table in the switch aren't used.

In an absence situation in the routing switch of incorrect setup of a routing table the exchange of packets between devices when using logical addressing will be erratic. Such situation often takes place in nodes switches in start state, after switching on. Errors in the table can result and power failures. In order to avoid erratic situations during Administration of a network traveling addressing as it represents a set of output ports of switches through which shall pass a packet to reach an assignment node is used. The routing table thus isn't involved.

The network structure specification is set on the basis of the table in which unique numbers of connected nodes and ports connected by communication links between this node are specified.

For each terminal node:

- the list of ports, where shall be set connection, is set;
- for each port the required transmission rate is set;

For each switch:

- the list of ports, where shall be set connection, is set;
- for each port the required transmission rate is set;
- values for adaptive group routing registers are specified;
- the contents for routing table are set.

All basic data are provided in the form of one file having a XML format.

The output data represent:

- the list of communication links on which there is connection, speeds at which it was succeeded to set connection;
- if on all required communication links there is connection, for all switches fields for adaptive group routing registers and routing table are created.

An output data represent one file in xml format. This file includes information on what setup is implemented according to input data and what is not implemented. The output file can be analyzed by the network administrator. If in network there are some failures, administrator can concentrate on them.

A. Format of an input file

This file is partitioned into the following main sections:

- network_structure has the following structure. To every line of this table specifying structure of a network, there corresponds network_connection subsection (the number of subsections corresponds to number of lines in the table). Each such subsection has the unique identifier corresponding to a unique identifier of a specific communication line, is set in the form of parameter. Further all communication lines are described. As the communication lines bidirectional, for it can't be defined a concept of the beginning and end. Therefore designation of network connection points is used. Network connection points describe number of the device and port number on which the communication links is connected;
- master_node defines the characteristic of the device which executes configuration functions and network administrations. This device has the unique number, for setup he needs a number of attempts to set a value for any parameter for any device of a network, the COM port name which will be used for interaction with a network and its speed.;
- switch_parameters consists of subsections of switch in which the specification of parameters of each switch which is a part of system is executed. The quantity of subsections of switch is equal to number of switches in system. In switch subsection in the form of parameters the identifier of a node and number of ports are set. For the separate switch parameters for setup is:
  - unique device number;
  - the list of port numbers specifying on existence connection or absence connection on each of them and in case of connection existence, speed for transmitting speed;
  - the description of adaptive group routing registers, separately for each register;
  - the routing table description, row number specifying and value which needs to be written;
- terminal_nodes_parameters consists of subsections of terminal_node in which the specification of parameters of each terminal node which is a part of system is executed. The quantity of subsections of terminal_node is equal to quantity of terminal nodes in system. In terminal_node subsection in the form of parameters the identifier of a node and number of ports are set. For each terminal node input parameters is:
  - unique device number;
  - the port list specifying on existence or absence of connection on each of them, in case of existence, a speed value.

In addition to an XML input file it is necessary to correctly specify temporal settings for network administration:

- parameter specifying a period after which it is necessary to realize administration;
- parameter specifying total quantity of cycles of administration.

B. Format of an output XML file

Output data are:

- the list of communication lines on which there is a connection, speeds at which it was succeeded to set connection;
- if on all required communication links there is connection, for all switches fields for adaptive group routing registers and routing table are created.
- file includes information on what setup is implemented according to input data and what is not implemented;
- the output file includes two sections:
  - switch_parameters – section in which are specified parameters of switches;
  - terminal_nodes_parameters – section in which are specified parameters of terminal nodes.

VI. Conclusion

Administration of devices of a distributed network is made both in a standalone mode, and as a part of an exploited complex. This algorithm provides automatic detection of the current configuration of a network. The developed software allows to trace attaching and detaching (an output from structure and a failure) network devices. Allows to set up quickly operation modes and to trace statuses of devices, provides collection, information display about network condition.

The operator of a network can look at results of operation in the output file. If it contains devices for which it wasn't succeeded to set required parameters, the operator can make the relevant decisions for deleting the arisen problem.

References

[1] ESA (European Space Agency), standard ECSS-E-50-12A, "Space engineering. SpaceWire – Links, nodes, routers and networks. European cooperation for space standardization", ESA Publications Division ESTEC, Noordwijk, The Netherlands, 2003

[2] ECSS-E-ST-50-52C. Space engineering. SpaceWire - Remote memory access protocol. Noordwijk, The ESA-ESTEC Requirements and Standards Division, 5 February 2010, 109 p.

Impacts of Faults on a SpaceWire Network

Session: SpaceWire Networks and Protocols, Short Paper

Michiya Hayama, Yosuke Yokoyama, and Riko Yagiu
Information Technology R&D Center,
Mitsubishi Electric Corporation,
5-1-1, Ofuna, Kamakura, Kanagawa, 247-8501, Japan
Hayama.Michiya@db.MitsubishiElectric.co.jp,
Yokoyama.Yosuke@ea.MitsubishiElectric.co.jp,
Yagiu.Riko@eb.MitsubishiElectric.co.jp

Isao Odagi, and Hiroto Namikoshi
Kamakura Works,
Mitsubishi Electric Corporation,
325, Kamimachiya, Kamakura, Kanagawa, 247-8520, Japan
Odagi.Isao@cb.MitsubishiElectric.co.jp,
Namikoshi.Hiroto@bk.MitsubishiElectric.co.jp

***Abstract*—SpaceWire is a standard for a communication interface in a satellite. It reduces a wiring cost for building a network. However, it becomes the complicated topology when we design the network architecture that considered Peer to Peer, a redundancy, etc. Furthermore, a component can share a line with the other components. Therefore, traffics generated by a component will effect to the other traffics. It is difficult to estimate traffics on a network. Additionally, estimating impacts on SpaceWire network by errors related to internal registers and buffers are difficult because SpaceWire has more complex protocol than that of the other communication interfaces such as MIL-STD-1553b. We developed a simulator based on NS-3 to simulate VHDL models with virtual network. We divide a SpaceWire network into function blocks (internal registers, buffers, communication lines etc), and evaluate impacts on each function block by faults. In this paper, we describe simulation results and propose necessary quality of each function block for fault tolerance system with SpaceWire.**

***Index Terms*— SpaceWire, Network, Simulator, VHDL, Faults Detection**

I. Introduction

SpaceWire has been used for internal communication interfaces in satellites [1-3] because it achieves high-speed data communication and flexible network topology. The main target satellite system is an unmanned and once the satellite is orbited, it is extremely difficult to maintain it manually. These systems have FDIR (Fault Detection, Isolation and Recovery) mechanism that detects faults and recoveries the systems automatically.

It is important to understand detail effects of faults and evaluate effectiveness of fault detection and countermeasures for faults in advance. Depending on this background, effects of faults on a SpaceWire network and countermeasures against faults are presented [4-9].

Generally, implementation bugs, Single Event Upsets (SEU) and Single Event Transients (SET) cause faults on SpaceWire nodes and routers. In [4,5], effects of SEU/SET on SpaceWire IP (Intellectual Property) is analysed using HDL simulator. However, estimating consequence from failure parts to the effects for network is difficult, because they inject faults at random.

TABLE I. Characterristics of The Simulators

Parameter	OPNET[10]	HDL Simulator[11]	Proposal[12]
Simulation Time	Fast	Slow	Middle
Accuracy	Character Level	Logic Level	Mixed
Open Source	No	depends on the simulator	Yes

Regarding countermeasures against faults in SpaceWire, it is proposed methods to construct redundant network with various constraint conditions automatically [4] and switch to the alternative route by fault detection with exchanging keep-alive messages [5].

Therefore, we developed a simulator that is able to handle HDL models with virtualized networks in order to evaluate effects of faults on a SpaceWire network and effectiveness of FDIR accurately [12]. Furthermore, we simulate behaviors of existing SpaceWire IPs with a redundant network using the simulator. The results of evaluation show relations (between failure parts and effects on a network) and a delay of a redundant operation. We also found that there are some cases which cause delays to switch route in redundant network due to undetected faults for a long time and it depends on implementations and traffic conditions. In addition, we propose requirements for SpaceWire implementations based on the results.

II. Overview of the Simulator

Simulators for SpaceWire have been reported in [10-11]. To evaluate for the effect on SpaceWire networks, simulation models should be close as possible to actual environment. The simulator [10] based on the network simulator (OPNET) has problems for the point of accuracy. Simulating whole system with HDL simulator achieves high simulation accuracy [11]. However, the simulator requres large amount of calculation. In this paper, we report our simulator which works as a mixed-mode simulator composed with the network simulator (NS-3)

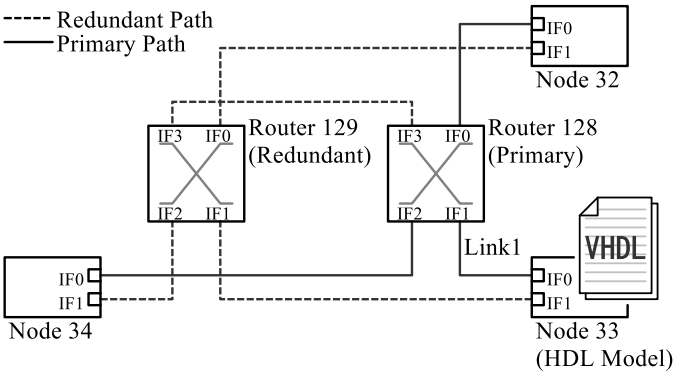

Fig. 1. Evaluation Network

TABLE II. NODE MODELS FOR EVALUATIONS

Models	HDL Model A	HDL Model B	Default Model
Description	Free Spacewire IP	Open-source SpaceWire IP Core	Ideal Model
Language	VHDL	VHDL	C++
FIFO	128 bytes	64 bytes	1024 bytes
Clock	10 MHz	Rx:166 MHz Tx:100 MHz	—
Max Credit	7	7	4

and the VHDL simulator (FreeHDL) [12].

The simulator is able to simulate large scale systems efficiently. Table I shows a comparison characteristics among simulators.

III. EVALUATION MODELS

Figure 1 shows the target network model to evaluate that is configured with 3 nodes and 2 routers. Various HDL models are applied to Node 33. We inject a bit-error into internal registers and evaluate effects to the network. For evaluate the switching redundant routes time, we implemented redundant functionalities described in Section C in each node and router.

A. Simulation Senario and Analyze the Simulation

We executed simulation 200 times per combinations of model (Model A and B) in Table II, register to inject an error in Table III and condition in Table V according to the scenario as bellow; After termination (Step 3), we compared transmitted packets with received packets and evaluated number of incorrect packets.

1) Reset Nodes and Make Traffics

Reset all nodes and routers after 20 μs from the timing to start simulation and each node starts to transmit packets.

2) Inject a Bit-Error to Node33

Injects a bit-error into an internal register of Node33 after 10ms from the timing to start simulation.

3) Terminate and Analyse the Simulation

Terminate the simulation after 100 ms from the timing to start simulation.

TABLE III. FUNCTION BLOCKS IN SPACEWIRE NODE MODELS

No.	Block	Function
1	State Machine	State Machine (FSM)
2	Transmitter	Read Pointer of FIFO
3		Write Pointer of FIFO
4		Transmit Credit Counter
5	Receiver	Read Pointer of FIFO
6		Write Pointer of FIFO
7		Receive Credit Counter

TABLE IV. ALTERNATIVE INTERFACE MAP

Node	IF 0	IF 1	IF 2	IF 3
Node32	IF 1			
Node33	IF 1			
Node34	IF 1			
Router128	IF 3	IF 3	IF 3	
Router129				

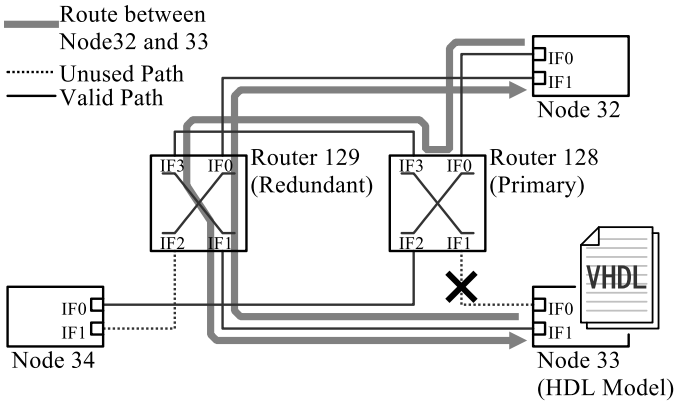

Fig. 2. Alternative Route after Fault Detection on Link1

B. Node Models

Table II shows simulation models. We use 2 types of HDL models for evaluation. The first model (HDL Model A) is released by CESR. The second model (HDL Model B) is developed in the open-source SpaceWire project. Each model is developed as an open-source model. The target bit-errors are injected into registers shown in Table III.

C. Fault Tolerant Methods

The method of fault detection and switching routes redundant network consists of following steps (1-3). We shows an example using transmit timeout on Interface0 of Node33 with Figure 2.

1) Packet Transmit Timeout

Each node resets a link, if the node does not detect transmission of EOP or EEP after a given time from the first data character transmission.

2) Packet Receive Timeout

Each node resets a link, if the node does not detect reception of EOP or EEP after a given time from the first data character reception.

TABLE V. EVALUATION CONDITION

Parameter		Condition A	Condition B
Number of Trials		200 (each HDL model and traffic condition)	
Transmit Timeout		1 ms	
Receive Timeout		1 ms	
Transmission Data		25 bytes + EOP	
Traffic Condition	Node33 to 32 (Constant Period)	800 pkt/s	1200 pkt/s
	Node32 to 33 (Poisson Arrival)	400 pkt/s (Average)	600 pkt/s (Average)
	Node34 to 32 (Poisson Arrival)	800 pkt/s (Average)	1200 pkt/s (Average)
	Node32 to 34 (Poisson Arrival)	400 pkt/s (Average)	600 pkt/s (Average)

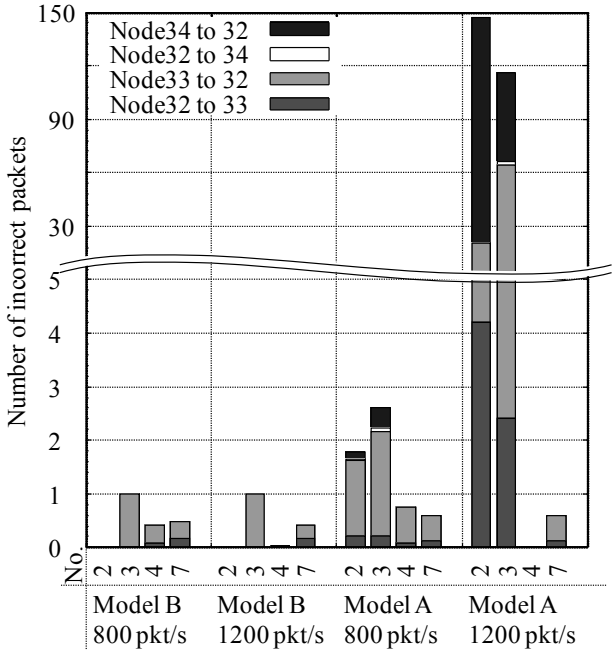

Fig. 3. Classify Incorrect Packets according to Source and Destination Address

3) Redundant Routing

Each node and router switches the routes to the alternatives according to Table IV if it detects link reset. This process runs only once.

When a transmit timeout occurs on Interface0 of Node33, each link with Interface0 of Node33 and Interface1 of Router128 is reset according to the Step1. Then, Node33 and Router128 update routing tables according to Step3 with Table IV. Alternative interfaces of Interface0 of Node33 and Interface1 of Router128 are defined as Interface1 and Interface3 respectively in Table IV. Node33 rewrites fields related to Interface0 as Interface1. And, Router128 rewrites fields related to Interface1 as Interface3. Finally, the route between Node33 and Router128 updates as Figure 2.

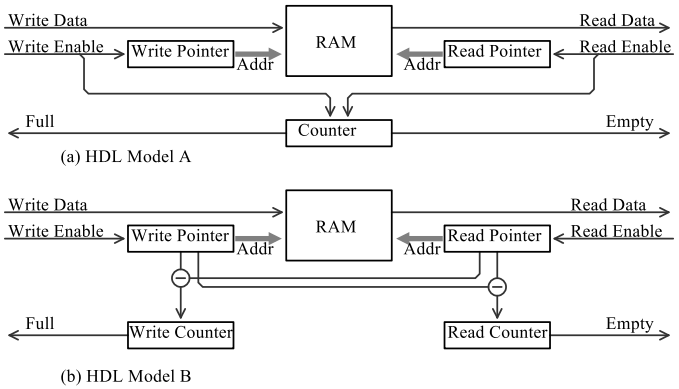

Fig. 4. Function Block Diagrams of FIFOs in HDL Model A and B

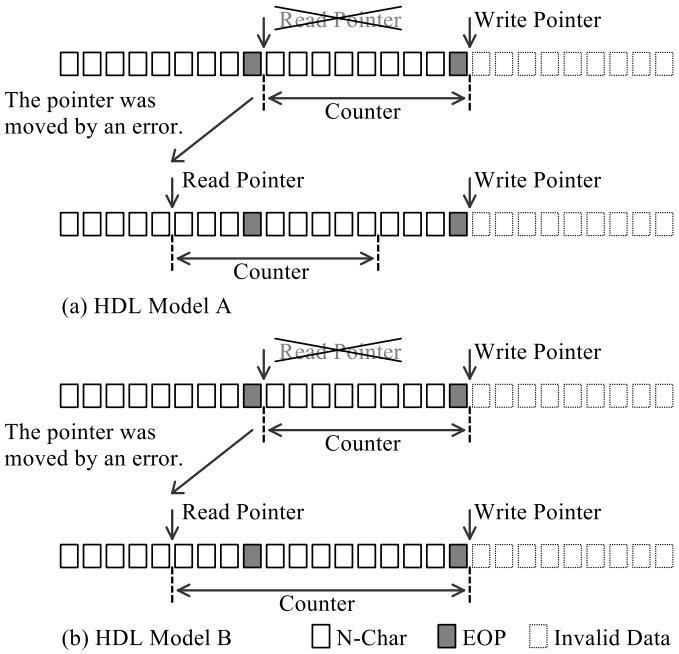

Fig. 5. Effects of an Error for Enqueued Packets in Model A and B

IV. EVALUATION RESULTS

We simulated the models according to Section III and evaluated number of incorrect packets and distributions of transmission times of incorrect packets. Table V shows simulation conditions. We ran the simulation with each HDL model (HDL Model A and B in Table II) and each traffic condition (Condition A and B in Table V). Each node generates packets with Poisson distribution except Node33. Node33 generates packets with constant period.

A. Effect of a Traffic Condition

Figure 3 shows the average number of incorrect packets when errors occur on internal registers No.2, 3, 4, or 7. Figure 11 shows all of the results. We found from Figure 3 that Model A caused more incorrect packets than Model B. In Condition B (1200 pkt/s), faults on Node33 had a major effect on packets transmitted from Node34.

We analyzed Model A and B in order to find the cause of the difference of number of incorrect packets. As a result of the

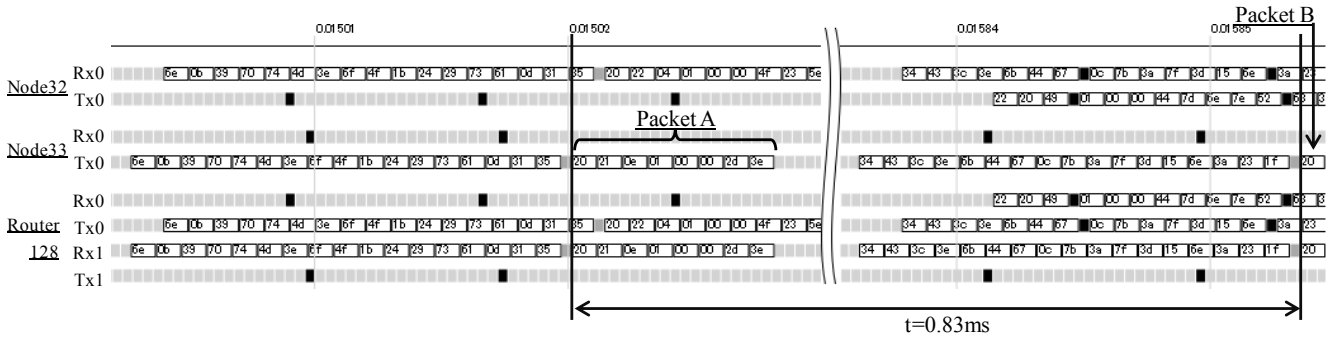

Fig. 6. Character Traces with HDL Model A (1200 pkt/s) and Injected Fault Type No.4

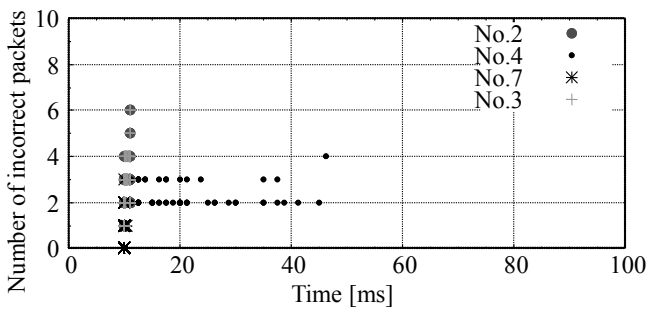

Fig. 7. Distributions and Number of Incorrect Packets with HDL Model A and 800 pkt/s on Node33

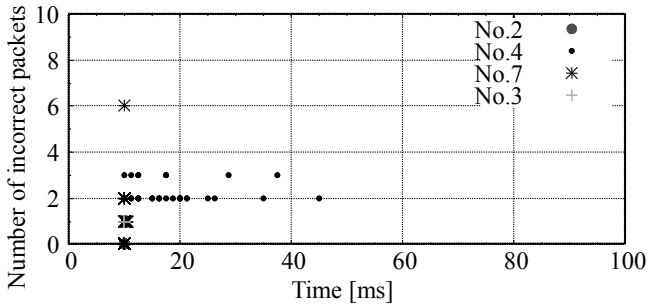

Fig. 8. Distributions and Number of Incorrect Packets with HDL Model B and 800 pkt/s on Node33

analysis, the structure of a FIFO in a transmitter and a configuration of transmit/receive timeouts caused the difference. The diagrams of FIFO in each model are shown in Figure 4. And effects of an error on enqueued packets in the FIFO are shown in Figure 5.

In Model B, as shown in Figure 5(b), all characters located between read address and write address are transmitted even if an error occurs because the number of characters stored in FIFO is calculated by subtracting read address from write address. On the other hand, in Model A, a part of characters located between read address and write address may not be transmited because the number of characters stored in FIFO does not change if an error occurs on read/write address registers, as shown in Figure 5(a).

If an EOP is not transmitted due to the error on read/write address registers, this can cause link occupation on SpaceWire network. Figure 6 shows character traces when a miss of a transmission of EOPs was occurred. We found from Figure 6

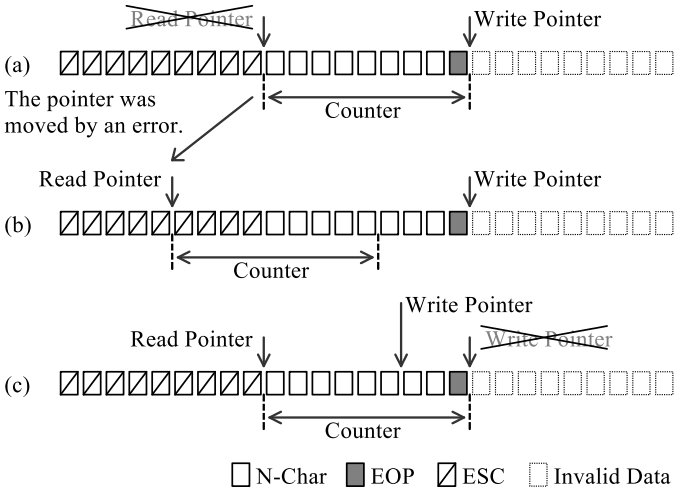

Fig. 9. An Operation of The Improvement of a Fault Detection

that Node 33 transmitted PacketA without EOP and continued to transmit NULLs for a long time. In addition, Node 33 could not detect the fault because the following PacketB was transmitted from Node 33 in 1ms. Finally, it caused occupation of the link between Router 128 and Node 32.

B. Delay of Fault Detection

Figure 7-8 shows distributions of transmission times of incorrect packets. We found that it takes 0-40ms to detect an error of register No.4 (Transmit Credit Counter). The delay of a fault detection is understood as follows. The register in Node 33 records the amount of receiver space in Router 128. Router 128 can detect the fault only when transmitted characters exceed receiver space in Router 128 due to the error. Therefore, the fault detection delayed for a long time.

C. Improvement of Fault Detection

According to Section III-A, when a packet transmission interval is shorter than a transmission timeout time, there is a possibility that Model A affects traffics of surrounding nodes because a fault detection does not work in some cases. In order to improve a fault detection capability of Model A, we implement the following method in the FIFO of the model. Figure 9(a)(b) show an operation of the improved FIFO. When a character is read, the FIFO writes an ESC character to the same address. The written ESC characters are transmitted when an error occurs in read/write address registers. This enables to

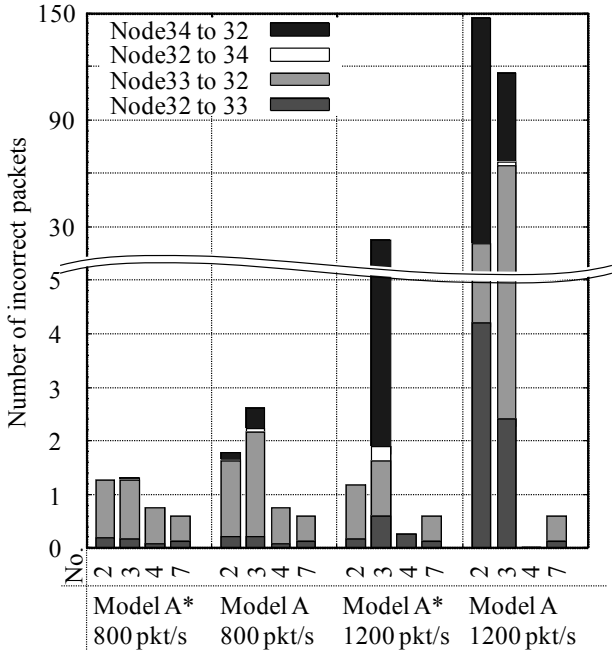

Fig. 10. A Comparison of Incorrect Packets between Model A without the Improvement and Model A with the Improvement

detect an error. (SpaceWire treat continued ESCs as an error.)

We run the simulation using the improved model with the same conditions as Section III. Figure 10 shows the average number of incorrect packets in 200 times simulations per each condition. Model A* shows the improved model. Incorrect packets decreased when an error occurred in the internal register No.3 (a read address register in the transmitter). When an error occurred in the internal register No.4, incorrect packets decreased by one quater, but we found misses of a fault detection. The misses are understood that the fault detection functionality detected only errors of read address register, as shown in Figure 9(c).

V. Conclusion

In this paper, we evaluated impacts of faults on a SpaceWire network and capabilities of redundant operations using existing HDL models. For the evaluation, we reported the simulator composed of the network simulator and VHDL simulator in order to simulate the models accurately and efficientry. The evaluation results indicate that an error in read/write address registers causes missing EOPs. Missing EOPs causes blocking traffics transmited by surrounding nodes. Futuremore, we found that detecting errors in Trasmit Credit Counter is delayed for a long time.

According to these analysis, we think FIFOs in SpaceWire IPs should be implemented avoiding mismatch between a address registers and a data counter. Also, a transmission timeout should be configured shorter than a packet transmission period.

Appendix

Figure 11 shows all of the evaluation results (internal registers No.1-7) in Section III.

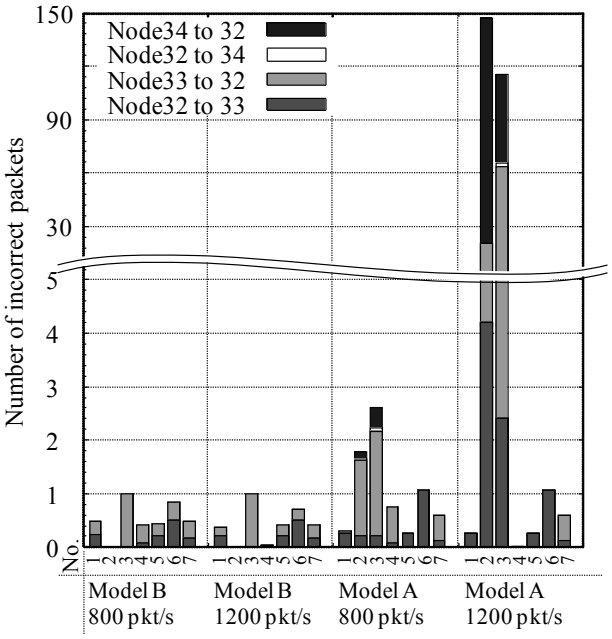

Fig. 11. All of the Evaluation Results (internal registers No.1-7) in Section III

References

[1] Hiroki Hihara *et. al.*, "Intelligent Navigation System with SpaceWire for Asteroid Sample Return Mission HAYABUSA2," Proc. of ISC 2013, pp.308-311, Jun. 2013.

[2] Satoko Kawakami *et. al.*, "Deterministic Implementation of SpaceWire on Data Recorder and Payload Interface Units", Proc. of ISC 2011, pp.189-192, 8-10 Nov. 2011.

[3] T. Takashima *et. al.*, "Space-Wire Applications for the MMO Spacecraft in BepiColombo Mission," Proc. of ISC 2007, 17-19 Sep. 2007.

[4] Jimmy Tarrillo *et. al.*, "Designing and Analyzing a SpaceWire Router IP for Soft Errors Detection," Proc. of LATW 2011, 27-30 Mar. 2011.

[5] Jimmy Tarrillo *et. al.*, "Improving Error Detection Capability of a SpaceWire Router IP," Proc. of RADECS 2011, pp.501-506, 19-23 Sep. 2011.

[6] Alexey Syschikov *et. al.*, "Toolset for SpaceWire Networks Design and Configuration," Proc. of ISC 2013, pp.149-153, 10-14 Jun. 2013.

[7] Muhammad Fayyaz *et. al.*, "Fault Tolerant SpaceWire Routing Topology and Protocol," Proc. of ISC 2010, 22-24 Jun. 2010.

[8] Christopher T. Dailey, "SpaceWire Network Packet Error Handling," Proc. of ISC 2011, pp.56-62, 8-10 Nov. 2011.

[9] Rakow G.P. *et. al.*, "SpaceWire Physical Level Redundancy Mechanism," Proc. of SMC-IT 2006, 17-21 Jul. 2006.

[10] Brice Dellandrea *et. al.*, "MOST: Modeling of SpaceWire Traffic," Proc. of ISC 2013, pp.281-285, 10-14 Jun. 2013.

[11] Artur Eganyan *et. al.*, "SpaceWire Network Simulator," Proc. of ISC 2010, 22-14 Jun. 2010.

[12] Michiya Hayama *et. al.*, "A Development of Network Evaluation Tool using SpaceWire Simulator (in Japanese)," 57th Space Sciences and Technology Conference, 9-10 Oct. 2013.

Spaceborne Unified Data&Information Network

SpaceWire Networks and Protocols, Short Paper

Wang Zhen
No.1 Dep.
Shanghai Institute of Satellite Engineering
Shanghai, China
simonlover121@163.com

Lu Guoping
Science and Technology Dep.
Shanghai Institute of Satellite Engineering
Shanghai, China
Lugp509@163.com

***Abstract*—This paper gives an application of SpaceWire Network on transferring spaceborne data & information. The on-board data is classified as payload data and platform management data. Accordingly, the on-board network is divided into payload data subnet and platform data subnet that are used to deliver payload data and platform management data respectively. Two different high-level protocol stacks are selected for the two subnets to satisfy the different transmission demands of payload data and platform management data.**

***Key Words*—SpaceWire, RVTP, time-triggered, retransmission.**

I. INTRODUCTION

As an advanced high-speed network, SpaceWire is aimed at being used as the sole on-board network in satellites[2], carrying different types of on-board data, which mainly include payload data, control data, status information of on-board equipments, clock synchronization information and so on. This paper analyzes their characteristics and transmission demands. Then it classifies on-board data as payload data and platform management data. Payload data have a high throughput and require the availability of a sustained, high bandwidth to be operational. Platform management data have a low throughput but require high reliability and have very strict time constraints. In order to satisfy the demands of payload data and platform management data, the on-board SpaceWire network is divided into two subnets in structure and function, that is high-speed payload data subnet and high reliable platform data subnet, and two different high-level protocol stacks are selected for them. Payload data are delivered by RVTP (Remote Virtual-channel Transfer Protocol)[1] in payload data subnet, while platform management data are delivered by RMAP protocol[5] in platform data subnet. Moreover, retransmission and time-triggered transmission strategy are used to provide QoS for platform management data.

II. ON-BOARD DATA CLASSIFICATION

The aim of building spaceborne unified data network using SpaceWire is to transfer the on-board data efficiently. The on-board data are all kinds of digital information exchanged between on-board devices, including payload data, control data, housekeeping data, time synchronization data and so on. According to their characteristics, these data are classified as two types: platform management data and payload data. Platform management data mainly consists of control data, housekeeping data and time synchronization data, which have a low throughput, require high reliability and have very strict time constraints. Payload data have a high throughput and require the availability of a sustained, high bandwidth to be operational.

In order to satisfy the transmission demands of payload data and platform management data, the on-board SpaceWire network is divided into two subnets in structure and function, that is high-speed payload data subnet and high reliable platform data subnet. Accordingly two different high-level protocol stacks are selected for the two different subnets.

III. THE DESIGN OF SPACE-BORNE UNIFIED DATA NETWORK

The design of spaceborne unified data network includes two parts: the network structure design and the network communication protocol design. The network structure provides hardware platform for the data transmission, while the network communication protocol defines the data encapsulation format and regulates the data transmission scheme to ensure the on-board data can be transmitted efficiently and reliably.

A. The design of network structure

The schematic diagram of the spaceborne unified data network constructed using SpaceWire is illustrated in Fig.1. The network structure consists of the SpaceWire router network and a number of SpaceWire nodes. SpaceWire nodes are the sources and destinations of data, which are all kinds of on-board devices or subsystems that are connected to the SpaceWire router network by SpaceWire interfaces. The SpaceWire router network provides a communication bridge for on-board devices or subsystems. The management unit controls the operation of the whole network, configures the network parameters and monitors the operation status of the network.

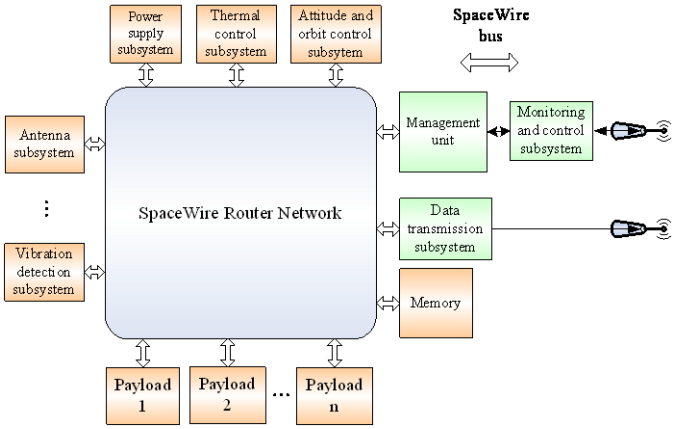

Fig. 1. The schematic diagram of the space-borne unified data network

A detailed network structure is illustrated in Fig.2. The SpaceWire router network comprises three SpaceWire routers which are interconnected by SpaceWire links. In order to increase the fault tolerance of the network, each router is dual redundant, i.e. prime and redundant. A number of SpaceWire nodes including the data management computer, processor, attitude and orbit control computer (AOCC) and the power supply controllers are connected to the SpaceWire router network by SpaceWire interfaces and SpaceWire links. The data management computer acts as the network management unit, which controls the operation of the platform management subnet, configures the network parameters and monitors the operation status of the network. The SpaceWire nodes are dual redundant as well as the routers so as to improve the reliability. The network can be extended according to practical application requirements.

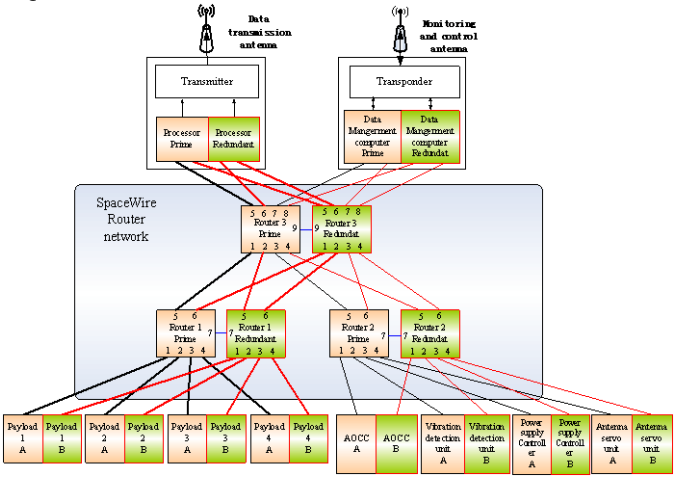

Fig. 2. The diagrammatic sketch of the space-borne unified data network

The spaceborne unified data network shown in Fig.2 is divided into payload data subnet and platform data subnet. The processor is the functional core of the payload data subnet, which is responsible for collecting all the payload data and transferring to ground through downlink after data processing. The data management computer is the functional core of the platform data subnet, which controls the transmission of status data, housekeeping data, time synchronization data and so on, and distributes the control information to the node-devices.

B. The design of network communication protocols

The on-board data transmitted in the space-borne unified data network are classified as payload data and platform management data. The former has a low throughput, very strict time constraints, and requires high reliability, while the latter has a high throughput and requires the availability of a sustained, high bandwidth to be operational. Two different high-level protocol stacks are selected for the two types of data so as to satisfy their different characteristic.

1) Payload data subnet protocol design

The payload data have a high throughput, high speed and high requirement of bandwidth. SpaceWire is a high-speed on-board network with data transmission speed ranging from 2 Mbps to 400 Mbps and the network bandwidth can be improved with the increase of the numbers of SpaceWire links and routers. The payload data subnet protocol stack is shown in Fig.3.

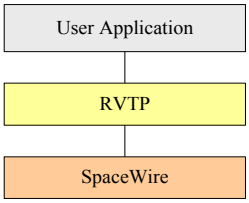

Fig. 3. Payload data subnet protocol stack

Combined SpaceWire PID and AOS data link layer, the RVTP (Remote Virtual-channel Transfer Protocol) encapsulates a CCSDS AOS Virtual Channel Frame into a SpaceWire packet[3] [4].

There are three main innovations of RVTP shown as follows:

- The RVTP is based on Virtual Channel which is firstly proposed.
- The RVTP packets are with fixed length which makes the data transfer delay predicable in a SpaceWire network.
- The RVTP provides FDIR function at the target node to facilitate fault location and recovery autonomously.

The complete format of the RVTP packet is shown in Fig.4.

First byte transmitted

	Target SpW Adress	.....	Target SpW Adress
Target Logical Adress	Protocol Identifier	Channel ID (MS)	Channel ID (LS)
VC Frame Count (MS)	VC Frame Count	VC Frame Count (LS)	Signaling Field
Frame Insert Zone (MS)	Frame Insert Zone (LS)	B_PDU Header (MS)	B_PDU Header (LS)
B_PDU Bitstream Data (First byte)	B_PDU Bitstream Data	B_PDU Bitstream Data	B_PDU Bitstream Data
B_PDU Bitstream Data	.....	.....	B_PDU Bitstream Data
B_PDU Bitstream Data	B_PDU Bitstream Data (Last byte)	EOP	

Last byte transmitted

Fig. 4. RVTP packet format

a) Target SpaceWire Address field: The Target SpaceWire Address field shall comprise zero or more data

characters forming the SpaceWire address which is used to route the RVTP packet to the target.

b) Target Logical Address field: The Target Logical Address field shall be an 8-bit field that contains a logical address of the target.

c) Protocol Identifier field: The Protocol Identifier field shall be an 8-bit field that contains the Protocol Identifier complied with the provisions of the related ECSS standards [4].

d) Channel ID field: The Channel ID shall be a 16-bit field that contains Frame Version Number, Spacecraft ID(SCID), Virtual Channel ID(VCID).

e) VC Transfer Frame Count field: The Virtual Channel Transfer Frame Count shall be a 24-bit field which contains a sequential binary count (modulo-16,777,216) of each Transfer Frame transmitted within a specific Virtual Channel.

f) Signaling field: The Signaling shall be an 8-bit field that contains Replay Flag, Virtual Channel Frame Count Cycle Use Flag, Reserved Spares, Virtual Channel Frame Count Cycle.

g) Frame Insert Zone field: The Frame Insert Zone shall be a 16-bit field that can be used to insert some special information according to user application, such as time, secret key.

h) B_PDU Header Field: The B_PDU Header shall be a 16-bit field that contains Reserved Spare and Bitsream Data Pointer.

i) B_PDU Bitstream Data Field: The B_PDU Bitstream Data Field shall be a fixed-length that follows, without gap, the B_PDU Header.

j) EOP character: The end of the RVTP packet shall be indicated by an EOP character.

2) Platform data subnet protocol design

Platform data subnet protocol stack is shown in Fig.5. Since platform data require high reliability and have very strict time constraints, some necessary classes of QoS are provided in the protocol stack.

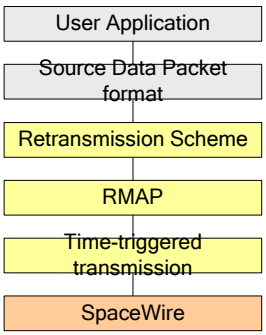

Fig. 5. Platform data subnet protocol stack

a) Retransmission Scheme

Although SpaceWire defines error detection, reporting and recovery techniques, it defines no means of recovering any data that are lost or that arrived at its destination in error. Since payload data have a high reliability requirements, retransmission scheme is necessary which provides recovery mechanisms when error occurs. If the data packet is missing, it should be retransmitted either through the same path or through a redundant path.

b) RMAP

To implement the retransmission scheme, there shall be a mechanism to let the source node know whether or not the destination node has received the data sent by the source node. RMAP is selected in the protocol stack, which has the capability for reception acknowledgement.

c) Time-triggered transmission scheme

Constructed by interconnection routers, SpaceWire network has asynchronous, multi-source characteristics. When data is transmitted in SpaceWire network, transmission delay may be uncertain due to the network obstruction. In order to make the data between network nodes interact with a reasonable and orderly manner, and to ensure the certainty of transmission delay, the spaceborne unified data&information network requires an unified transmission timing scheduling for various types of data. This time scheduling can be achieved by time triggered transport mechanism whose basic work principle is: the communication cycle of the network is divided into series of time slices, and each node transmits its data within a specified time slice to avoid or minimize data transmission conflicts.

d) The flow of the platform data subnet

the platform data generated in the source node are firstly encapsulated in the Source Packet format as defined by the user and then are encapsulated into the RMAP packet. Before transferring, the source node shall keep a copy of the RMAP packet. When the RMAP packet arrives at the destination node an acknowledgement is sent back to the source node. When the source node receives the acknowledgement which shows that the data have been transmitted correctly, it can free the buffer containing the copy. If the acknowledgement shows that some error has occurred in the received data, or no acknowledgement is received within a time-out period due to either the RMAP packet or the acknowledgement being lost, the RMAP packet can be retransmitted to recover from the error. Platform data flows from different source nodes are scheduled according to the time-triggered transmission scheme.

IV. CONCLUSION

This paper has analyzed the characteristics of on-board data and classified them as payload data and platform management data. They are delivered in different subnets by two different high-level protocol stacks so that both of their requirements can be satisfied.

REFERENCES

[1] Wang Zhen, Dong Yaohai, The Remote Virtual-Channel Transfer Protocol, unpublished

[2] ECSS-E-ST-50-12A, Space Engineering - SpaceWire - Links, nodes, routers and networks. 24 January 2003.

[3] CCSDS 732.0-B-2, AOS Space Data Link Protocol. Blue Book. July 2006.

[4] ECSS-E-ST-50-51C, Space Engineering - SpaceWire protocol identification. 5 February 2010

[5] ECSS-E-ST-50-52C, Space Engineering - SpaceWire Remote memory access protocol. 5 February 2010

SPACEMAN: A SpaceWire Network Management Tool

SpaceWire Networks and Protocols, Short Paper

Witold Hołubowicz
Adam Mickiewicz University
Poznań, Poland
holub@amu.edu.pl

Piotr Lancmański, Krzysztof Romanowski
ITTI Sp. z o.o.
Poznań, Poland
{Piotr.Lancmanski, Krzysztof.Romanowski}@itti.com.pl

Vangelis D. Kollias, Nikos Pogkas
TELETEL SA
Athens, Greece
{V.Kollias, N.Pogkas}@teletel.eu

***Abstract*—The recently developed SpaceWire Plug-and-Play protocol offers a number of possibilities in network management, including standard methods of discovery and verification. This paper presents a network management prototype tool which uses the Plug-and-Play mechanisms to produce a topology and configuration database with an XML representation of a SpaceWire network, using an XML profile that is currently under definition by ESA. That repository can then be used to produce a graphical visualization of the network. The mechanisms which make network discovery possible can also be used to perform active network management, where the network engineer modifies the network topology and node configuration beginning with the graphical network view. The tool is currently being developed in the SPACEMAN project. The architecture and implementation details are presented, as well as validation and demonstration setup.**

***Index Terms*—SpaceWire, network management, Plug-and-Play.**

I. Introduction

The increasing complexity and functionality of SpaceWire [1] networks results in the growing burden of managing them. On the other hand, there is demand for shortening development times, with a vision of missions being launched in days or weeks. This calls for a common framework, in the sense of protocols and tools, which could streamline integrating equipment from diverse vendors.

The Plug-and-Play protocol, which has been developed for several years, offers facilities useful for network management, including standard methods of network discovery and verification. Recently it has entered the draft standard phase [2]. The standardization efforts are paralleled by breadboarding, testing, and validating the protocol [3], activities indispensable for final adoption of the standard, which in turn is necessary for market to offer Plug-and-Play compliant systems.

The SPACEMAN project, launched this year with funding from ESA and involvement of ITTI and TELETEL, aims at developing a prototype network management tool based on the Plug-and-Play mechanisms, the observations and conclusions from practical implementation of the protocol being as important as the functionality of the prototype.

This paper presents the SPACEMAN network management tool as seen at an early stage of the project. The objectives and basic requirements are outlined, followed by a view of the architecture of the tool and preliminary implementation information. Finally, a setup planned for validation of the tool is discussed.

II. Objectives

With the general objective of enhancing and facilitating the process of administration of SpaceWire networks, the basic capabilities required of the tool are automatic discovery of network topology including identification of nodes, routers, and links through implementation of the network discovery protocol based on the Plug-and-Play protocol specification, as well as representation of the topology and configuration data as XML SpaceWire network profiles. Essential user interface facilities include visualization of the network topology in real time with Plug-and-Play devices connecting/disconnecting to/from the network dynamically, and graphical SpaceWire network modelling.

In particular, a number of specific functional and non-functional requirements have been identified. The functional requirements can generally be grouped according to the main tasks the tool is going to support:

- *Device and network discovery*. The tool, when connected to a SpaceWire network, has to find

information (configuration parameter values) on all the SpaceWire devices it can reach, as well as on the topology of their interconnections, whether or not there are any loops in the topology of the network. Since the Plug-and-Play protocol is of primary interest in the project, the range of information sought is determined by what can be delivered by the network management service as specified in section 5.3 of the draft protocol specification [2].

- *Device configuration*. It will be possible for the tool to set all configuration parameters specified in the draft specification as writeable. This does not mean that there cannot occur an adverse effect of such a write, e.g. if wrong routing table content is been written. It should be noted that the process of device discovery involves assignment of the device ID parameter to the device discovered, so these two categories are strongly related.
- *Monitoring*. As much as possible, the state of the network – the devices and the links – should be monitored and any changes reported. As there is no mechanism in Plug-and-Play that would automate that, this will require employing some form of polling. However, devices directly connected to the SPACEMAN tool can also be monitored by capturing their traffic entering the tool.
- *Device commanding*. The tool has to provide a possibility to send specific command packets on user request to any device it can reach.

Regarding the nonfunctional requirements, it could be noted that the tool is expected to be capable of repeating the discovery process and acquiring the values of the fields specified in the draft specification at least once per second.

III. Architecture

The current state of the network is the key object of interest in management activity, thus the central entity of the tool is the model of the network. The network is mapped on a graph, with elements of the following types:

- *Router*. This represents a device called router or routing switch or switch (the terminology expected to be subject to changes due to standardization efforts [4]). It is a vertex of the graph, with a limited number of edges (generally up to 31, following the SpaceWire standard [1]). A router can forward packets.
- *Node*. This corresponds to a device that cannot forward packets, even if it has more than one edge. It is the node rather than the end-point (a component of node, cf. [4]) that has been adopted as the real world corresponding device, since it can have a higher layer of protocols, including the Plug-and-Play network management service, and also it collects all its end-points under a single vertex. Note, however, that this is not the only possibility (cf. [4]), and that the resulting vertex of the graph is forbidden from participating in any path (unless it is the source or destination).
- *Link*. This represents a cable connection between two devices (whether routers or nodes). This is an edge in the graph. Putting aside the area of simplex SpaceWire, a connection requires signals travelling in both directions, so undirected graph has been adopted. This does not preclude assigning non-symmetrical attribute values at the level of vertices on both sides of the edge. A directed graph could alternatively be adopted, with each link represented as two edges in the graph.

The graph (whether directed or undirected) needs to permit multiple edges between vertices (e.g. if there are multiple connections between routers) as well as cycles.

Discovering the network involves searching the graph. Although this is a well-established area, observing how the discovery proceeds is of particular interest, especially if more than one network management tool is allowed to perform discovery on the same network at the same time. Therefore the view of the network model (including a slowed down animation of its being created) is the second major architectural entity.

The third one is an XML importer and exporter. Since the work on standardizing an XML network profile is ongoing, the tool will adopt a profile temporarily, which will be changed if necessary. The importer will include a validator against an XSD schema.

Finally, the most important condition for the tool to do any practical work is communication with the SpaceWire network. This requires a physical connection – SpaceWire interfaces – as well as a logical communication on the appropriate level, i.e. the level of the Plug-and-Play protocol. The two components are being provided by TELETEL in the form of the iSAFT Protocol Validation System [5,6] and Plug-and-Play API (Application Programming Interface), which provide the SPACEMAN software with physical access to the network and with Plug-and-Play protocol interface to it, respectively. The iSAFT platform is equipped with four SpaceWire ports, making it possible to access the network at four different ports or to try management by more than one instance of the tool. The iSAFT SpaceWire ports can also be used to monitor selected points in the network continuously, with packets captured and shown in real time or off-line.

IV. Implementation

The functional blocks to implement the main functions of the tool are shown in Fig. 1. The core software of SPACEMAN is being developed primarily in C++ with the intent to be portable in the sense of possible to recompile and run both on Windows and Linux platforms, and also in the sense of running either on the iSAFT platform itself, or on a separate computer, connecting to the iSAFT over the (Ethernet) network. The PnP API will run on the iSAFT platform in any case.

The iSAFT will be connected to the SpaceWire network via a 10X SpaceWire router [7].

The development work is in progress. Figure 2 shows output from the graphical renderer block – views of several models produced by the network model generator block. The model generator is dedicated to produce graph-based network

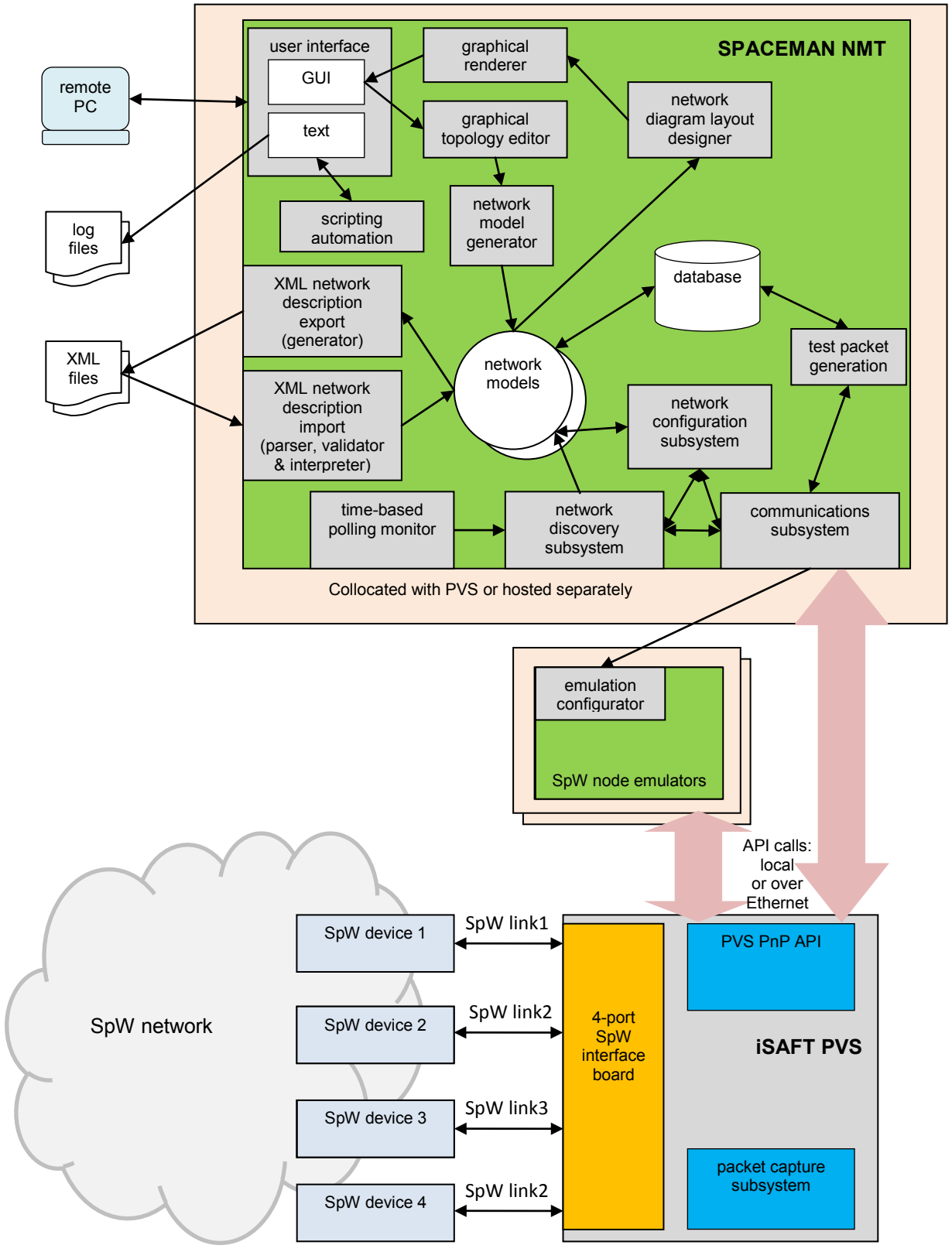

Fig. 1. Primary functional blocks of the SPACEMAN tool.

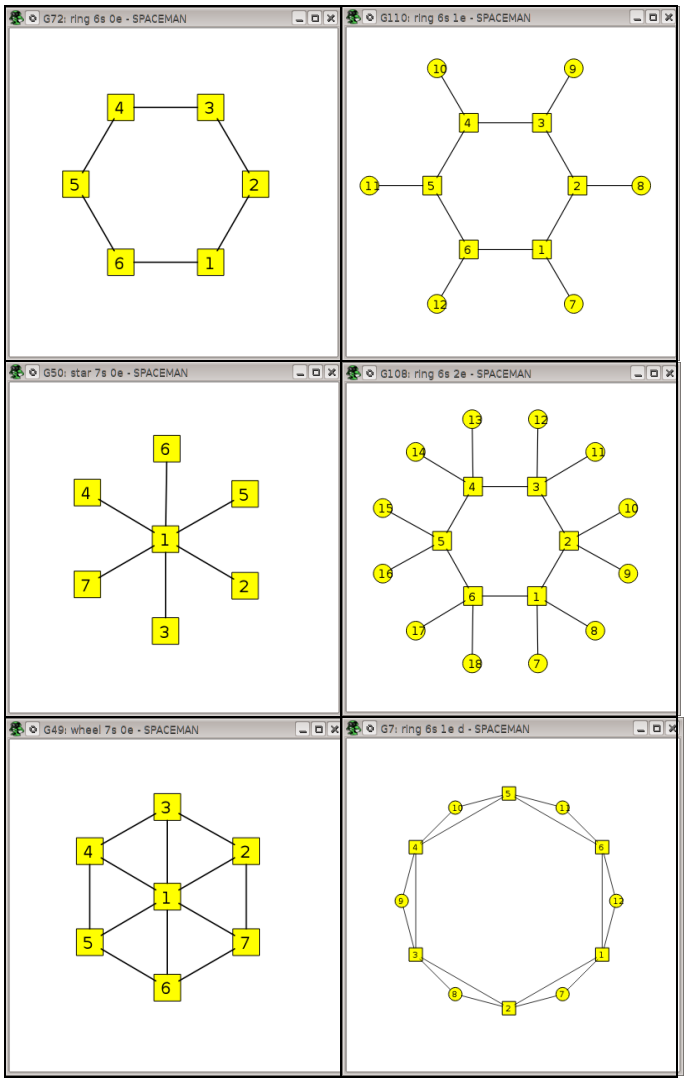

Fig. 2. Example network models generated by SPACEMAN.

models based on either interactive design by the user or – as in this case – on parameterized predefined generic models. Squares denote routers, circles – nodes, and it can be seen from the last model that a node can have more than one link.

V. Validation

Proper validation of the tool and the implementation of the Plug-and-Play protocol require connecting to a Plug-and-Play compliant network. However, at the time of writing there are no compliant routers or nodes available in the project. As a remedy, additional software elements are being developed.

Though the 10X router is not Plug-and-Play compliant, it does provide a range of configuration information sufficient to facilitate device identification and network discovery. It also does permit setting configuration, including e.g. the routing table. This functionality can be accessed via the RMAP protocol [8]. An implementation of the RMAP API for the iSAFT does exist, and the SPACEMAN tool will implement access to the router via a driver, as a device-specific layer mentioned in the Plug-and-Play draft specification.

In the absence of Plug-and-Play nodes, a node emulator will be developed, which will respond to Plug-and-Play messages as specified by the protocol. The emulator will connect via the iSAFT PnP API to one of the SpaceWire hardware ports of the iSAFT. With four physical ports, up to three instances of the emulator can be run simultaneously, emulating a network of three nodes and one eight-port router, the router being physical.

VI. Conclusion

The SPACEMAN network management tool is a prototype implementation aiming at showing the possibilities of the Plug-and-Play protocol in SpaceWire network management. This paper described the objectives, requirements, and architecture of the tool from an early stage perspective. Full demonstration of the tool is expected in the first half of 2015.

Acknowledgment

This work is being funded by the European Space Agency under contract no. 4000109438/13/NL/Cbi.

References

[1] European Cooperation for Space Standardization, “Space engineering – SpaceWire – Links, nodes, routers and networks,” ECSS-E-ST-50-12C, 31 July 2008.

[2] European Cooperation for Space Standardization, “Space engineering – SpaceWire – Plug-and-play protocol,” ECSS-E-ST-50-54C (draft), 22 March 2013.

[3] D. Jameux, “Towards SpaceWire Plug-and-Play ECSS standard,” Proc. of the 4th Int. SpaceWire Conference, San Antonio 2011, pp. 33-40.

[4] D. Jameux, A. Tavoularis, “SpaceWire standard revision,” Proc. of the 5th Int. SpaceWire Conference, Gothenburg 2013, pp. 248-256.

[5] A. Tavoularis, V. Kollias, K. Marinis, “iSAFT Protocol Validation Platform for on-board data networks,” DASIA Conference, Warsaw, 2014.

[6] http://teletel.eu/isaft-spacewire-mil-std-1553-simulator/

[7] C. McClements, S. Parkes, G. Kempf, “SpW-10X SpaceWire Router User Manual,” 30 April 2008.

[8] European Cooperation for Space Standardization, “Space engineering – SpaceWire – Remote memory access protocol,” ECSS-E-ST-50-52C, 5 February 2010.

Protocol Design for Wireless Extension of Embedded Networks: Overview of Requirements and Challenges

SpaceWire networks and protocols, Short Paper

Ekaterina Balandina, Yuriy Sheynin
St. Petersburg University of Aerospace Instrumentation
Saint-Petersburg, Russia
ekaterina.balandina@fruct.org, sheynin@aanet.ru

Yevgeni Koucheryavy, Sergey Balandin
Tampere University of Technology
Tampere, Finland
yk@cs.tut.fi, sergey.balandin@fruct.org

Abstract — **The available standards of protocols for embedded networks, e.g., SpaceWire have been designed with the consideration of wired link connections. But in practice there are many use cases and applications that would be better to address by a wireless connection. For example, it might be useful to install temperature or some other sensor at the edge of satellite's solar panel, so that the core nodes of the embedded network can regularly receive information from them. But providing access to such sensors by means of traditional wired links is a complex technological task. Moreover use of wired link in this scenario would be very expensive and not reliable.**

In this paper we make an overview of use cases that drive demand for wireless extension of the embedded networks. The paper summarizes studies on general requirements and key challenges related to deployment of the wireless extension to the embedded networks. The main focus of the study is on collecting and analyzing requirements and restrictions that affecting design of the datalink and network layers of the embedded networks' protocols. As a reference technology for this study we selected SpaceWire standard. The paper defines a scope of the corresponding problem domain and proposes ways to address the identified problems. The paper is a product of a technology exploration project, which is target to result in a list of recommendation for the wireless extension of the SpaceWire protocol and propose a set of questions plus the general scope definition of the initial tasks for a working subgroup on designing wireless protocol extension of SpaceWire.

Index Terms — **Wireless, SpaceWire, Embedded Networks, Networking, Spacecraft Electronics.**

I. Introduction

Development of the wireless telecommunication in the beginning of the 20th century transformed the corresponding technologies into the preferable mean of communications that replaced traditional wired channels in many areas. Wireless links enable mobility and usually have lower maintenance cost, plus the damaged risks, due to nature forces and human factor are much smaller. Of course wireless connection can be also damaged by natural or artificial noise and interference, but wireless networks are more dynamic, so they could be more easily reconfigured and repaired and especially important that such management could be done also distantly by use of management wireless connection.

The new boost of the wireless technologies started with exploration of space and launch of first satellites. The only possible way of communication between earth and satellites is by means of wireless technologies. There are a lot of wireless technologies that can be used and combined for wireless channels in spacecrafts. For example NASA project "Optical Communications and Sensor Demonstration (OCSD)" is focusing on the spacecraft-to-earth/earth-to-spacecraft communications and represents a usage of asymmetric communications: optical beam for transmission of large amounts of data to the Earth and radio-frequency system to receive some commands from the Earth [1].

But even though wireless communications are most natural for external communications of the satellites, we still do not see ready solutions of wireless standard for onboard systems. In our project we are targeting for a solution for small-satellite missions when it is crucial to collect and send all relevant information, while using minimum of internal resources, especially energy.

Recently we have seen rise of interest to idea of using wireless links for onboard networks on the spacecrafts. There are many forces that fuel this trend, e.g., as spacecraft onboard systems are large and very complex. Often need of installing new cable connections makes it very difficult to change end-system location and impose restrictions on the weight. This means that limited amount of space onboard and great number of cables made it impossible to reconfigure some constructions and put to alternative place some of it elements [2]. Benefits of using wireless networks in space help to overcome these imperfections but there is also number of challenges that they are going to face with. So in this paper we summarize research papers on modern trends in design of the spacecraft onboard networks.

The selected research area is rather new and there are only a few teams doing research, development and prototyping of wireless communication systems for spacecrafts. For example, already in 2009 Delft University of Technology (Netherlands) has tested fully autonomous sun sensor consisted of a sun sensor, solar battery and wireless data link [3].

There are also studies in the field of inter-satellite communications. Missions of small spacecrafts face with

challenges like power scarcity, attitude stability, limitations of total mass and volume. Question that researcher are trying to solve is selection of such communication technology that is ready to face with all limitations of the hardware and can function in the space environment and meet the requirements of both inter satellite and space-to-ground data links. As it is finalized in [4] the need is in the reduction of the communication costs through adopting common infrastructure for these purposes.

The rest of paper is organized as follows. In second section we summarize the target benefits foreseen from use of developing wireless technologies for onboard systems of the unmanned spacecrafts. The third section summarizes environmental characteristics that define the requirements that shall be met by the sensor nodes in order to have them part of wireless sensor network. The fourth section gives and overview and introduction to Integrated Modular Avionics standard. The fifth section describes currently used example of wireless sensor network for health monitoring of astronauts. After that we summarize suggestions of the consultative committee for space data systems (CCSDS). In the end of paper we provide the main conclusions and the list of used literature.

II. Benefits for Onboard System of the Spacecrafts that will be Brought by Use of Wireless Technologies

As was mentioned before, availability of wireless links will bring a lot of benefits to the onboard informational systems of the spacecraft. For example, decreasing amount of wires and reducing launch mass of the spacecraft that will lead to the following benefits:

- Decreasing cost of the spacecrafts design as according to current data the mass-to-cost ratio is more than €100,000 per kilo including launch cost. At the same time mass of cables is accounted for 20 to 35% of the total spacecraft mass [5].
- High design flexibility of wireless connections over wired and the cost of design of end-to-end wireless paths is double or triple cheaper than similar design costs for wired path [6].
- Links sustainability, which is ensured by redundancy of wired connections requires to in average double number and triple distance of cables, while for wireless network the required redundancy is for transmitters and receivers, which results in much smaller increase of weight.

Also adaptability in the sense that most of the resources that can be reused on different stages of the mission shall be designed to allow repurpose them. One more case is that wireless technologies make possible to use the multipath redundant spacecraft system because of almost free cross-strapping. As a result development of wireless network segments for onboard information system opens new horizons for development of new types of applications for spacecrafts and satellites.

III. Environmental Characteristics that Define Requirements for the Wireless Sensor Nodes

Tatiana Vladimirova and et al. describe environmental problems with which motes will face while working in the open space or onboard the spacecraft [7]. A number of environmental risks determine the operability and survivability of fragile wireless nodes:

- Mechanical (shock, vibration, acceleration): Brittle electronic elements are not suitable for applications where extreme shock, vibration, and/or acceleration exist.
- Atmospheric (corrosion, debris, vacuum): Corrosion is a big challenge for low-Earth orbit (LEO), industrial/chemical and biomedical applications.
- Thermal (extremes, limited heat transfer): Thermal extremes and cycling are sharpen in a vacuum, as thermal radiation is the only possible method for heat transfer between space and a mote.
- Energetic (radiation, including charged particles): Intensive radiation conditions are experienced in space. High-energy charged particles are the reason of single-event effects (SEEs). A total dose hardness of 5-10 Krad (SiO2) is preferable for organization of a multi-year mission in LEO. Single-event hardness is also preferable, though to define hardness levels through testing is very expensive and is usually expected and mitigated through software and hardware design redundancy.
- Dynamic (free-fall orbit, high velocity mobility, attitude disturbance torques): Orbital velocity in LEO is approximately 7.5 km/s. Natural, but undesirable perturbations change the orbit over time. This factor must be completely realized, and key parameters for example communication range can be selected properly. The freefall environment also presents unique challenges. The dominant effect is when objects in orbit “float” and change orientation or “attitude” influenced by perturbations from solar pressure, gravity gradients, magnetic fields, and aerodynamic drag. This is not a problem if the sensor technology does not have pointing requirements. However, if attitude control is required, solutions are quite difficult at this scale.

IV. Integrated Modular Avionics Standard

Wireless communications enhancement for data exchange between modules of onboard system of spacecraft is quite prominent, especially as extension proposals for Ancillary Sensor Network (ASN) and Integrated Modular Avionics (IMA). IMA is real-time computer network airborne systems. These networks consist of a number of computing modules capable of supporting numerous applications of differing criticality levels. IMA concept relies on functional isolation between operating system partitions to limit propagation of failures within avionics software. Plus it is needed to simplify software validation and verification (V&V). IMA replaces point-to-point cabling with a “virtual backplane” data

communications network. The network connects software-configurable computing modules that can adapt to changes in operating modes or respond to an avionics system fault. There is a potential path between any of these modules, with the software and network defining the active Virtual Links to support effective partitioning. In the event of failures, the system can quickly reconfigure its software functions (in pre-determined ways), resulting in a very robust system.

In continuation, Airbus Avionics Full-Duplex Ethernet (AFDX) [8] and ARINC 664 Aircraft Data Network Part 7 were defined as time-deterministic network standards. Their main goal is to maturate technology for use in commercial products to increase acceptance and adoption by the aerospace industry. Aeronautical Radio Incorporated (ARINC) [9] is a standard Real Time Operating System (RTOS) interface for partitioning computer resources in the time and space domains. ARINC 653 standard also specifies Application Program Interfaces (APIs) for abstraction of the application layer from the underlying hardware and software service. It allows to the host to have several applications of different software levels on the same hardware in the context of Integrated Modular Avionics architecture. One of the services provided by ARNIC is Satellite Navigation and Air Traffic Control and Landing Systems (SATNAV and ATCALS). There is a formal mapping of ARINC 429 to AFDX and a similar mapping can be done for MIL-STD 1553B and similar protocols [10]. This makes it easier to port legacy software code to AFDX environments and help to benefit flight-certification efforts. Any AFDX end-station port interface can source several virtual links (VLs) and can at the same time receive other VLs, if it is required by the application. A lot of physical cables needed for ARINC 429 are replaced by a single network cable plant containing the same circuits and implemented in time-division multiplexed VLs. This helps to save weight, volume, and wiring complexity. In real-time control systems freshness of information much more important than integrity that is why errors are rejected rather than corrected.

V. Examples of Use Cases: Wireless Sensor Network for Health Monitoring on Piloted Spacecrafts

The Intelligent Systems Division at NASA Ames Research Center has been developing WSN technology for useing aboard spacecraft for Integrated System Health (ISHM) monitoring of structures funded by the NASA Engineering and Safety Center and Exploration Technology Development and Demonstration Program. Mesh-enabled WSNs provide appropriate failure tolerance and SPA provides dynamic fault management responsible for low-power, low-cost ancillary sensing solutions for spacecraft. Proposed in [11] architecture and technical opportunity of creating wireless failure-tolerant sensor networks is based on integration of Zigbee and SPA technology together into SPA-Z architecture. Zigbee provides effective management of WSNs using its own proprietary internal methods.

Monitoring health parameters and support of life systems is discussed in [12, 7] as one example of usage of small devices with resource limitations and wirelessly connected. Authors of [7] called them mote which is an abbreviation for 'remote' node and refers to the individual units of sensing in a wireless sensor network. Three types of commercial sensors were tested: TelosB motes from Crossbow [13], BTNode from Art of Technology [14] and High Powered Modules (HPM) from Jennic [15]. Each maintenance or out of work device or data channel can lead to the unchangeable consequences and ending of the mission. But as space industry is very expansive it is quite crucial to avoid such situations. That is why in the next paper we will focus particularly on restrictions that are important to take into account while adopting wireless technologies to the spacecraft environment: power level of transmission, jamming of wireless signal, and physical location. These factors are even more important assuming system work in the exploding environment, RF exposure levels in excess of governmental limits, and electromagnetic compatibility [16].

Also authors of [7] are concerned with the application of standard wireless protocols for communication inside the satellite (intra-satellite communication and data gathering) and communication between satellites (inter-satellite communication). Despite the fact that standard wireless commercial off-the-shelf (COTS) protocols are widely used terrestrially they are not so popular in the space application domain. Also authors of [7] stated that there is the possibility of using ZigBee-Pro systems for maximum 1.6 Km range at 250 Kbps for sensor networking in inter-satellite applications could also be considered. There are several sources of radiation onboard the spacecraft: natural sources and radiation from electronic devices of the spacecraft. Internal frequencies and bandwidths of on-board equipment vary from a few hundred Hz to several GHz. Electromagnetic interference can lead to failures of the electronic devices on board spacecraft or even permanent damage that have to be maintained. If sensor nodes don't have shielding it makes them very sensitive to EMI.

Two communication standards—ARINC 429 and MIL-STD 1553B—have dominated in commercial and military aviation [9]. The MIL-STD 1553B is used mostly in military aircraft for flight-critical control and control of various mission systems. Both are half-duplex communication standards. ARINC 429 connects LRUs via a point-to-point cabling scheme; MIL-STD 1553B connects multiple devices via a common bus. These standards are currently used in production aircraft and spacecraft. There are some deficiencies in performance of these standards which has led to development and adoption of extended and modified versions suitable for modern aircrafts [9].

VI. Suggestions of the Consultative Committee for Space Data Systems (CCSDS)

The Consultative Committee for Space Data Systems (CCSDS) is an international platform for development of communication standards for spaceflights. Currently CCSDS unites scientists from 26 countries whose goal is to combine interoperability enhancement, development costs and risks reduction. CCSDS has several different types of regular publications which are categorized into several groups

according to the following colors: Blue - Recommended Standards; Magenta - Recommended Practices; Green - Informational Reports; Orange – Experimental; Yellow – Record; Silver - Historical.

Recently the committee issued the Magenta book [16] with recommendations on design of the low-level protocols for wireless networks in the monitoring and control systems onboard of the spacecrafts. The main goal is to make possible for various sensors (produced by different vendors and with different high level application on top of it) to enter the star topology network and connect to the gateway. This book describes two approaches: single-hop contention-based access and single-hop scheduled access. Both approaches can be applied to the star-topology network. However, peer-to-peer exchange of data scenario in mesh networks is not in the scope of this article. One more consideration is that in the book it is assumed that the gateway of the PAN is able to communicate with the backbone network. The book doesn't describe this functionality as it is usually implemented on the network layer of the OSI which is out of the scope of the document. For the same reason acknowledgement and retransmission functionality are not mentioned in the book. Authors describe CSMA-CA and TDMA as two possible medium access schemes both have prons and cons but accent is made on TDMA usage as interference avoidance schemes such as frequency hopping are much more easily implemented in a TDMA. Maintaining connectivity in a mesh network topology is also easily implemented in TDMA (as it supports multi-hop relay traffic with battery powered nodes on a low duty cycle (long sleep period, short active period). Standards which it is recommended to follow are IEEE 802.15.4-2011 for single-hop contention-based communications and ISA100.11a-2011 for single-hop scheduled medium access communications. Recommendations that are listed in the book include restrictions on wireless technology which include risks associated with the selected radio frequency band, transmission power level, and physical location. There are some factors that should be taken into account:

1) Operation in explosive environments;
2) RF exposure levels in excess of governmental limits;
3) Electromagnetic Compatibility (EMC).

CONCLUSION

The main goal of current pre-phase of our project was to study and prepare an overview of research and development activities on wireless sensor networks for spacecrafts. As a result we identified the key players in the field and mayor projects and standards that have been developed and are under development. In particular we collected and analyzed requirements and restrictions that affect design of the datalink and network layers of the embedded networks' protocols for the SpaceWire protocol stack. Also we prepared own list of recommendations for further development and identified partners for cooperation on further project development. The work will be continued and progress reported in the next paper.

ACKNOWLEDGMENT

The paper is done as a kick-off of field studies of Ekaterina Balandina in the dual degree doctoral program of Tampere University of Technology and Saint-Petersburg University of Aerospace Instrumentations.

REFERENCES

[1] "Optical Communications and Sensor Demonstration (OCSD)" May 2013, Available On-line [http://www.nasa.gov/directorates/spacetech/small_spacecraft/ocsd_project.html#.U5h6kXLV_VF].

[2] http://cenic2011.cenic.org/program/slides/cenic-2011-zigbee-sensor-net_foster.pdf

[3] Will H Zheng, John T Armstrong, "Wireless Intra-Spacecraft Communication: the Benefits and the Challenges", 2010 NASA/ESA Conference on Adaptive Hardware and Systems.

[4] Richard Alena, Yosuke Nakamura, Nicolas Faber, David Mauro, "Heterogeneous Spacecraft Networks: Wireless Network Technology Assessment", IEEE Aerospace Conference, 2014.

[5] David Jameux, Christian Fraboul, "Introduction to unmanned spacecraft on-board communications: evolution of timeliness needs", Proceedings of the 2nd International Workshop on Worst-Case Traversal Time, 2012.

[6] "Flight-tested technologies", July 2009, Available On-line [http://www.esa.int/Our_Activities/Technology/Flight-tested_technologies].

[7] Tanya Vladimirova, Christopher P. Bridges, George Prassinos, Xiaofeng Wu, Kawsu Sidibeh, David J. Barnhart, Abdul-Halim Jallad, Jean R. Paul, Vaios Lappas, Adam Baker, Kevin Maynard and Rodger Magness, "Characterising Wireless Sensor Motes for Space Applications", Second NASA/ESA Conference on Adaptive Hardware and Systems(AHS 2007), 2007.

[8] Shabaz I Kazi, "Architecting of Avionics Full Duplex Ethernet (AFDX) Aerospace Communication Network", InternatIonal Journal of ElectronIcs & CommunIcatIon Technology, Vol. 4, Issue 4, pp 73-77, 2013.

[9] Richard Alena, John Ossenfort IV, Kenneth Laws, Andre Goforth, Fernando Figueroa, "Communications for Integrated Modular Avionics", IEEE Aerospace Conference, 2007.

[10] ARINC 664 Part 7 Standard, "Aircraft Data Network Part 7. "Avionics Full Duplex Switched Ethernet (AFDX) Network," Appendix B, 2005.

[11] Richard Alena, John Ossenfort, Thom Stone, and Jarren Baldwin, "Wireless Space Plug-and-Play Architecture (SPA-Z)", IEEE Aerospace Conference, 2014.

[12] Richard Alena, Fernando Figueroa, John Ossenfort "Intelligent Wireless Sensor Networks for Spacecraft Health Monitoring", 2012.

[13] Crossbow Technology Inc., http://www.xbow.com/Products BTNodes - A Distributed Environment for Prototyping.

[14] Ad Hoc Networks, http://www.btnode.ethz.ch

[15] Jennic Homepage, http://www.jennic.com

[16] Magenta Book: The Consultative Committee for Space Data Systems, Recommendation for Space Data System Practices, Spacecraft Onboard Interface System — Low Data-rate Wireless Communications for Spacecraft Monitoring and Control, May 2013, Available On-line [http://public.ccsds.org/publications/archive/882x0m1.pdf]

Components (Long)

GR718 – Radiation-Tolerant 18x SpaceWire Router Based on the DARE 180 nm Library

SpaceWire Components, Long Paper

Jonas Ekergarn, Sandi Habinc, Fredrik Ringhage, Fredrik Sturesson, Martin Simlastik
Aeroflex Gaisler AB
Kungsgatan 12, SE-411 19 Gothenburg, Sweden
info@gaisler.com

IMEC
Kapeldreef 75, 3001 Leuven, Belgium
info@imec.be

Martin Suess
European Space Agency
Keplerlaan 1, 2220AG Noordwijk ZH, The Netherlands
martin.suess@esa.int

Steven Redant, Kurt Stinkens, Geert Thys, Jagadeesa Das Arul Mahesh

***Abstract*— GR718 is a radiation tolerant 18 port standalone SpaceWire router component that has been developed by Aeroflex Gaisler together with IMEC (BE), in an activity initiated by the European Space Agency under ESTEC contract 4000105402/12/NL/Cbi.No. Out of the 18 SpaceWire ports, 16 use on-chip LVDS transceivers, and two use LVTTL signaling. Included also is the mandatory configuration port, as well as an internal port for system level testing. All ports are capable of operating in 200 Mbit/s. UART and JTAG interfaces, that gives access to the on-chip AMBA AHB bus, are provided for configuration and debugging. SPI and GPIO interfaces are accessible through the configuration port, which allows SPI devices to be accessed and general purpose signaling to be performed through RMAP commands. In addition to the mandatory features in the current ECSS SpaceWire standard, GR718 supports group adaptive routing for path addresses, and packet distribution. It also includes support for the incoming SpaceWire standard revision 1 (ECSS-E-ST-50-12C Rev.1), the SpaceWire-D protocol, and the SpaceWire Plug-and-Play protocol currently being developed for ECSS. The technology used is UMC 180 nm, using the DARE library from IMEC, and the package is a 256 pin CQFP. A development board for evaluation and software development has been manufactured as well.**

***Index Terms*—SpaceWire, Networking, Spacecraft Electronics**

I. Introduction

Both ESA and several companies in the space industry have indicated 16 as the most viable number of SpaceWire ports for routers in the near future. Aeroflex Gaisler's intentions with the GR718 development was to provide this key component with a new 18 port SpaceWire router ASIC.

The design is based on the GRSPWROUTER configurable SpaceWire router IP core. The IP core supports from 2 to 31 ports of three different types: SpaceWire, AMBA and FIFO. The SpaceWire ports implements an encoder-decoder compliant to ECSS-E-ST-50-12C [1] and provides an external SpaceWire interface. FIFO ports provide 9-bit parallel interfaces with control signals in each direction (read/write), which can be used to interface external units or to cascade two or more routers without any glue logic. The AMBA ports interface to an AMBA AHB bus using DMA on the bus. All three port types connect to the switch matrix of the IP core using identical FIFO based interfaces. There is no way to distinguish the three ports on the SpaceWire packet level and upwards. The configurability provided by the IP core makes it usable in many different applications. It has already been used in several standard rad-hard components on Actel RTAX2000SL and RTProASIC3 FPGAs, and is also used in the Next Generation Micro Processor (NGMP) system-on-chip activity funded by the European Space Agency.

During the development phase, two configurations of the IP core were identified as potential candidates for the final ASIC: one with 16 SpaceWire ports with on-chip LVDS transceivers, and two additional ports, either SpaceWire LVTTL ports or FIFO ports; and the other with 16 SpaceWire ports and two internal AMBA ports connected to a PCI interface. Both configurations were evaluated in detail to determine which one would eventually be used for manufacturing.

The final choice fell on the configuration with 16 LVDS SpaceWire ports and two LVTTL SpaceWire ports, where the only difference between the two different SpaceWire port types is the I/O type of the pads.

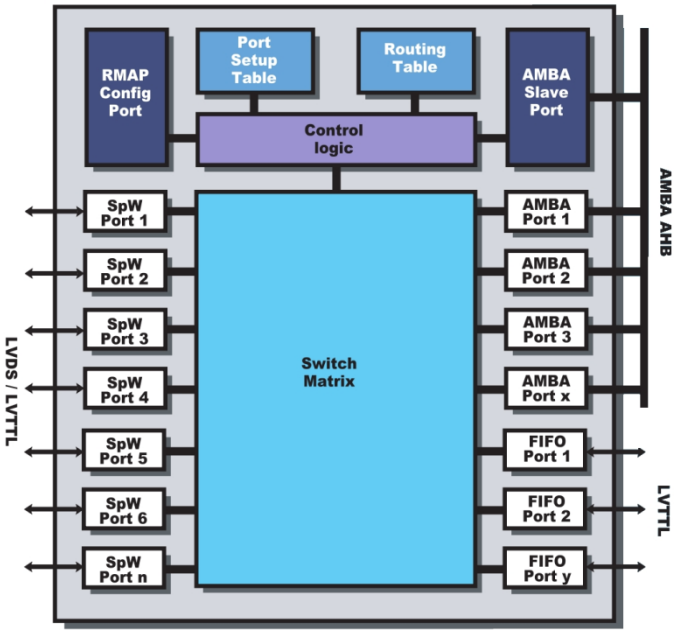

Fig. 1. GRSPWROUTER IP core overview

The choice to include two additional SpaceWire ports instead of two FIFO ports was motivated by the pin count of the selected package, as well as the fact that more and more processor devices have built-in SpaceWire ports (with LVTTL signaling), and therefore parallel FIFO ports would not be readily used without the need for an FPGA device between the router and the processor. It is also not that difficult to include SpaceWire links in FPGAs, considering the large variety of SpaceWire IP cores available.

One of the applications of the GRSPWROUTER IP core's FIFO ports is to cascade one or more routers without any glue logic. However, the SpaceWire ports will work equally good for this purpose. In most cases cascading would be done on a printed circuit board, and it is well understood how to route SpaceWire signals on such a board. The FIFO interfaces are most useful when connecting directly to external processors and memories. To use a SpaceWire link instead will require the insertion of glue-logic providing a complete SpaceWire codec, which would typically be done using a FPGA, which increases design complexity considerably. It is however anticipated that the need to interface to external processors using parallel interfaces will decrease in the future since most processors will be equipped with SpaceWire interfaces.

Other considerations that were taken into account during the design phase were such as whether or not to include support for the incoming revision 1 of the SpaceWire standard (ECSS-E-ST-50-12C Rev. 1), and the new SpaceWire-D and SpaceWire Plug-and-Play protocols. The problem has been the lack of a firm schedule for finalization of these standards. In fact, none of the standards were completed at the time of tape-out. However, Aeroflex Gaisler is actively involved in the work of finalizing the revision 1 of the SpaceWire standard, and has also been reviewing and discussing the two other protocols with the developers. In this way the risk of implementing something that will later change in the protocols have been mitigated.

II. GR718 Functional Overview

The full GR718 architecture includes the following modules: SpaceWire Router, SPI Controller, UART Interface, JTAG Interface, General Purpose I/O Interface, SpaceWire In-System Test (SIST), System Level Test Configuration, AMBA AHB controller and AMBA APB controller.

The SpaceWire router implements a SpaceWire routing switch as defined in ECSS-E-ST-50-12C. Among the features supported by the router are: group adaptive routing, packet distribution, system time-distribution, distributed interrupts, port timers to recover from deadlock situations, and SpaceWire-D packet truncation based time-slot violations.

A total of 20 ports is provided, where port 0 is the mandatory configuration port, ports 1-18 are SpaceWire ports, and port 19 is a custom port called the SIST port. Each SpaceWire port contains a SpaceWire codec, and provides an external SpaceWire interface. The SIST port provides a FIFO interface which is internally connected to a SpaceWire In-System Test module (described later). The configuration port provides a target for the Remote Memory Access Protocol (RMAP) defined by ECSS-E-ST-50-52C [2], and an AMBA AHB slave interface, both used for accessing internal configuration and status registers. The configuration port also provides a SpaceWire Plug-and-Play interface, allowing device identification. The ports allowed for configuration accesses can be restricted if needed, using several configuration options.

For diagnostic and test purposes, UART and JTAG interfaces are provided. These low pin count interfaces are suitable in the small package but at the same time have sufficient bandwidth. Both the UART and JTAG interfaces act as masters on the internal AMBA AHB bus and gives access to the complete set of registers.

The SPI and General purpose I/O interfaces are accessible through the router's configuration port, which allows SPI devices to be accessed, and general purpose signaling to be performed directly through RMAP commands, or through the UART and JTAG interfaces.

An auxiliary time- / interrupt-code interface is present, for sending and receiving time- / interrupt-codes through external pins. Parts of the interface use dedicated pins, while the rest are multiplexed on the general purpose I/O pins.

III. Packet Routing Features

The router's switch matrix can connect any input port to any output port. Access to each output port is arbitrated using a round-robin arbitration scheme based on the address of the incoming packet. A single routing-table is used for the whole router, where access to the table is arbitrated using a round-robin scheme based on the input port number. Both addresses and input port can be assigned either high or low priority.

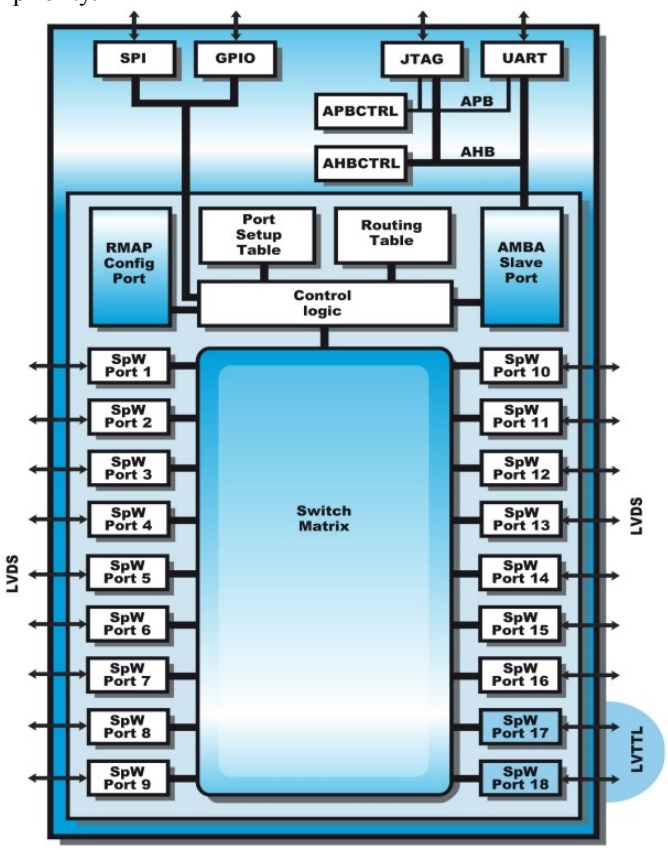

Fig. 2. GR718 overview

All the addressing modes, such as path, logical, and regional logical addressing are supported. Group adaptive routing is fully supported, meaning that both path and logical addresses can be individually configured to use one or more output ports. A unique feature is the support for packet distribution, which can be used to implement multicast and broadcast addressing. Also packet distribution can be enabled for any address.

Each router port is equipped with a timer which can be individually enabled/disabled. The timer can be used to recover from potential deadlock situations resulting from either a stalling source node or stalling destination node.

IV. SpaceWire Standard Revision 1 Support

An upcoming revision 1 of the SpaceWire standard is planned for the near future. The new revision will contain some changes that affected the GR718 development. Some of the new additions may result in some old devices not being forward compatible. The final details of the updates have not been decided yet and there is no date set for when this will be ready, so there was a considerable risk in implementing these new features before the standard was finalized.

Three changes were identified as having a technical impact on the GR718 development. The first one is the addition of timers in routers. The GRSPWROUTER IP core already contained programmable packet timers for each port, which meant that no changes were required. However, an addition to the functionality was made in order to be able to distinguish between overrun and underrun timeouts.

The second change is a modification of the link interface FSM. Two requirements have been identified that potentially can cause the SpaceWire codec to make unwanted transitions. These are unlikely corner cases and very few if any problems have been seen in practice. This modification will probably not affect backward compatibility with older SpaceWire codecs, so the risk of including this modification in GR718 was estimated to be very low.

The final and most complicated change is the addition of distributed interrupts. The distributed interrupt scheme introduces two new control codes, called interrupt-code and interrupt-acknowledge-code, which uses one of the reserved control bit combinations of Time-Codes. It must therefore be made sure that they cannot interfere with the normal Time-Code facilities. All existing devices might not be forward compatible with revision 1 compliant devices due to the interrupt- / interrupt-acknowledge-codes.

The distributed interrupt scheme was identified as the part of revision 1 that caused the highest implementation risk if included in GR718. Therefore the router was made flexible enough to allow ports' handling of the new control codes to be configured individually. In this way the router can be used as a device that enables old and new equipment to be used in the same SpaceWire network.

The distributed interrupt scheme is defined by [3], and GR718 supports all the requirements put on routers, as well as some optional features to minimize the effects of errors such as a babbling idiot. Due to the uncertainty regarding some details in the specification, GR718 was given a high degree of configurability how to handle the distribution of interrupt- / interrupt-acknowledge-codes.

V. SpaceWire-D Support

There is a new protocol emerging called SpaceWire-D, where D stands for deterministic [4]. This is anticipated to be widely used in the future to provide deterministic and low-latency transfer of control and command information while still preserving the high bandwidth of SpaceWire. It basically consists of a time-slotting table replicated in each unit (node or router) in the SpaceWire network. Therefore a router

needs to have support for SpaceWire-D if it is used in a network utilizing that protocol.

GR718 implements support for SpaceWire-D by monitoring packet transfers. In the case of a packet being transferred while a Time-Code is received, the packet is truncated and an EEP is inserted at the end of the packet. The truncation can be individually enabled/disabled per port, and there is a programmable Time-Code filter per port as well. The filter allows for each port to have different Time-Code values or ranges that truncates packets. The programmable filters also allows distributed interrupt-codes to truncate packets.

GR718 implements status bits that inform software if a packet has been truncated due to a received Time-Code. There is also an option to automatically send an interrupt-code when the truncation occur.

VI. SpaceWire Plug-and-Play support

SpaceWire Plug-and-Play is an upcoming standard that allows SpaceWire routers and nodes in a network to be identified and configured, and is defined by [5]. The standard uses RMAP commands and replies for communication, but with a different protocol ID.

GR718 includes basic support for SpaceWire Plug-and-Play, which covers device identification and support for network discovery. Extended capabilities, such as routing table configuration, and port configuration through SpaceWire Plug-and-Play, was not included due to the fact that the standard was not considered mature enough at the time of implementation. The SpaceWire Plug-and-Play functionality can be disabled by means of a configuration pin.

VII. SpaceWire In-System Test

A built-in self-test is provided for the verification of the SpaceWire router and codec functionality. The SpaceWire In-System Test (SIST) protocol provides a means for verifying larger part of the designs' functionality without the need to generate high speed test patterns and observe results at high frequencies.

The internal SIST module is connected to the router via a dedicated FIFO port. The external side of the SIST module is connected to the AMBA APB bus, which is only accessible through the JTAG and UART (debug-) interfaces. Thus it is not possible to configure the SIST module via a SpaceWire link.

The SIST module can generate and send SpaceWire packets via the internal FIFO port. It can also receive SpaceWire packets via the FIFO port and check there contents. The packets are generated deterministically and can therefore also be easily checked on reception.

The packet format is similar to the commands defined for the RMAP protocol (ECSS-E-ST-50-52C):

- SpW Address (0 to 31 bytes)
- Logical Address (1 byte)
- Protocol ID (1 byte)
- Transaction Identifier (2 bytes) (i.e. seed)
- Data Length (3 bytes)
- Header CRC (1 byte as per ECSS-E-ST-50-52C, covering header from Logical Address, inclusive)
- Data (0 to 16 MiB-1) (data is a pseudo-random generated bit string based on the seed)
- Data CRC (1 byte as per ECSS-E-ST-50-52C, covering all Data bytes)
- End-Of-Packet

Packets of up to 2^{24} bytes can be generated and checked. Sequences of up to 2^{16} packets can be generated, or auto repeat can be enabled. The data is generated by means of a 16-bit wide LFSR, with a programmable polynomial. The stated of the LFSR (a.k.a. seed) at the beginning of the data in the packet is transmitted as part of the packet header, allowing each packet to be checked independently. The seed can also be used to detect dropped packets. The length of the packet data field is sent in the packet header. The only managed parameter is the polynomial; everything else can be derived from the packet header.

Packets are automatically generated in an initiator, the contents of a packet is deterministic. Packets are automatically checked in a target when received, providing statistics. The initiator and target are normally the same end-point in a SpaceWire network, but may be different.

The SIST module also allows direct data read and write to the internal FIFO port, as well as sending and receiving signaling codes (time-codes and distributed interrupts).

The packet follows the format defined by SpaceWire protocol identification – ECSS‑E‑ST‑50‑51C [6] format. The address bytes can be used for path addressing or regional logical addressing in a SpaceWire network.

The SIST functionality is protected by means of a protected general on/off register (protection done by expected fixed pattern in data). It is not accessible through SpaceWire RMAP or SpaceWire PnP accesses to configuration port 0. The SIST module can also be clock-gated to save power (default at reset) via JTAG and UART interfaces.

VIII. Power-saving Features

GR718 incorporates the following power saving functions:

- Disabling of unused on-chip LVDS receivers/transmitter
- Disabling of unused off-chip LVDS receivers/transmitter or repeater devices
- Clock-gating of unused SpaceWire ports

The existing power-down functionality provided for the LVDS I/O cells in the DARE+ library is being utilized.

Signals for disabling the off-chip LVDS devices are shared with the external pins provided for general purpose I/O. It is possible to control up to 18 external LVDS devices, with one external pin per devices.

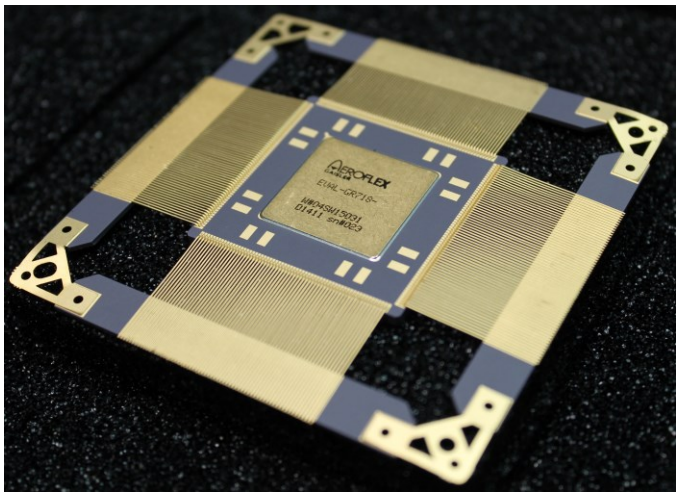

Fig. 3. GR718 device

IX. Results

GR718 prototype devices has been manufactured in 180nm UMC CMOS technology, based on the DARE+ library from IMEC (BE). The technology is radiation hard, with at least 300 krad(Si) TID tolerance, high SEL tolerance and SEU hardened flip-flops. The package used is a custom 256 CGFP.

The target speed for the SpaceWire links was 200 Mbit/s. However, during functional testing and validation the devices has been found to operate successfully at 240 Mbit/s.

The GR718 device uses 1.8V and 3.3V supply, and the typical power consumption is 3W when running all 18 SpaceWire ports in 200 Mbit/s.

A development board has been developed together with Pender Electronic Design. The board comprises a custom designed PCB in a 6U Compact PCI format, making the board suitable for stand-alone bench top development, or if required, to be mounted in a 6U CPCI Rack. The purpose of this equipment is to provide developers with a convenient hardware platform for the evaluation and development of software for the GR718. The principle interfaces and functions are accessible on the front and back edges of the board, and secondary interfaces via headers on the board.

X. Conclusion

The overall activity has resulted in tape-out of an advanced multi-port SpaceWire router ASIC, and the manufacturing of a development board, all now available for national and international space industry.

During the GR718 development Aeroflex Gaisler has been participating and contributing in the ongoing standardization work of the distributed interrupt scheme that will be part of the SpaceWire standard revision 1, as well as the upcoming SpaceWire Plug-and-Play standard. These extra efforts are expected to pay off with an advanced multi port SpaceWire router ASIC which enables coexisting of older and newer equipment in the same network.

An additional unplanned task performed was the development of a new custom package, needed in order to improve electrical characteristics and support higher clock frequencies.

References

[1] ECSS - Space Engineering, SpaceWire - Links, nodes, routers and networks,
ECSS-E-ST-50-12C, July 2008

[2] ECSS - Space Engineering, SpaceWire - Remote memory access protocol, ECSS-E-
ST-50-52C, February 2010

[3] Yuriy Sheynin, Distributed Interrupts in SpaceWire Interconnections, International SpaceWire Conference, Nara, November 2008 (outdated)

[4] SpaceWire-D - Deterministic Control and Data Delivery over SpaceWire Networks, Draft B, April 2010, ESA Contract Number 220774-07-NL/LvH

[5] ECSS - Space Engineering, SpaceWire Plug-and-Play protocol, ECSS-E-ST-50-54C
Draft, March 2013

[6] ECSS - Space Engineering, SpaceWire protocol identification, ECSS-E-ST-50-51C,
February 2010

Galvanically Isolated SpaceWire

SpaceWire Components, Long Paper

Michael Epperly, Steven Torno
Space Systems and Engineering
Southwest Research Institute
San Antonio, TX, USA
mepperly@swri.edu, storno@swri.edu

***Abstract*— The physical signaling layer for SpaceWire is specified in ECSS-E-ST-50-12C as LVDS per ANSI/TIA/EIA-644. LVDS is a current mode, differential interface that offers several benefits over other types of physical interfaces particularly in the areas of noise rejection, electromagnetic noise generation, and power supply decoupling. While LVDS offers a marginal immunity to common mode voltage differences of about 1.2V, systems with physical separation, separate power supplies or a non-common ground reference can exceed this limit. Galvanic isolation, using transformer coupling similar to that implemented by standard Ethernet interfaces, can AC couple the interface and reduce the sensitivity to common mode imparting an increased level of reliability to the overall system. Unfortunately, standard SpaceWire does not readily support such isolation. Ideally, an AC coupled interface has a near 50% bit transition density to keep the reference near the midpoint of the voltage span. The encoding of data and the clock recovery scheme of SpaceWire data on a transformer coupled physical layer presents a reference offset problem since its data is neither randomized nor encoded use a leveling code such as 8B/10B. Worse yet, with SpaceWire's clock recovery encoding, a high transition density on the data line can result in a subsequent low transition density on the strobe line. This paper describes the development and execution of a galvanically isolated SpaceWire network using a transformer isolation approach. The problems encountered, analysis performed, and techniques used to implement a reliable galvanically isolated SpaceWire network are presented and discussed.**

***Index Terms*—SpaceWire, Galvanic isolation, Randomization, Bit-Transition Density**

I. Introduction

This paper discusses the development of a Galvanically isolated SpaceWire interface operating at up to 133 Megabits per second (Mbps), and the pitfalls and difficulties encountered in that development.

Galvanic isolation can provide noise immunity and tolerance of common mode differences between interfaces. As such it is desirable in high reliability applications if it can be implemented without adding excessive overhead and complexity.

To understand the issues involved with creating a reliable transformer coupled SpaceWire link we first present a basic discussion of SpaceWire.

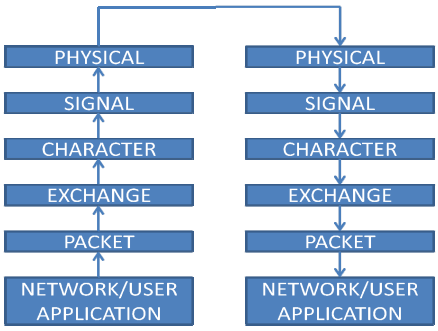

Fig. 1. SpaceWire Layered Protocols

Like all standard interfaces, the SpaceWire interface can be broken down by layers. The definition begins with the physical layer which includes connectors and cables. The next higher level, the signal layer, defines the electrical interface and data/clock encoding. Above the signal level is the character level where control and data characters are defined. The next level is the exchange level where the protocol for establishing and controlling the link is defined. The Packet level defines how the user data or "Cargo" is encapsulated before it is handed off to the next lower level. A graphical representation of these layers is provided in Fig. 1.

The electrical interface physical layer is defined for SpaceWire is defined as Low Voltage Data Signalling (LVDS) per ANSI/TIA/EIA-644.

The physical interface coding creates a pair of LVDS signals described as Data and Strobe. The Data line is a non return to zero (NRZ) representation of the data and the strobe toggles its state whenever Data does not change from one bit time to the next (consecutive 1s or 0s). See Fig. 2.. The

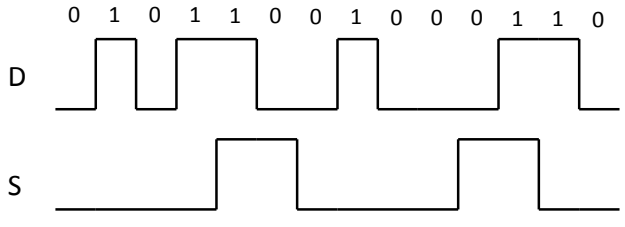

Fig. 2. SpaceWire Data and Strobe

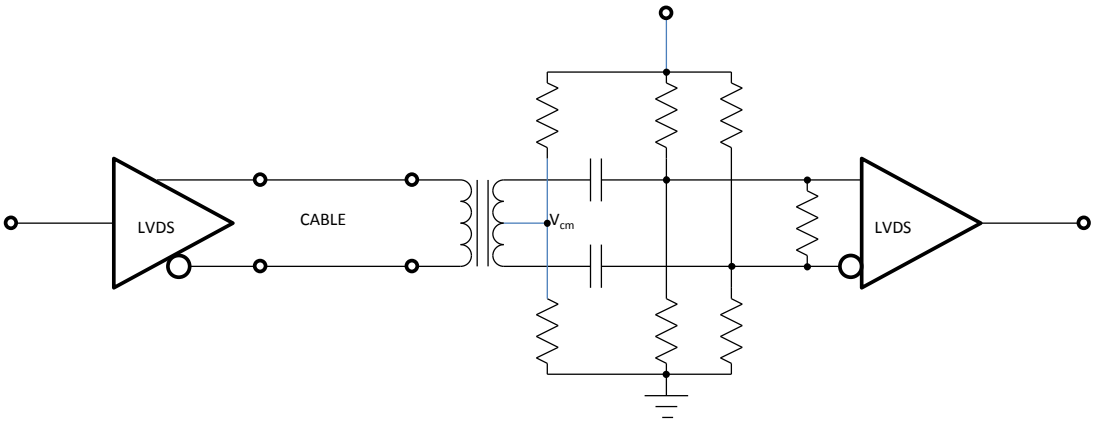

Fig. 3. Galvanic Isolation Schematic

minimum signalling rate of Spacewire is 2Mbps and a maximum is not specified. In this paper, the maximum signalling rate was limited by the performance of the hardware to 133 Mbps.

The Character level of the protocol defines a 10-bit data character set and a variable length control character set. The first bit of either is a parity bit and the second bit is set (1) for a control word and clear (0) for a data word. There is no other encoding of the data and the strobe is created from the data as described earlier.

The Exchange level defines the protocol sequence of control characters to establish and maintain the link and is not pertinent to this paper.

The Packet level provides a protocol for sending a packet between entities on the SpaceWire Network. A packet consists of a destination address, user data (cargo) and an EOP (End of Packet) control character. It is within the user data that this paper is primarily concerned.

The primary problem with galvanically isolating an interface is that there isn't a solid Direct Current (DC) reference point for the differential signals to use and the circuit performance is affected by the actual data transmitted. A long sequence of ones or zeros will cause the reference to be pulled one direction or another and lead to bit errors. With a single data stream, such a bias can be controlled via randomization or other encoding techniques. Unfortunately, the nature of SpaceWire's Character level protocol coupled with the Data and Strobe encoding of the Physical level makes this a more challenging endeavor.

We used a circuit recommended by Aeroflex for our galvanic isolation circuit and tailored component selection based on our laboratory results.

Beyond the actual galvanic isolation circuit, we did not want to perturb the SpaceWire standard any more than necessary and set down the rule that any additional coding would occur within the cargo and all other protocols would remain the same. This effectively made the encoding a software function only and meant that we could easily update our algorithms to improve performance.

II. Goals and Constraints

There were several goals for this development effort, the primary being the realization of a reliable and robust galvanically isolated SpaceWire link. This meant that the

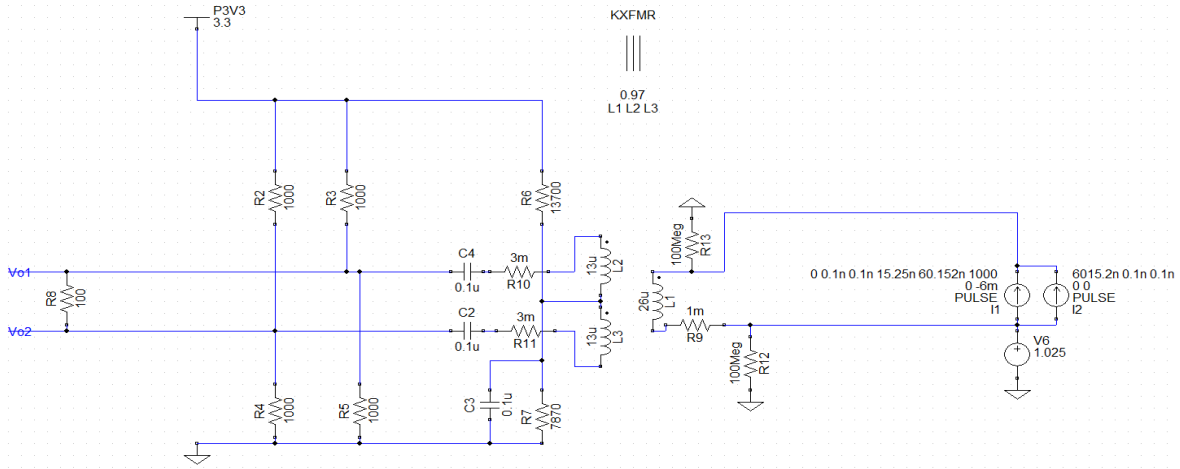

Fig. 4. Galvanic Isolation Model

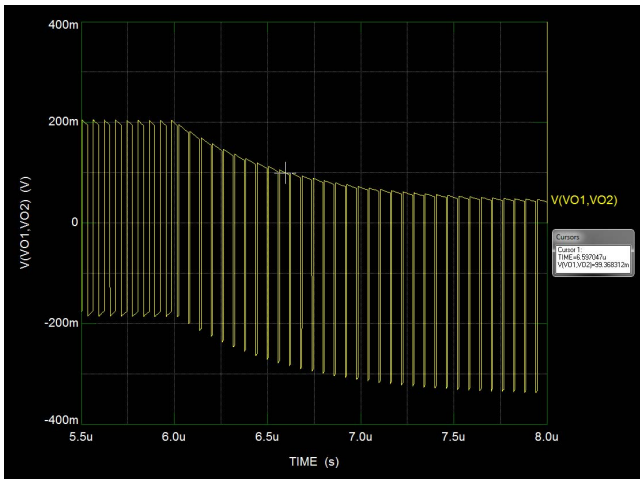

Fig.5. Plot Showing DC Drift

galvanic isolation had to be transparent. Other goals and constraints are described below

There are commercially available tools for analyzing and verifying a SpaceWire link so it was important to keep the electrical interface and the lowest level protocols in place. In particular, the clock recovery encoded between the DATA and STROBE lines. Routers and any other intermediate hardware also needed to be functional so no mid-level protocol changes were acceptable.

The SpaceWire interface in this application was implemented in an FPGA that was resource constrained and already working so any encoding and decoding must be performed at the "Cargo" level which, in this application, meant in the software. This benefited the development effort by simplifying changes and allowing a rapid turn-around of any encoding technique to hardware.

III. Galvanic Isolation Circuit

Figure Fig. 3 is our galvanic isolation circuit. It is similar in design to a circuit presented by Aeroflex at the 2008 International SpaceWire conference. The circuit is simple and requires an isolation transformer only on one end of the interface. The LVDS transmitter and receiver devices are the

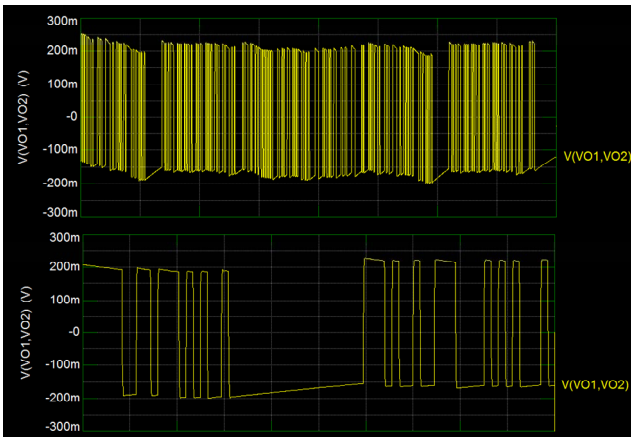

Fig.7 Plot Showing Ideal Random Operation

Aeroflex UT200SpWPHY01. The transformer is an up-screened T-330SCT from Pulse Electronics.

Initial performance without any encoding applied was as expected. The interface would lock and establish at 133Mbps. When the test pattern in our "cargo" was pseudo-random data the link ran with zero bit errors. When other data patterns were presented, the results were also as expected. With all zeros or ones, the link would establish and transfer data until the DC offset was sufficient to exceed the capabilities of the LVDS receiver. Alternating ones and zero data also dropped the link since that pattern has minimal transitions on the Strobe line. It was not surprising that this link would required some form of encoding technique to provide a robust and reliable link. A simulation model was created so that different encoding techniques could be applied and modeled. The model is shown in Fig. 4

The UT200SpWPHY01 has a minimum +/-100mV threshold for deciphering a signal, so the worst case signaling of repeated 0s, or repeated 1s, fails quickly after it starts transmitting (with min/max differential signal falling to <50mV). To get an idea of how quickly, another simulation was run with an initial alternating 0s and 1s pattern that then transitions to nine 0s and one 1 after 6us. As can be seen in Fig.5, after nine packets the signal falls below the +100mV threshold for detecting 1s at the PHY receiver. Fig.7 shows ideal performance.

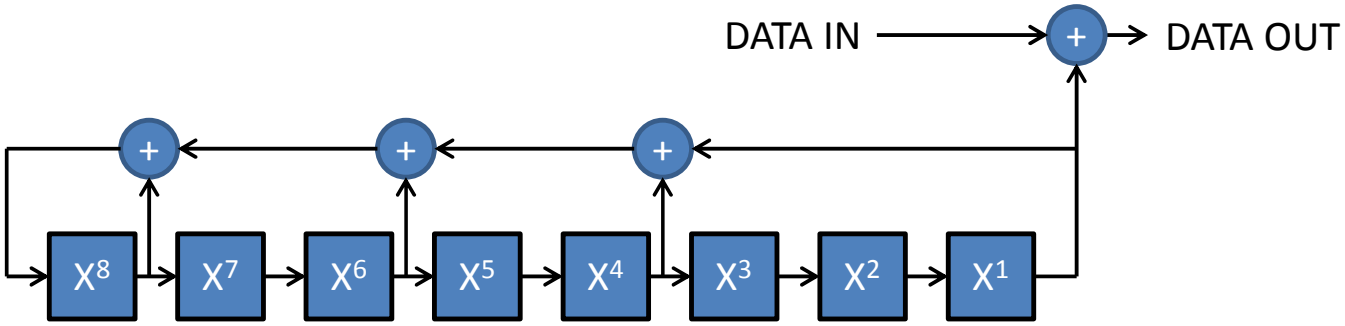

Fig. 6. CCSDS Recommended Randomizer

Table 1. 4b/6b Code

DATA	ENCODING	STROBE XITION	DATA	ENCODING	STROBE XITION
0	011010	1	8	001011	2
1	101001	1	9	100011	3
2	011001	2	A	010011	2
3	110001	3	B	110010	2
4	001101	2	C	001110	3
5	101100	2	D	100110	2
6	011100	3	E	010110	1
7	110100	2	F	100101	1

The simulation shows that the circuit can operate with margin with a sustained 30% transition density. Note that the 30% transition density must be sustained on both the DATA and STROBE signals. We also determined that if the transition density was 45% or better, a continuous string of 19 ones or zeros could be tolerated (note that this pattern could only occur on the strobe due to the parity and control bit appended to DATA)

Additional circuit level simulations were performed to determine the sensitivity of the galvanic isolation circuitry to component values and the topology changes to the isolation circuit. The following variations to the isolation circuit were performed:

A. Removal of DC decoupling caps

Removal of caps has no effect on differential voltage; common-mode voltage decreases (makes sense, the two voltage dividers are now “fighting”).

B. Remove bias for center tap

No effect. This may have more to do with the specifics of model (although it's accurate, the center tap may be to deal with the voltage offset of a real LVDS signal)

C. Remove bias (0.5 voltage dividers)

Without the bias and keeping the caps in place, the differential voltage is now the actual voltage (ie common mode voltage now 0V).

Removing the bias AND the decoupling caps has no benefit to the differential voltage levels, but shows that they're not necessary. The common mode voltage just goes to the center tap bias from the transformer.

D. Change bias for center tap incrementally

No effect seen on the differential signal; the DC bias capacitors prevent any effects from propagating to the receiver.

E. Change 0.5 voltage dividers bias incrementally

Change common mode voltage of received differential signal. This could be optimized by setting it to 1.2V instead of 1.65V, but not required.

F. Change termination resistor

The differential voltage peak-to-peak on the receiver is directly proportional to the termination resistor, as one would expect.

IV. Randomization

The first technique applied to fortify the link was to use the standard CCSDS recommended randomization as shown in Fig. 6 using the polynomial:

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1$$

The randomizer generates a pseudo random sequence that is exclusive-OR'd with the "cargo" on a bit by bit basis. The randomization is likewise removed at the receiving end by again, exclusive-OR'ing the pseudo-random sequence with the incoming cargo data. The pattern is restarted with each packet and all lower level SpaceWire protocols are observed.

In actual real world operation the Randomizer proved to be sufficient to maintain a reliable link since most data sets are predictable and repeat. When analyzed for a random data set there is a possibility for the link to drop with a particular data set. A spreadsheet was developed to simulate the randomizer along with the data formatter so statistics could be collected and evaluated. With that spreadsheet it was easy to create a data set that created a long string of ones or zeroes on either the STROBE or the DATA.

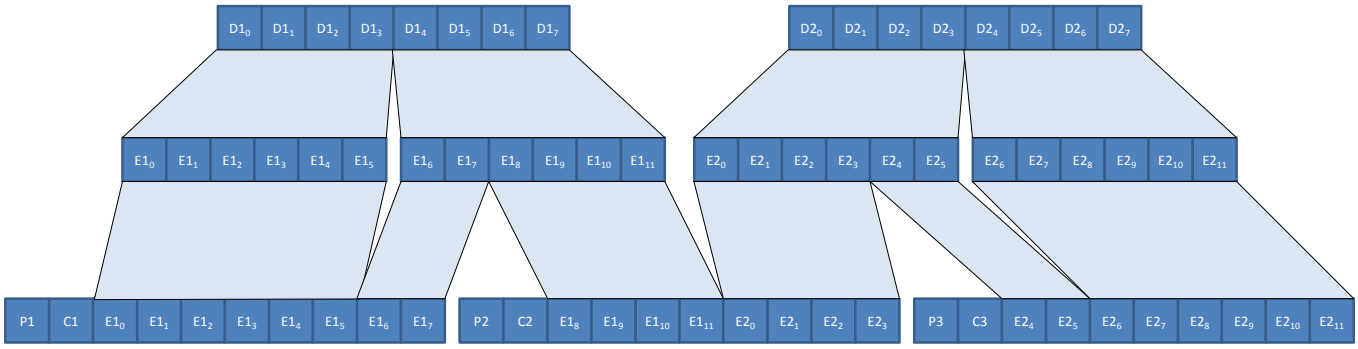

Fig. 5. Modified 16b/30b Encoding

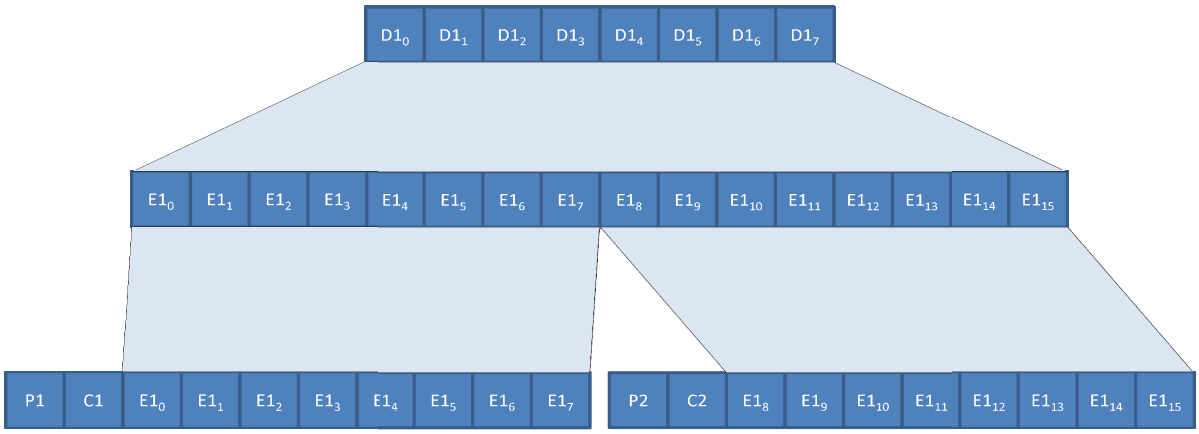

Fig. 6. 8b/20b Encoding

While the likelihood of such a pattern occurring is relatively low in a random environment, it does exist which disqualifies the Randomizer as a potential encoding technique for a critical application. It is worth noting that a changing the randomizer seed by one bit eliminated the link to sensitivity to that patter - but of course made it vulnerable to another.

V. 16B/30B DC BALANCED CODE

Prior work has indicated that a DC Balanced Code may be the best approach to encoding a Galvanically Isolated SpaceWire Link. As with the randomizer, this encoding will be applied at the "Cargo" level. Implementing the DC balanced code impacts usable bandwidth since redundant data must be added to insure sufficient transition densities. The code applied is a modified 4b/6b that is built up into a 16b/24b data set meaning that the packet must be an even number of octets (this was not a significant issue in our application). The 16b/24b encoded data is split into three 8-bit words and encoded per the SpaceWire protocol results in a 16b/30b or close to a 50% overhead link.

Each nibble of the cargo is encoded with a 6-bit word as shown in Table 1. The alternating 0/1 codes are not used since they minimize STROBE transitions. Since each nibble guarantees at least 1 STROBE transition, four nibble will guarantee at least 4 transitions. Four nibbles from two DATA bytes are each encoded creating a 24-bit result. The 24-bit result is split into three bytes which are each further encoded with two additional bits creating a 30-bit result. See Fig. 8. There are 2^{16} possible patterns from the 30 bit DATA result and those correlate to 2^{17} possible patterns for the STROBE (double the DATA since the STROBE uses the preceding bit to determine if it will change state). A spreadsheet was built with 131,072 entries corresponding to every possible state of the DATA and STROBE bits for on 30-bit codeword. From this spreadsheet the maximum number of consecutive ones or zeros was determined along with the average number of ones and zeros so that the resulting DC bias can be determined. Statistics for the encoding technique are as tabulated in

Table 2

Table 2. 16b/30b Encoding Performance

Parameter	Value
Average DATA Transition Density	45%
Average STROBE Transition Density	50%
Longest Consecutive Bits DATA	3
Longest Consecutive Bits STROBE	11

The big advantage to the DC balanced system is that it is deterministic. Every possible pattern of a 30 bit code block was analyzed. The "floating ends" (strings of ones or zeroes at the beginning or end of the code block) were appended to the worst case floating end to calculate the longest consecutive number of ones or zeroes possible within the "cargo". Referring to Table 1 you can see that the modified 16b/30b pattern performs extremely well.

There is a significant cost in efficiency. SpaceWire already uses a form of 8b/10b encoding so there is already a 20% bandwidth hit. The modified drops that efficiency down to 53%. The other layers of SpaceWire protocol drop this efficiency even further. With our candidate 133Mbps, the theoretical maximum data rate is 62 Mbps.

VI. 8B/20B DC BALANCED CODE

Additional modeling and testing was performed with a lower efficiency 8b/20b code that is actually a 8b/16b code that is further encoded by the SpaceWire encoder by the addition of the Parity and Control bits. The 8b/16b code was created to take advantage of known bit patterns to force the STROBE to experience more transitions. This was accomplished by forcing the first eight data bits to always have even parity and

the second eight data bits to have odd parity. With this constraint, the code block will always begin with a 0 (starting with the command bit) and end with a 0 (parity) forcing a transition on the STROBE. Creation of the customer 8b/20b code is shown in Fig. 9.

The 8b/20b code was easier to simulate since there are only 2^8 possibilities to examine meaning that in the "cargo" portion of the packet, there are only 256 type of possible codeblocks. Each DATA field codeblock was created and the maximum number of consecutive bits and the average number of ones calculated. The resulting strobe was then created from each of the codeblocks and the maximum number of consecutive bits and average number of ones calculated. Since every packet ends and begins with a zero, it was easy to determine what the maximum values for the STROBE were as they could not propogate across multiple 20-bit codeblocks.

The code itself is derived from a byte by splitting it into two nibbles that are then encoded per the 4b/6b code shown in Table 1. Two eight bit codewords are the created form the 6-bit codes and inserted into two SpaceWire 10-bit fields. The first 8-bit codeword is created by appending a one in the seventh bit and appending an even-parity bit for bit 8. The second 8-bit codeword is created by appending a zero in the seventh bit and an odd-parity bit for bit 8. The combination of parity and seventh bit encoding forces additional transisitons on the STROBE while keeping the DATA transitions at exactly 50%.

Table 3. 8b/20b Encoding Performance

Parameter	Value
Average DATA Transition Density	50%
Average STROBE Transition Density	45%
Longest Consecutive Bits DATA	4
Longest Consecutive Bits STROBE	7

The 8b/20b has better performance for DC bias, but at a much greater impact to efficiency. The efficiency is a dismal 40% but it is a bit more robust than the Randomizer or the 16b/30b encoding. The 8b/20b does not have the even number of octet constraint that the 16b/30b encoding required.

VII. Summary and Conclusions

Galvanic isolation for SpaceWire does provide many benefits, but without an effective encoding technique is not practical. Simple randomization proved effective in the laboratory environment and with random or even actual payload data worked quite well with no data dropouts or bit errors observed. And randomization achieved this performance at no impact to the link efficiency. However, there is a statistical probability that a data pattern will cause the link to drop and in a critical spacecraft operation, this may not be acceptable.

The modified DC-balanced code performed exactly as expected but at a significant impact to the system bandwidth. The codeblocks were easily realized with a lookup table and provided to the SpaceWire interface as a sequence of two or three bytes. Bit transition densities were sufficient and consecutive strings short. If galvanic isolation is a necessity, and the user can afford the impact to bandwidth (and even number of octet restraint) the presented code will provide a robust and reliable link

References

[1] European Cooperation for Space Standardization, *Space engineering: SpaceWire - Links, nodes, routers and networks* (Noordwijk: ESA-ESTEC, 2008).

[2] Larson, J. (2010). Elimination of Common Mode Voltage Requirements for LVDS used in SpaceWire. *International SpaceWire Conference* (pp. 291-296). Nara: University of Dundee.

[3] Kimmery, C. E. (2011). DC-Balanced Character Encoding for SpaceWire. *International SpaceWire Conference* (pp. 269-278). San Antonio: University of Dundee.

[4] Consultative Committee for Space Data Systems. (2003). *TM Syncrhonization and Channel Coding: Blue Book CCSDS 131.0-B-1*. Washington DC: National Aeronautics and Space Administration.

Radiation-Tested Extended Common Mode LVDS Components

SpaceWire Components

Volodymyr Burkhay, André Rocke
SPACE IC GmbH
Garbsener Landstraße 10, D-30419 Hannover, Germany
v.burkhay@space-ic.com, a.rocke@space-ic.com

Giorgio Magistrati, César Boatella Polo, Gianluca Furano, Farid Guettache
European Space Agency
Keplerlaan 1, NL-2201 AZ Noordwijk ZH, The Netherlands
giorgio.magistrati@esa.int, cesar.boatella.polo@esa.int, gianluca.furano@esa.int, farid.guettache@esa.int

***Abstract*—Extended Common Mode LVDS components from SPACE IC have been tested for their radiation hardness. The collected test results are introduced and discussed.**

***Index Terms*—LVDS, extended common mode, SpaceWire component, SOI, radiation test, TID, SEE.**

I. Introduction

SPACE IC GmbH has been founded by the developers of Extended Common Mode LVDS as a spin-off from the Silicon-On-Insulator (SOI) technologies supplier TELEFUNKEN Semiconductors. SPACE IC will bring space-grade Extended Common Mode LVDS components to the market, which are hermetic packaged and screened according to ESCC specifications. LVDS translator IC components are widely used for SpaceWire (SpW) applications and are absolutely essential for aerospace equipment manufacturers. Recently an extensive demand on radiation hard LVDS components suitable for extended common mode applications at high communication speed arose [1]. Those would help solving some currently existing robustness issues.

The radiation testing started with a high dose-rate Total Ionizing Dose (TID) test on unbiased components followed by Single Event Effects (SEE) tests with heavy ions and laser beam.

II. Tested Components

The SPACE IC extended common mode capable LVDS components comprise LVDS receivers and drivers manufactured using the TELEFUNKEN Semiconductors fully isolated Silicon-On-Insulator (SOI) technology TFSMART2.

Generally SOI technologies are known to mitigate SEE due to much smaller volume of charge collecting silicon compared to bulk devices [2]. If the SOI devices are fully isolated, as this is the case in TFSMART2, they are immune to latch-up thus no single event latch-up can occur. Additionally TFSMART2 features body ties for each device type, which due to charge diversion phenomena in SOI technology enhances the SEE immunity [2]. Combining bipolar and 3.3V CMOS logic devices having 0.35µm minimum feature size with high voltage DMOS devices up to 100V on the same die, this BCD IC manufacturing technology offers a high potential for space applications [3]. Besides latch-up it is also inherently resistant to such parasitic effects as substrate leakage and others thanks to SOI, which improves the performance and makes it suitable for high temperature range.

The extended common mode capable LVDS components have been designed for the combination of the RS-485 receiver input voltage range and high-speed performance and efficiency of LVDS, providing robust but also fast communication channels. Those ICs translate the LVDS signals to 3.3V CMOS/TTL and vice versa with max provided data rate of 400Mbps and higher. The max data rate of such translators is limited by the CMOS I/O circuits.

The radiation testing has been performed on two component types: the LVDS receiver SPLVDS032 [4] and the complementary LVDS driver SPLVDS031 [5]. The SPLVDS031 is a 400Mbps Quad LVDS Line Driver optimized for high-speed, low power, low noise transmission over controlled impedance (approximately 100 Ω) transmission media (e. g. cables, printed circuit board traces, backplanes). The SPLVDS031 accepts four LVCMOS signals and translates them to four LVDS signals. Its differential outputs can be disabled and put in a high-impedance state via two enable pins. Low 300ps (max) channel-to-channel skew and 250ps (max) pulse skew ensure reliable communication in high-speed links that are highly sensitive to timing error. Supply current is 23mA (max). LVDS outputs conform to the ANSI/EIA/TIA-644-A standard. The SPLVDS032 is a 400Mbps Quad LVDS Line Receiver that has to be used in conjunction with the SPLVDS031. The SPLVDS032 accepts four LVDS signals and translates them to four LVCMOS signals. Its outputs can be disabled and put in a high-impedance state via two enable pins. The SPLVDS032 input receiver supports wide input voltage range of -7V to +12V for exceptional noise immunity. Supply current is 7mA (max). LVDS inputs conform to the ANSI/EIA/TIA-644-A standard.

The first application for these products in space is for SpaceWire networks, the use of SpaceWire on board a satellite is rapidly growing from a single point-to-point link btw an instrument characterized by a high data rate and the P/L mass memory to the unique type of bus/network present on the Payload part of Spacecraft in charge to transfer not only scientific data but also housekeeping and command and control messages issued by the OBC. Additionally examples of Platform units as RTU connected to the OBC through a SpaceWire link already exist.

III. Radiation Tests

A. Total Ionizing Dose Test

This test has been performed at the ESTEC ^{60}Co facility using a high dose-rate of 4.5krad/h [3].

The ICs of each of both component types have been divided into 6 groups: 5 irradiated groups and one control group; each irradiated group contained 5 ICs. The 5 groups of both components have been irradiated to the total dose of 5krad, 10krad, 20krad, 40krad and 100krad respectively and the parameter drifts have been measured. (There was a shipping period of 2 days between irradiation and post-radiation measurements.) Then the ICs annealed 7 days at room temperature and 5 hours at the temperature of 100°C, subsequent measurements followed.

The test results are shown in Fig. 1. The drifts of all examined parameters are shown relative to their pre-radiation values. The data points "5krad" to "100krad" are calculated from the mean values of the 5 different groups of ICs irradiated to the corresponding total dose. The data points "after room temperature anneal" and "after hot temperature anneal" belong to the group of ICs irradiated to 100krad total dose.

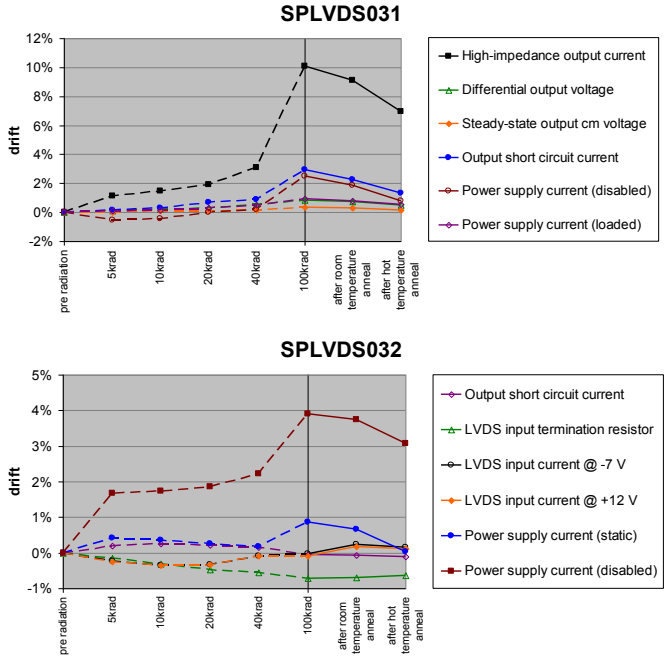

Fig. 1. High dose-rate unbiased TID test results

The shown test results are looking plausible, since the observable drift trend is constant through the total dose steps. The data points near 0% might be more influenced by measurement tolerances. The highest parameter drift is 10% whereas the majority of parameters doesn't show measureable drifts. The SPLVDS031 parameters "Differential output voltage" and "Steady-state output common mode voltage" show low drifts. They indicate that the voltage reference circuit was not significantly impacted by the radiation. The "Output short circuit current" of SPLVDS031 shows that the drift of the current reference circuit might be approximately 3%. The SPLVDS031 parameter "High-impedance output current" has wide tolerances. It shows 10% drift at 100krad total dose which might indicate some degree of degradation in gate oxide properties.

Finally, all tested parts keep their complete functionality after irradiation to the given TID radiation doses, room temperature annealing and accelerated ageing. No critical drifts or specification limit violations have been observed.

B. Heavy Ions Single Event Effects Test

The purpose of single test for heavy ions test is to determine the sensitivity of Single Events Phenomena (SEL, SEU and SET for this application) against LET of incident ions and extract the cross section saturation and LET threshold for calculation and simulation of SEE in orbit.

The SEE test has been conducted in respect of ESA guideline: Single Event Effects Test Method and Guidelines ESCC Basic Specification No. 25100. The test has been performed on two different pairs (driver-receiver pair) of component samples with the case lid removed, the two samples in a pair were irradiated separately and the not irradiated sample was a part of the test equipment for the DUT. Every component has been tested for SEL/SEU/SET. The DUT was a part of SpaceWire communication channel and the behavior has been observed using Link Analyzer and Digital Signal Oscilloscope (see Fig. 2.). The test equipment used in this configuration is able to capture failures causing data corruption and display accurately the behavior of the SpaceWire link during these events. The digital signal oscilloscope captures accurately SET behavior of the devices, being both common and differential mode distortions to the LVDS signal, as well as transients on the CMOS logic outputs of the LVDS receiver.

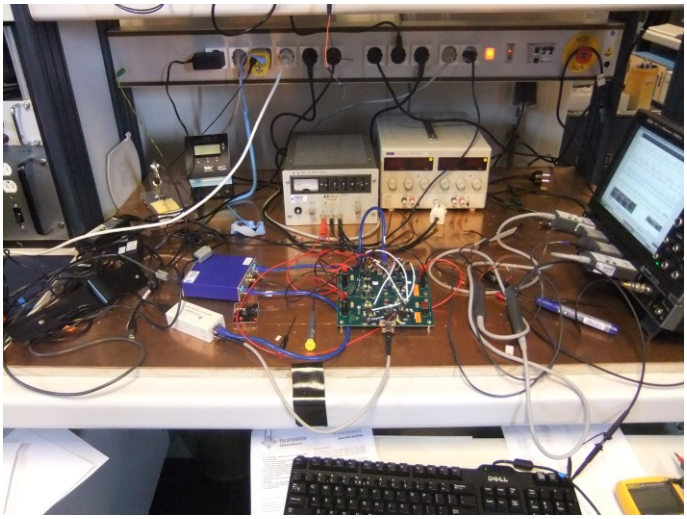

Fig. 2. DUT under functional test in ESTEC Avionics Lab

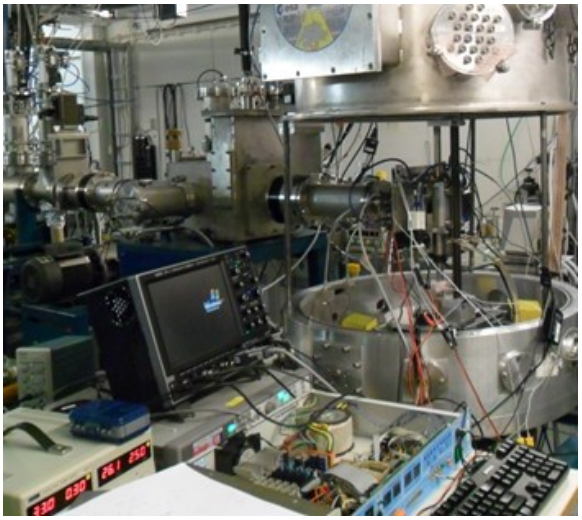

Fig. 3. LVDS under test at RADEF

The test has been performed at RADEF. The RADEF facility is located in the Accelerator Laboratory at the University of Jyväskylä (JYFL). The cyclotron used at JYFL is a versatile, sector-focused accelerator for producing beams from Hydrogen to Xenon. The test has been performed by exposing to heavy ions the LVDS chain based on Space IC SPLVDS031 (driver) an SPLVDS032 (receiver) composing a SpaceWire transmission channel. Driver and receiver have been irradiated separately.

During this test Xenon, Krypton, Iron and Argon ions has been used. The component has been irradiated in air at normal direction (normal incidence has an angle of 0°). The DUT is positioned after 1cm from beam pipe exit. Table 1 summarizes ion characteristics. For each irradiation run a fluence of 5x106 ions/cm2 has been reached.

TABLE I. ION TYPES USED IN THE TEST ON LVDS ICS

Ion	Kinetic Energy in vacuum (MeV)	Kinetic Energy at DUT surf (MeV)	Air distance (mm)	Angle (degree)	Range (µm)	LET (MeV/mg/cm^2)
$^{40}_{Ar}$ +12	372	295	20	0°	70	12.7
$^{56}_{Fe}$ +15	523	408	10	0°	73	20.8
$^{82}_{Kr}$ +22	768	570	10	0°	70	35.1
$^{131}_{Xe}$ +35	1217	856	10	0°	65	65.2

Each device has 4 ports:

One port has been used for static test – static test mode #0: for the driver "1" and "0" have been set at the input and the differential output has been monitored to detect SET induced by radiation. For the receiver the input (differential) has been left unconnected, the output monitored for SETs.

One Port (for the driver and the receiver) has been fed with a clock generated with a pulse generator. The output has been monitored in order to detect variation of the duty cycle (trigger condition: variation > 10-20%) – dynamic test mode #0.

The remaining two ports have been used for transmission of a SpaceWire signal (Data and Strobe) – Link speed 100 Mbit/s. Data have been generated by a Star Dundee Conformance Tester and the traffic monitored by Star Dundee Link Analizer2. The Link Analyzer can generate a trigger for the DSO – dynamic test mode #1.

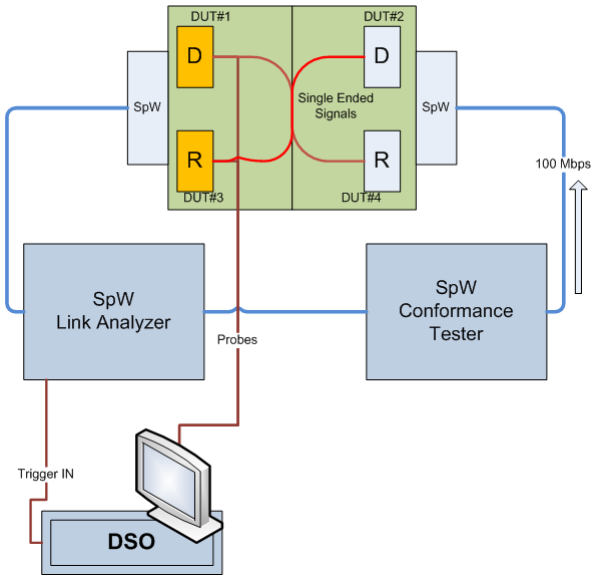

Fig. 4. SET dynamic test mode #1

The results of the Heavy ions test campaign on LVDS are the following.

No SEL was observed up to LET of 60 MeV/(mg/cm^2) at 70-75°C and 3.6V bias voltage during test session with fluence of 1*10^7 ions/cm² for both parts.

The driver SPLVDS031 has shown some SET events only at the higher LET 65.2 MeV/(mg/cm²) during static test mode #0 (see Fig. 5), however neither effects on the duty cycle of the transmitted clock (dynamic test mode #0) nor disconnection/parity errors have been detected.

Fig. 5. SET on SPLVDS031

The receivers SPLVDS032 have shown SETs that have produced disconnections/parity errors on the two ports of the IC used for SpaceWire. SETs have been detected and cross-section measured using Xenon, Krypton, Iron (see Fig. 6). No SET has been observed with Argon (LET = 12.7 MeV/(mg/cm^2)).

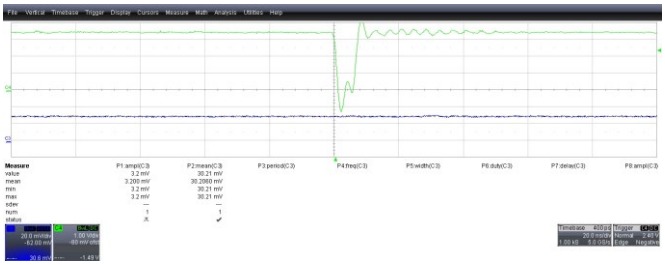

Fig. 6. SET on SPLVDS032

C. Laser Beam Single Event Effects Test

To evaluate the heavy ions SEE test results on SPLVDS032 a laser beam test has been additionally performed on this type of components at MAPRAD srl (Perugia, Italy).

The test conditions were:

- Steady-state "1" (external fail-safe network)
- Solid-state laser with 915nm wavelength
- Laser spot 5-10um
- Working distance of about 12mm
- Pulses length of 1us

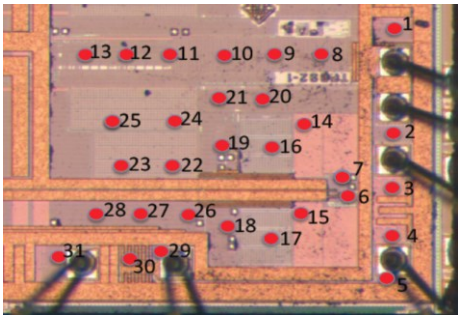

Fig. 7. Laser irradiation points on SPLVDS032 channel circuit

Although 30 points on the whole LVDS receiver channel circuit have been irradiated (see Fig. 7), no visible reaction at the output has been observed.

One possible reason for this might be, that the suspected SET sensitive areas are covered by a thin metallic layer. Another possible reason – the circuit might be not sensitive to the used laser pulses.

IV. Conclusion

Two Extended Common Mode LVDS component types will be made available on the market in space-grade quality.

High dose-rate unbiased TID test and SEE heavy ions and laser beam tests have been performed with promising results.

After the TID exposure none of the component specifications were violated and all tested parts kept their complete functionality.

Under Single Event Effect tests no SEL sensitivity has been observed. The LVDS driver has shown no critical SETs. The LVDS receiver has shown SETs disturbing the data transmitting above LET= 12.7 MeV/(mg/cm^2), but the communication with SpaceWire protocol was still working.

The laser beam test on the LVDS receiver didn't show any correlation with the heavy ion test results. An improved laser beam test is in preparation.

The low dose-rate TID test on biased components will follow soon.

References

[1] J. Ilstad, "ESA's Requirements for future LVDS devices", LVDS Application Workshop, Noordwijk, June 2011

[2] A. Samaras, "JUPITER Mission and Strong Environment" Radiation Specification for System conception, JUICE Mission, CNES, July 2012

[3] V. Burkhay, G. Ilicali, A. Rocke, "Radiation Test of TFSMART2 Technology using Extended Common Mode LVDS and DC-DC Converter Components", 4th International Workshop on Analogue and Mixed Signal Integrated Circuits for Space Applications, Noordwijk, August 2012

[4] SPACE IC, "SPLVDS032RH Quad LVDS Line Receivers with Extended Common Mode" Datasheet, July 2014, http://www.space-ic.com

[5] SPACE IC, "SPLVDS031RH Quad LVDS Line Driver" Datasheet, July 2014, http://www.space-ic.com

[6] G. Magistrati, "Laser Beam Test and SEE Test Report of Telefunken TF6002" , ESA unclassified - for official use, February 2013

[7] G. Magistrati, "Telefunken LVDS SEE Test Report" , ESA unclassified - for official use, August 2013

Atmel's Rad-Hard Sparc V8 Processor
200Mhz Low Power System on chip integrating state of the art Spacewire Router

Nicolas Ganry
Atmel Aerospace, Nantes, France,
nicolas.ganry@atmel.com

Guy Mantelet
Atmel Aerospace, Nantes, France,
guy.mantelet@atmel.com

Steve Parkes
STAR-Dundee Ltd, STAR House, 166 Nethergate, Dundee, DD1 4EE, United Kingdom
steve.parkes@star-dundee.com

Chris McClements
STAR-Dundee Ltd, STAR House, 166 Nethergate, Dundee, DD1 4EE, United Kingdom,
chris.mcclements@star-dundee.com

Abstract— The AT6981 is a new generation of processor designed for critical spaceflight applications, which combines a high-performance SPARC® V8 radiation hard processor, with enough on-chip memory for many aerospace applications and state-of-the-art SpaceWire networking technology from STAR-Dundee. The AT6981 is implemented in Atmel 90nm rad-hard technology, enabling 200 MHz operating speed for the processor with power consumption levels around 1W. This advanced technology allows strong system integration in a SoC with embedded peripherals like CAN, 1553, Ethernet, DDR and embedded memory with 1Mbytes SRAM. The device is ITAR-free and is developed in France by Atmel Aerospace having more than of 30years space experience. This paper describes this new SoC architecture and technical options considered to insure the best performances, the minimum power consumption and high reliability. This device will be available on the market in H2 2014 for evaluation with first flight models targeted end 2015.

Keywords—spaceflight processor, SpaceWire, system-on-chip, networks, spacecraft onboard data-handling, radiation hard processor

I. Introduction

There is a continuous demand for more and more processing power on-board spacecraft to handle sophisticated instrument control, intense data processing and compression, and real-time attitude and orbit control. In addition, increasing autonomy of spaceflight systems requires intelligent on-board management of spacecraft resources. The required processing capability has to be provided at minimal power consumption and it has to be readily integrated into the on-board data-handling and avionics systems. The Atmel AT6981 Castor device is a radiation-hard-by-design processor being developed by Atmel to fulfill this need, providing exceptional processing power per milliwatt and integrating a comprehensive set of peripheral interfaces.

AT6981 is a SPARC® V8 rad-hard processor running at 200MHz and integrating 8 LVDS links SpaceWire router at 200Mbit/s developed in collaboration with STAR-Dundee. Atmel present the status on this new standard space processor during the 2014 International Spacewire conference.

II. Atmel's Unrivalled Flight Heritage

Over the last 16 years, Atmel has been steadily building a space microprocessor strategy based on SPARC architecture. With worldwide sales of over 3000 flight models featuring the Atmel TSC695E and already around 1000 flight models with the Atmel ATF697F, Atmel's SPARC processor roadmap has an unrivalled flight heritage. The upcoming AT6981 rad-hard SPARC V8 processor benefits from this solid experience.

III. AT6981 Product Description

The AT6981 is based on the rad-hard LEON2FT processor and integrates all commonly-used space peripherals including 1553, CAN, SPI, UART, Ethernet & DSU (Debug Support Unit) for debug purposes. A fully-compliant IEEE754 FPU and a LEON2 SPARC V8 native MMU are also embedded. This SoC integration is done in 90nm rad-hard Atmel technology enabling at least 200 MHz operating speed for the processor while consuming less than 1W when used in the same configuration than AT697F. Atmel continues to offer best-in-class power-to-performance ratios that offer more possibilities for space applications by reducing costs, sizes and embedded power supply.

The AT6981 embeds SpaceWire engines offering up to 8 links LVDS router. The SpaceWire links support full hardware initiator and RMAP. Targeted implementation enables some other advanced hardware SpaceWire functionalities such as plug-and-play and determinist.

This state-of-the-art SpaceWire IP has been developed by STAR-Dundee. The AT6981 benefits from strong design cooperation between STAR-Dundee and Atmel to achieve an embedded system with bandwidth of at least 200Mbit/s.

The AT6981 will be available in 256 MQFP package.

In addition to a powerful SPARC V8 processor core with a high level of integration and performance, the AT6981 embeds a 1-Mbyte hardened SRAM memory for PCB area savings and fast access at full CPU speed. For external memory, SRAM and DDR1/2 interfaces are proposed.

In order to facilitate analog-to-digital operations and provide an even higher level of integration, the AT6981 embeds a waveform generation (PWM) dedicated unit for analog control/command and proposes some ADC/DAC interfaces for analog acquisitions/conversions.

The AT6981 is a rad-hard by design processor that will be space qualified and will support:

- Total dose of 300Krads (Si) according to the MIL-STD883 method 1019
- No Single Event Latch up below a LET threshold of 95 MeV.cm²/mg
- No Single Event Upset below a LET threshold of 10 MeV.cm²/mg and a cross section of 5 E-8 cm2/bit
- SEU error rate better than 1 E-3 error/device/day

IV. AT6981 Castor Architecture

The architecture of the AT6981 Castor device is illustrated in Fig. I.

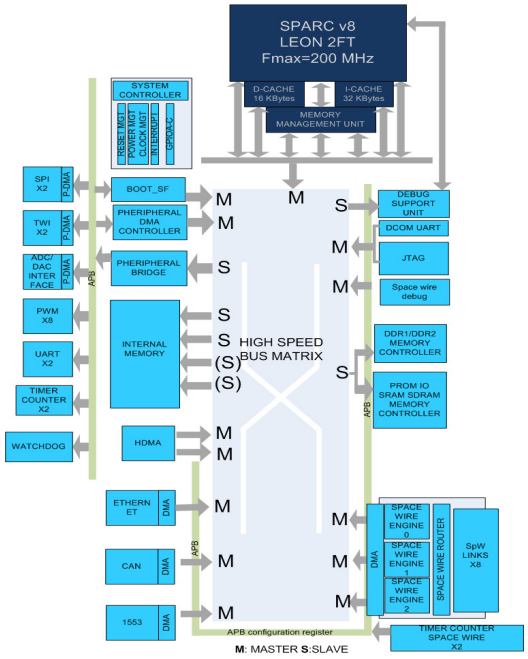

Fig. I: AT6981 Architecture

The AT6981 Castor device is a complete system on chip, with processor, memory and peripherals interconnected via a high performance interconnection switch matrix. The switch matrix at the heart of the AT6981 device connects the processor, memory banks, SpaceWire engines and other IO functions. Several internal RAM blocks are provided to support concurrent memory accesses by the processor and IO facilities.

The three SpaceWire engines, Ethernet, CAN and MILSTD 1553 interfaces are all connected as master devices to the switch matrix allowing them to read and write to the memory using distributed DMA capability.

Lower speed peripheral devices including SPI, TWI, UART, timers, watchdog timers, PMW, ADC interface, DAC interface, parallel input/output and interrupts, are connected via an APB bus and peripheral bridge to the switch matrix.

Various forms of external memory (PROM, SRAM, SDRAM and DDR) can be attached directly to the AT6981 devices, providing ready of expansion of the internal memory when required.

Each of the major components within the AT6981 Castor device will now be described in more detail.

A. Processor

The processor is a SPARC® V8, LEON2-FT processor with integrated floating-point unit, providing excellent processing performance:

- > 150 MIPS (Dhrystone 2.1)
- > 40 MFLOPS (Whetstone)

The particular SPARC V8 architecture is a 32-bit architecture using a 5-stage pipeline and eight register windows. Tighly coupled instruction and data cache memory is provided on chip as follows:

- 32 kbyte Multi-sets Data Cache
- 16 kbyte Multi-sets Instruction Cache.

Native MMU of the LEON2 SparcV8 architecture is activated. The processor has an Advanced High-performance Bus (AHB) and includes a memory management unit (MMU) with up to 32 table entries.

The design can support an internal clock frequency of 200 MHz with a processor input/output toggling frequency of 100 MHz. The core is designed for low power operation with exceptionally low power per MIPS.

The integrated floating-point unit supports 32 single-precision and 64-bit double precision fully compliant to IEEE 754 floating-point standard.

The processor supports booting from both 8-bit and 40-bit PROM interfaces, from serial PROM Flash and from Spacewire link

This single CPU core architecture device allows an easy and safe migration of the software from AT697F without compromise performances. AT6981 benefits from all development tools available for LEON core as it offers a standard DSU interface for trace and debug.

B. Interconnection Switch Matrix

The AT6981 bus architecture is unique on space market. This device takes benefit from Atmel strong IP portfolio and powerful architecture coming from the commercial microcontroller business where Atmel is one of the leaders today.

The AT6981 System on Chip is built around a HMatrix bus which is multi AHB compliant and brings some AHB arbitration mechanisms to support multiple transfers. By this well proven Atmel technology, conflicts management for concurrent access is becoming much easier, even completely transparent for the CPU core running software.

For example, you can manage in parallel all those activities without impact the main CPU core execution:

- Run three Space Wire 200Mbit/s transfers
- Run two 1553 communication flows
- Run two high speed CAN transfers
- Run a MAC Ethernet 100Mbit/s connection
- Run SPI or TWI sessions as well

Each peripheral can be connected to any of the eight protected memory areas and can take benefit from the 200MHz x 32bits AHB bus bandwidth without disturbing CPU internal operations. During full speed transfer session, processor is never interrupted and has a fully deterministic behavior to manage control of all operations. Switching capacity insure any of 16 masters to be connected to any 16 slaves at maximum. This capacity is half used on Castor device which allow some redundancy.

Switch matrix IP used the same hardened techniques than all other IPs with TMR and use of hardened cells.

This architecture, which provides up to 6.4 Gbit/s bandwidth per Hmatrix links, is ready for targeted future evolution like SpaceFibre, Gbit Ethernet and multi-core. It will enable a smooth transition for coming product derivatives of this high speed SPARC® V8 architecture.

C. Memory

As well as the on-chip processor cache the AT6981 Castor device includes 1 Mbyte of radiation tolerant SRAM. The internal memory is separated into eight banks each of 32k x 32-bits, so that several concurrent transfers into and out of memory can be supported by the interconnection switch matrix.

In addition to the internal memory the AT6981 Castor device support the direct connection of a range of external memory devices, including PROM/Flash, SRAM, SDRAM and DDR devices. EDAC protection for external memories is provided as required.

The memory interface can be configured to operate as 8-bit or 32-bit wide as AT697F. It is also possible to load a program through spacewire link into the internal SRAM for standalone operation, without external RAM memory.

D. Peripherals

In addition to SpaceWire, the AT6981 Castor device includes many other commonly used data and control interfaces used on board spacecraft. It provides:

- Redundant CAN Bus interfaces supporting version 2.0 Part B of the CAN bus specification and providing 15 channels,
- Redundant 1553 interfaces which can operate as a bus controller or as a remote terminal,
- An Ethernet interface which can operate at 100 Mbits/s.

Each of these interfaces has its own DMA controller.

A comprehensive range of lower speed peripheral interfaces is also provides on-chip: SPI, I2C, ADC, DAC, UART, Timers …

The slower speed peripheral devices are connected to an Advanced Peripheral Bus (APB) which is bridged to the AHB interconnection switch matrix.

Attached to the APB are two Serial Peripheral Interfaces (SPI) with a dedicated DMA controller, two Two Wire Interfaces (otherwise known as I2C interfaces) again with a dedicated DMA controller, and two eight-bit UARTS.

Four 32-bit timers are provided along with a 32-bit watchdog timer. The interrupt controller provides support for 31 interrupts. 96-bits of general parallel input/output is provided.

Analogue interfacing is supported with a pulse width modulator (PWM) for analog control/command and parallel interfaces for connection of Analogue to Digital Converters (ADC) and Digital to Analogue Converters (DAC). The ADC and DAC interface to support efficient analogue data acquisition into memory.

External FPGA connection to AT6981 is possible through parallel interface up to 200Mhz or through SPI link up to 100Mhz, both with capability of DMA transfer.

E. Spacewire

SpaceWire is one on the main data-handling interfaces used on board spacecraft today [1][2]. The AT6981 Castor device includes extensive, state-of-the-art support for SpaceWire; a SpaceWire router with eight external SpaceWire ports and three SpaceWire protocol handling engines designed especially for Castor. Protocol support is provided for the SpaceWire Remote Memory Access Protocol (RMAP) [3], the SpaceWire plug and play protocol [4], and SpaceWire-D the deterministic data delivery protocol [5]. Extensive time-code support is also provided including multiple time-code counters and distributed interrupt time-codes [6].

The SpaceWire architecture used in the AT6981 Castor device is illustrated in Fig. III.

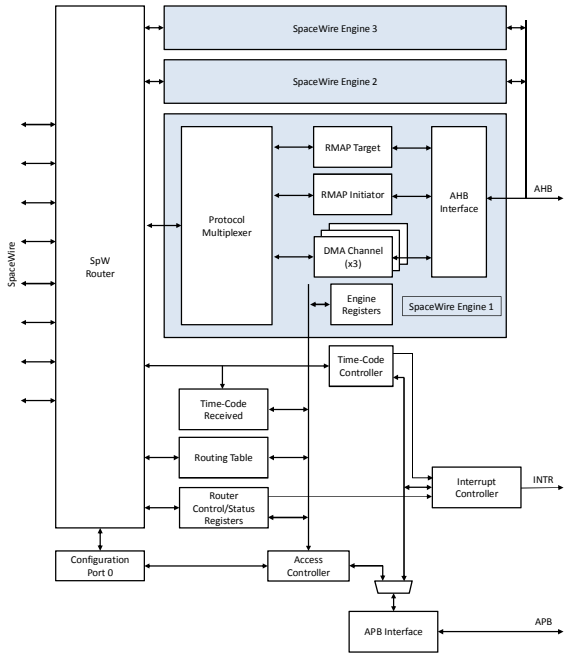

Fig. II: SpaceWire Architecture of Castor

1) SpaceWire Router

The Castor on-chip SpaceWire router has eight SpaceWire interfaces, three interfaces to the SpaceWire Engines and an internal configuration port which supports configuration of the SpaceWire router and engines using the SpaceWire RMAP protocol. An APB interface may also be used to configure and read status registers from SpaceWire engines, time-code controller and SpaceWire router. The router based on proven STAR-Dundee technology [7] supports many advanced features such as start-on-request, disable-on-silence, and watchdog timers on all SpaceWire ports.

2) SpaceWire Engines

The Castor device contains three SpaceWire engines which can support the rapid processing of various SpaceWire protocols. Comprehensive support is provided for the SpaceWire Remote Memory Access Protocol (RMAP) which is widely used on board spacecraft for configuration and control [8]. Support is also provided for user defined protocols with a multi-channel, protocol selective DMA controller. Each SpaceWire engine contains an RMAP Target, RMAP Initiator, DMA controller, Protocol Multiplexer and a set of configuration and control registers.

3) RMAP Target

The RMAP Target allows Castor to act as an RMAP Target device, receiving, reacting and responding to RMAP commands from a remote RMAP Initiator device. When an RMAP command is received, its header is checked and if valid, the information it contains is used to authorise or reject the command. Authorisation can be carried out by the Castor processor, or by automatically checking that the RMAP command wants to read or write from a predefined area of memory. If the command is authorised, it is executed and data written to or read from the specified area of memory. An RMAP reply is then sent to the initiator of the RMAP command, containing an acknowledgement along with any data read from memory. A 16-byte verified write buffer is provided to support verified write commands where the data associated with a write command is validated using a CRC before it is written to memory. This supports commanding of critical activities using RMAP.

4) RMAP Initiator

The RMAP Initiator off-loads the Castor processor from the generation of RMAP commands. The processor defines the RMAP commands to be sent, placing the commands and any write data in memory and reserving space in Castor memory for any data expected to be returned in response to read commands. The RMAP Initiator is then started and will automatically send all of the commands and collect the replies, informing the processor when the entire set of RMAP transaction is complete.

5) DMA Packet Transmission

The DMA controller supports multiple concurrent transmit channels which can be programmed to send one or more SpaceWire packets. A packet consists of one or multiple data chunks stored in different memory locations. This allows the packet header to be stored in a separate location to that of the packet data content. The DMA controller is given a list of the chunks making up a packet and will construct the required packet as it is being sent.

The sending of CCSDS Packet Utilizatoin Standard (PUS) [9] packets is supported in Castor by providing a means for computing the CRC-16 checksum in hardware.

Continuous transmission of packets is supported with a circular buffer mechanism containing data and packet descriptor pointers. Interrupts can be set to monitor the progress of transmission of packets without halting the actual operation. This makes it possible to achieve the maximum SpaceWire data-rate with minimum CPU utilization. Errors in one channel do not affect the operation of other channels.

6) DMA Packet Reception

A DMA receive channel receives data from the protocol multiplexer and writes it to Castor memory. Each receive channel is associated to a particular SpaceWire protocol or packet type using a packet filter in the protocol multiplexer which switches packets based on their first four bytes. Packets which are received on the same DMA channel are stored contiguously in memory and their packet length is stored in packet descriptors. Reception of PUS packets is supported by providing the hardware computation of its CRC-16. CRC-8 calculation is also supported in hardware.

Continuous reception of packets is provided with a circular buffer mechanism that stores data and packet descriptor

pointers. It is possible to enforce that a packet is not split at the end of a memory region. Interrupts can be set to monitor the progress of packets received without halting the actual operation. The user application or software driver frees the space used by packets once the data received has been processed. This procedure allows data to be received at the maximum SpaceWire data rate with minimum CPU utilization. When an error occurs the reception is halted and the processor is interrupted.

7) Protocol Multiplexer

The protocol multiplexer multiplexes packets to and from the RMAP Target, RMAP Initiator and the DMA controllers. When appropriate it uses the SpaceWire Protocol Identifier [10] to perform the multiplexing. Alternatively other relevant information at the start of the packet can be used.

When sending packets, the multiplexer selects the next packet to be sent from one of the possible sources (RMAP Initiator, RMAP Target, three DMA transmit channels) and waits for the end of packet before selecting the next packet to be transmitted.

When receiving packets, the protocol de-multiplexer checks the first four packet bytes against a configurable pattern and mask to determine the destination of the packet; either the RMAP target, the RMAP initiator or a specific DMA channel. The pattern and mask are programmable by the host processor through the APB registers. This allows multiplexing of packets according to their SpaceWire Protocol Identifier or other information at the start of the packet.

8) Time-Code Controller

The SpaceWire time-code controller is able to forward received time-codes and to generate time-codes from software, processor timer interrupt or an internal dedicated time-code master counter. The time-code controller has a time-code register for each of the four time-code flags, therefore allowing independent time-code forwarding for each flag code.

The time-code controller stores the last time-code received for each type of control flag and can indicate to the host that a time-code has been received through the status/interrupt interface.

The controller can act as a time-code master either by software insertion of a time-code, sending time-code on a processor timer interrupt or by setting up an internal time-code master counter, which automatically sends time-codes periodically. The time-code frequency can be controlled by the host software with up to 1 micro-second precision.

9) SpaceWire Interrupt Controller

The SpaceWire interrupt controller provides event notification to the host processor for packet, time-code and error events.

F. Power Management

The AT6981Castor device is a low power device with dedicated mechanisms for adapting power consumption to the level of processing performance required by the application. Specific techniques used for power management include:

- Programmable clock functions that provide the clock for each main function and peripherals, which are able to adjust the clock speed and to gate it off completely.
- Dedicated reset for each major function which allows them to be reinitialized locally after their clock restarts.

The estimate power consumption of the AT6981 Castor device is as follows:

- Core stand-by current target: <100mA, mostly internal memory leakage
- Core operating current target: 5mA/MHz

G. Rad Hard by design

The AT6981 Castor device is designed for spaceflight application and is fault tolerance by design. It uses low level, full triple modular redundancy (TMR) along with single event transient (SET) filtering to provide radiation tolerance of its internal logic. Memory is protected using EDAC which is capable of single error correction and double error detection.

All internal memories have a dedicated scrubber with internal EDAC in order to manage auto correction.

This scrubber is fully programmable on period of the scrubbing cycle and the protected RAM array. It is an additional value to the external EDAC capability provided with the 1Mbytes of on chip available high speed SRAM to allow customer own correction management.

All Memory blocks are designed in a way to never have any adjacent bits for a same word. This technique simplifies strongly the error management activities which allow using only a simple EDAC for data single event protection. By this way it's not needed to implement an heavy TMR mechanisms to protect register files which trigger some potential performances limitation.

TMR mechanisms are implemented on all logic of the design with also an SET filtering method.

Rad hard libraries on this proposed 90nm technology are developed by Atmel in France based on all well proven libraries from Atmel commercial products. AT6981 benefits from the strong 30 years' experience of Atmel France in rad hardening techniques.

H. Debug and Test Facilities

The AT6981 Castor device has comprehensive debug support with a processor debug support unit (DSU) supporting debugging, trace memory and hardware watch points. The DSU can be accessed through a UART, IEEE 1149.1 JTAG Interface, or at high speed through a SpaceWire interface.

I. Operating Range

The AT6981 Castor device operates from two supply voltages, minimizing the need for external power supply components while keeping power consumption low:

- 3.3V +/- 0.30V for input/output,
- 1.0V +/- 0.15V for the core.

Castor operates over an ambient temperature range of -55 to +125°C (Tj max 145°C).

The inputs/outputs are cold sparing.

V. AT6981 SoC Verification

AT6981 is a complex System on Chip device where complexity management and control at each stage of the development is key to insure the best product maturity and time to market. Several verification steps are handled all along the component development cycle as follow:

- Performances verification at early stage of the design. All IPs have been unitary tested regarding functionality and performances with modelisation techniques and profiling. All IPs have been also ported and verified on FPGA
- A full front end RTL functional validation has been done after the IPs integration. Full set of Atmel embedded software described later has been used to exercise all applicative configurations.
- Multilayer package routing validation with a special focus on all differential links DDR and LVDS to verify propagations and synchronizations. During this phase RLC effects have been minimized by electrical simulations as much as possible to insure the 200Mhz IO toggling
- Several Tests vehicle have been initiated to validate the Rad Hard libraries used w 90nm selected silicon to support radiations effects. Those test vehicle have been used to assess:
    - SET width variation at different ion energies to size clock skewing of the TMR cells
    - EDAC and TMR errors feed-back handled by processor TRAP interrupt
    - IOs libraries cold sparing mechanisms and ESD protection level
    - Devices charge sharing effects to limit multiple devices upset
    - SEU and SEL libraries capabilities
- Set up a pre silicon development kit based on FPGA to validate design in the system with all peripherals and embedded software. It's the way to anticipate the applications tests before prototypes are available

At each stage of testing, traceability is insured between product requirements and test results. For final system tests with hardware and software, applications cases are used to handle the system test coverage.

VI. AT6981 Evaluation kit, Software & Services

Several options proposed by Atmel and partners for the evaluation kit including hardware and software services according to the development steps and customer engagement.

1) Atmel FPGA pre silicon evaluation environment

In order to manage system validation of AT6981 device before silicon freeze, Atmel has developed a FPGA test board based on Xilinx Virtex which contains the final and complete RTL code. This FPGA board supports all castor peripherals interface like Spacewire, CAN, 1553, DDR, etc … With this board, end user can manage a full functional evaluation of the CASTOR device, develop and exercise the full embedded software. Level of performances will be of slightly limited compare to real silicon device but largely enough to port real time operating system like RTEMS or VXWorks.

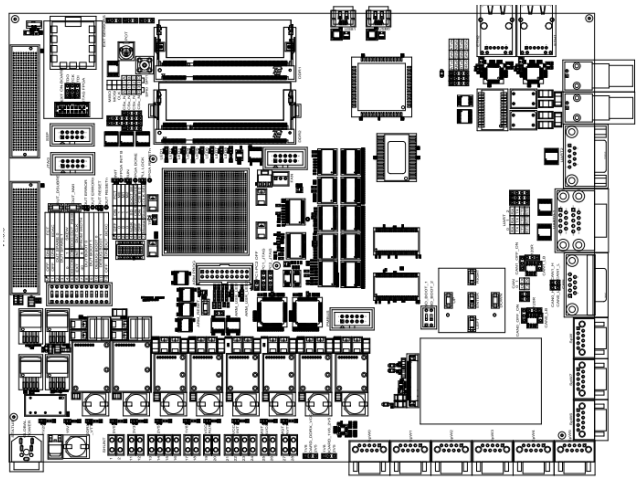

Fig. III: Atmel FPGA pre silicon environment

2) Atmel Hardware evaluation kit

As soon as first AT6981 prototypes will be available, associated Atmel Hardware evaluation kit will be also available to complete application tests and to manage performances assessment. AT6981 hardware kit design will be definitively aligned on the current Sparc V8 kit architecture like for AT697F or ATF697FF.

Fig. IV: Atmel Hardware evaluation kit

3) Extended Hardware kit from partners

StarDundee is providing an extended Spacewire FPGA Test Board which allow to explore some extended capabilities related to SpaceWire interface, SpaceWire protocol engines and SpaceWire router.

4) Atmel Software and Services

With AT6981 Atmel hardware kit and Atmel pre silicon environment, Atmel is offering a complete ecosystem of software and tools which are used for the full chipset validation and qualification. This guarantees the best starting point for end user application development. A full software package including embedded software drivers and libraries together with a Basic Tools set for debug, download and trace are proposed to AT6981 developers.

For embedded software package proposed with AT6981, software drivers and libraries architecture concept is fully reused from Atmel industrial and automotive microcontrollers. This Atmel named ASF architecture is widely deployed and proven with already some OS port facilities available.

Atmel ASF is structured in stacks. Each stack is composed of two layers:

- The drivers layer
- The services layer

The software is also embedding a Bootloader mechanism.

Main stack are:

- The memory stack:

The aim of the memory stack is to implement a memory virtualization. The virtualization allows to use every types of supported memory in the same way (memory controller). It also allows protecting memory at processes level (Memory Management Unit) and dynamic allocation.

- The I/O stack:

The I/O stack manage general purpose IO, PWM and ADC/DAC interface

- The communication stack:

The communication stack allows data transfer through all communication interface using standard protocols such as TCP/IP. The stack allows communication through all present interfaces:

- o Ethernet
- o Mil STD 1553
- o Space Wire
- o CAN
- o TWI
- o SPI
- o UART

- The system stack:

The system stack handle with architecture specificities such as system traps, windows management, interrupt. It also manages system self test and diagnostic (e.g. edac).

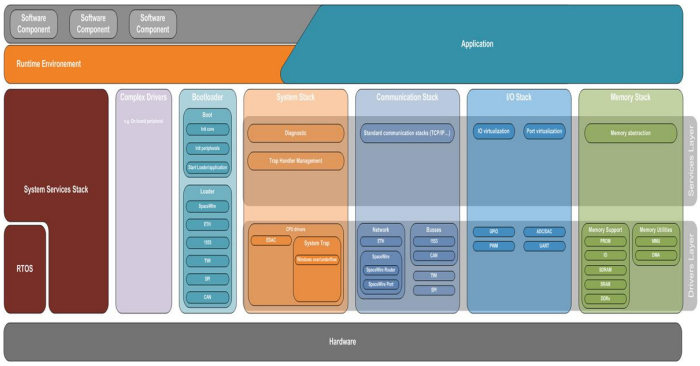

Fig. V: AT6981 embedded software architecture

By taking advantage of all proposed software buildings blocks available with the AT6981, end user is able to manage his own system design and improve targeted application time-to-market.

5) Extended Software services from partners

AT6981 developers will also benefit from additional services and extended tool set that will be proposed by Atmel partners as the comprehensive software development environment suite from STAR-Dundee.

For more information related to Star-Dundee proposed services with AT6981, please refer to http://www.star-dundee.com/

VII. AT6981 SCHEDULE

The AT6981 is in its final stages of development and evaluation by customers is targeted to start Q3 2014. Flight models are targeted to be full space qualified end 2015.

VIII. AT6981 TARGETED SPACE APPLICATIONS

Providing integration of more peripherals and memory blocks around the SPARC V8 processor core enables size, weight and cost improvements for today's space applications: on-board computing, data handling, telemetry/telecommand, remote terminal units, sensors, instruments and payloads. In addition to strong system integration value, the AT6981 enables more powerful processing with 200MHz and embedded fast memory to follow higher bandwidth capabilities of peripherals with SpaceWire 200Mbit/s.

In this section several applications of the AT6981 Castor Processor will be described. A generic functional block diagram of a spacecraft avionics system is illustrated in Fig. IV, from which the various applications of Castor will be explored.

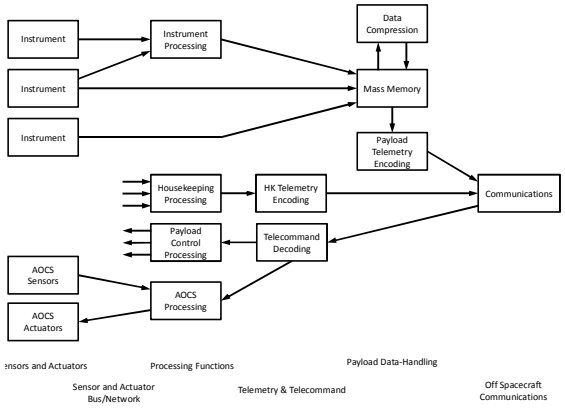

Fig. VI: Spacecraft Avionics Functions

A. Instruments

The instruments or payloads are the reason for the spacecraft being launched. Instruments support the science, earth observation or commercial application of the spacecraft. The instruments sense the environment across the electromagnetic spectrum and convert sensor signals into digital data that can be handled on board the spacecraft and sent to Earth. Instruments may be passive, receiving information from the environment, or active sending out signals which interact with the environment, reflecting some of the transmitted energy back to the sensor on board the spacecraft. Instruments may be relatively simple requiring very basic control, or sophisticated requiring substantial interactive control.

The Castor device is perfect for instrument processing owing to its wealth of onboard resources and its low power consumption. A typical instrument control application will run within the on-chip memory of Castor and only the peripherals that are required need to be activated and consume power. The on-chip SpaceWire technology is able to off-load the communications from the Castor processor simplifying the sending of instrument data to the onboard mass memory unit. The SPI, TWI, ADC/DAC and parallel IO interfaces of Castor are designed to support direct connection to the instrument electronics. If an FPGA is being used within the instrument Castor is readily connected to the FPGA. The integrated SpaceWire RMAP target handling capability is perfect for controlling the instrument with minimal processor overhead.

B. Instrument Processing

Some instruments may require a substantial amount of processing power and may require inputs from other sensors or instruments in order for the instrument processing to be carried out. In this case a separate instrument processor may be required. This may be located within the main instrument itself or in a separate instrument processing box.

Castor has substantial processing capability (>150 MIPS) with a floating-point unit that is able to provide 40 MFLOPS. This is adequate for many demanding instrument processing applications. Where more processing power is required two or more Castor devices may be easily networked and used in parallel thanks to the embedded SpaceWire router.

C. Mass Memory

The onboard mass-memory provides data storage facilities for the various instruments, managing the available storage, allocating it according to mission priorities sent from ground via the telecommand system.

A processor is typically used to control the mass-memory system, allocating memory resources to the various instruments or more specifically to particular application processes identified by their application process identifiers (APIDs). SpaceWire has been used in several mass memory systems either for control, or for memory module interconnection or both. The Castor processor with its embedded SpaceWire router and MILSTD 1553 interfaces is just right for mass memory control. Its on-chip memory is perfect for command buffering and the integrated SpaceWire Engines with their multi-channel DMA controllers are able to offload the processor from much of the work distributing commands. The internal SpaceWire router can also save on one of the SpaceWire router devices that may be needed for the mass-memory internal network, or the external payload data-handling network.

D. Data Compression

A data compression processor may be attached to the mass-memory to compress data from one or more of the payloads to save on downlink telemetry bandwidth, enabling more payload data to be sent to ground. Both loss-less and lossy (e.g. JPEG) compression may be required to cover a variety of different types of data.

The Castor device with its high processing power can be used for some data compression tasks, especially if high-speed data is buffered into the mass-memory first and subsequently read out, compressed and written back to the mass-memory, before being telemetered to ground. The Castor device is not suitable for very high data-rate compression, but if an FPGA or ASIC compression device is being used then Castor may have a role in configuring and controlling the data compression

device, and in performing some of the less regular processing required for rate control.

E. Housekeeping and Autonomy

The overall health of the spacecraft, including power consumption, operating temperature, and battery status is monitored by the housekeeping function. To perform this monitoring function it gathers housekeeping information from the various spacecraft subsystems, and reports this information to ground via the telemetry encoding function and communications subsystem. Autonomous operation of the spacecraft may be supported in the event of anomalous conditions occurring.

Castor has a wide variety of interfaces to support housekeeping. It also has adequate processing capability to support complex autonomous operational modes should that be required. Where a SpaceWire network is being used for on-board data-handling, it is straightforward for Castor to support periodic housekeeping information gathering from any device on the SpaceWire network. The SpaceWire Engines in Castor include RMAP Initiators which may be set up with a sequence of RMAP commands to acquire housekeeping information. This sequence of commands can then be triggered every so often by the processor and the RMAP initiator will send out all the commands, gather the replies, put the data from those replies into specified areas of Castor memory, and then signal to the processor that the entire information gathering exercise is complete. Any errors or missing replies are also reported to the processor. This offloading of routine SpaceWire communication tasks from the Castor processor is a major benefit of the Castor architecture.

F. Telecommand and Payload Control

The instruments are activated and deactivated by the payload control processing function, which is in overall control of the spacecraft operation. It determines what onboard resources are activated at any particular time. It receives telecommands from the communications subsystem which are decoded by the telecommand decoding function.

Software running on Castor is able to decode incoming telecommands, validate those commands and execute them. Once again the large amount of on-chip memory provided in Castor is often adequate for the telecommand and payload control processing.

G. AOCS Processing

AOCS processing receives required attitude and orbit parameters from the telecommand system, and is responsible for maintaining the required orbit and attitude of the spacecraft. To achieve this, the AOCS processor will read the AOCS sensors (e.g. star-tracker, gyro, accelerometer, GPS) to determine its current pointing and orbital position, and control various AOCS actuators to achieve the desired attitude and orbit. AOCS processing is mathematically intense and often requires support for floating-point maths. The processing has to be deterministic.

Castor has the floating-point capability necessary for most AOCS processing applications, its on-chip memory is adequate for most AOCS software, and its range of peripheral interfaces is sufficient for almost all important spacecraft AOCS sensors.

IX. CONCLUSION

International Spacewire conference 2014 is the ideal place to update the worldwide space community on the progress of new coming device. This processor is a highly capable processor, which operates with low power consumption, incorporates substantial on-chip memory and includes an extensive set of on-chip peripherals. It has been designed specifically for spaceflight, meeting the requirements of many different onboard processing applications. On top of component itself, a whole ecosystem including some Hardware and Software facilities that will be provided by Atmel and partners in order to facilitate fast ramp up, reuse & time to market for space actors.

REFERENCES

[1] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 1, European Cooperation for Space Data Standardization, July 2008.

[2] S. Parkes & P. Armbruster, “SpaceWire: Spacecraft onboard data-handling network”, International Astronautical Congress 2008, paper IAC-08-B2.4.1.

[3] ECSS, “SpaceWire - Remote memory access protocol”, ECSS-E-ST-50-52C, Feb 2010.

[4] P. Mendham, S. Parkes, “SpaceWire Plug-and-play: a Roadmap”, International SpaceWire conference, Nara, Japan, 2008.

[5] S. Parkes, Albert. Ferrer, S Mills, A Mason, “SpaceWire-D: Deterministic data delivery with SpaceWire”, International SpaceWire conference, Russia, 2010.

[6] Yuriy Sheynin, Sergey Gorbatchev, Liudmila Onishchenko, “Real-Time Signalling in SpaceWire Networks”, International SpaceWire conference, Nara, Japan, 2008.

[7] S. Parkes, C. McClements, G. Kempf, S. Fishcher, P. Fabry, A. Leon, “SpaceWire Router ASIC”, International SpaceWire Conference, Dundee, 2007.

[8] Chris McClements, Steve Parkes, “SpaceWire RMAP IP Core”, International SpaceWire conference, Russia, 2010.

[9] ECSS standard ECSS-E-70-41A, “Ground systems and operations – Telemetry and telecommand packet utilization”, European Cooperation for Space Data Standardization, January 2003.

[10] ECSS standard ECSS-E-ST-50-51C, “SpaceWire engineering: SpaceWire protocol identification”, European Cooperation for Space Data Standardization, February 2010.

SpaceFibre (Long)

SpaceFibre Implementation, Test and Validation

SpaceFibre, Long Paper

Steve Parkes, Chris McClements, David McLaren,
Angel Monera Martinez,
Space Technology Centre, University of Dundee,
166 Nethergate, Dundee, DD1 4EE, UK
smparkes@dundee.ac.uk

Albert Ferrer Florit, Alberto Gonzalez Villafranca,
STAR-Dundee Ltd.,
STAR House, 166 Nethergate, Dundee, DD1 4EE, UK

***Abstract*—SpaceFibre is a multi-gigabit/s data link and network technology for use onboard spacecraft. Compatible with SpaceWire at the packet level, SpaceFibre runs over electrical and optical media. It provides extensive quality of service (QoS) and fault detection, isolation and recovery (FDIR) capabilities that are designed specifically for spacecraft applications. This paper provides a short introduction to SpaceFibre and then describes how SpaceFibre is being implemented. It introduces some SpaceFibre test equipment and explains how SpaceFibre has been validated. SpaceFibre is designed to support high data rate payload data-handling like synthetic aperture radar (SAR), multi-spectral imaging systems and fast mass memory. It is an ideal candidate for the next generation of spacecraft interconnect, being an open standard designed specifically for spacecraft applications.**

***Index Terms*—SpaceWire, SpaceFibre, Network, Spacecraft Onboard Data-Handling, Quality of Service, FDIR, Next Generation Interconnect.**

I. Introduction

SpaceFibre [1][2][3] is a very high-speed serial data-link being developed by the University of Dundee for ESA for use with high data-rate payloads. The draft SpaceFibre standard has been written by the University of Dundee for ESA and has been reviewed by the international spacecraft engineering community. It has also been simulated and implemented in several forms. SpaceFibre is currently being integrated into several third party beta test applications to help refine the standard.

The SpaceFibre standard is described in section II. The design of a SpaceFibre IP core is outlined in section III. A radiation ASIC implementation of SpaceFibre is described in section IV. Currently available test equipment and future test equipment for SpaceFibre is considered in section V. The ways in which the SpaceFibre standard has been validated are explained in section VI.

II. The SpaceFibre Standard

SpaceFibre is currently a draft standard being specified by the University of Dundee with contributions from several other organisations. The protocol stack for SpaceFibre is illustrated in Fig. 1.

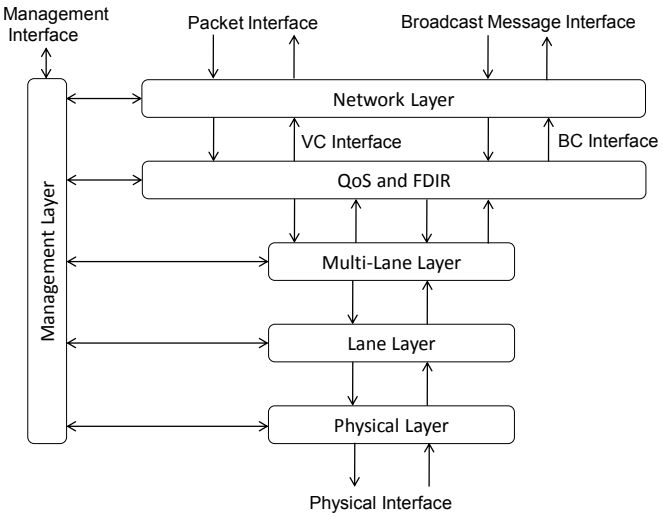

Fig. 1. SpaceFibre Protocol Stack

The network layer protocol provides two services for transferring application information over a SpaceFibre network; the packet transfer service and the broadcast message service. The Packet Transfer Service transfers SpaceFibre packets over the SpaceFibre network, using the same packet format and routing concepts as SpaceWire. The broadcast message service broadcasts short messages carrying time and synchronisation information to all nodes on the network.

The QoS and FDIR layer provides quality of service and flow control for a SpaceFibre link. It frames the information to be sent over the link to support QoS and scrambles the packet data to reduce electromagnetic emissions. It also provides a retry capability; detecting any frames or control codes that go missing or arrive containing errors and resending them.

The Multi-Lane layer is able to operate several SpaceFibre lanes in parallel to provide higher data throughput. In the event of a lane failing the Multi-Lane layer provides support for graceful degradation, automatically spreading the traffic over the remaining working links. It does this rapidly without any external intervention.

The Lane layer initialises each lane initialisation and re-initialises the lane when an error is detected. Data is encoded into symbols for transmission using 8B/10B encoding and

decodes these symbols in the receiver. 8B/10B codes are DC balanced supporting AC coupling of SpaceFibre interfaces.

The Physical layer serialises the 8B/10B symbols and sends them over the physical medium. In the receiver the Physical layer recovers the clock and data from the serial bit stream, determines the symbol boundaries and recovers the 8B/10B symbols. Both electrical cables and fibre-optic cables are supported by SpaceFibre.

The management layer supports the configuration, control and monitoring of all the layers in the SpaceFibre protocol stack.

The SpaceFibre standard has been simulated, implemented and reviewed at all stages of its research, design and development. The lane and QoS layers of SpaceFibre are fully defined and have been extensively tested with simulations by at least three independent organisations, and by implementation in FPGAs. The physical layer is well on the way to being complete with final inputs on eye pattern masks etc. to be added. The multi-lane layer has been designed and simulated, and is currently in the process of being implemented and tested in FPGAs. Once this testing is complete and the specification updated to resolve any issues found, draft G of the SpaceFibre standard will be issued and an ECSS working group will be convened to finalise the standard for formal approval.

The SpaceFibre network layer will be a separate standard document. The network layer uses the same packet format as SpaceWire and supports path and logical addressing.

III. A SpaceFibre IP Core

A SpaceFibre IP core has been designed and developed to test and validate the SpaceFibre specification. This has been updated and used to re-validate each revision of the SpaceFibre standard. A block diagram showing the interfaces to the IP Core is given in Fig. 2. The current version SpaceFibre IP core is complaint to draft F3 version of the SpaceFibre standard and supports all its features with the exception of multi-laning.

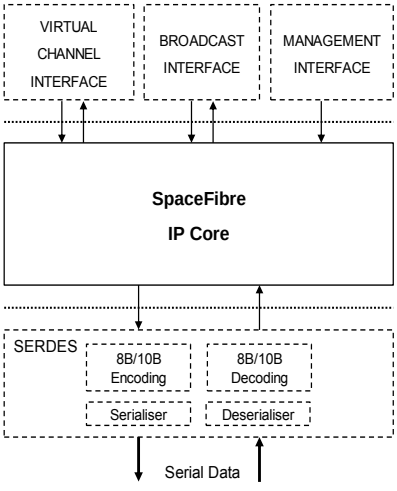

Fig. 2. SpaceFibre IP Core Interfaces

The SpaceFibre IP Core is designed to interface with an 8B/10B encoder/decoder and SerDes. This allows the SpaceFibre IP Core to be used with space qualified SerDes such as the TLK2711-SP device from Texas Instruments. The application interface to the SpaceFibre IP core comprises three separate interfaces:

1. A virtual channel interface, which is used to send and receive SpaceFibre packets over the virtual channels in the interface.
2. A broadcast interface, which is used to send broadcast messages over the SpaceFibre network.
3. A management interface, which is used to configure, control and monitor the status of the SpaceFibre interface.

The FPGA resources required for a SpaceFibre link with a single virtual channel are detailed for various types of space qualified, radiation tolerant FPGAs in Fig. 3. to Fig. 5. The utilisation for an 8 virtual channel interface is about twice that of a single virtual channel interface.

```
**********************************************************
Device Utilization for XQR4VLX200CF1509
**********************************************************
Resource                          Used    Avail   Utilization
----------------------------------------------------------
IOs                               618     -           -
Global Buffers                    0       32        0.00%
LUTs                              4299    89088     4.83%
CLB Slices                        2150    89088     2.41%
Dffs or Latches                   2525    178176    1.42%
Block RAMs                        5       336       1.49%
   RAMB16                         5
Distributed RAMs
   RAM16X1D                       67
DSP48s                            1       96        1.04%
----------------------------------------------------------
```

Fig. 3. SpaceFibre Single Virtual Channel Xilinx Virtex 4 FPGA Utilisation

```
**********************************************************
Device Utilization for XQ5VLX330TEF1738
**********************************************************
Resource                          Used    Avail   Utilization
----------------------------------------------------------
IOs                               618     -           -
Global Buffers                    0       32        0.00%
LUTs                              3331    207360    1.61%
CLB Slices                        833     51840     1.61%
Dffs or Latches                   2523    207360    1.22%
Block RAMs                        3       324       0.93%
   RAMB18                         1
   RAMB18SDP                      4
Distributed RAMs
   RAM32M                         13
DSP48Es                           1       192       0.52%
----------------------------------------------------------
```

Fig. 4. SpaceFibre Single Virtual Channel Xilinx Virtex 5 FPGA Utilisation

```
**********************************************************
Device Utilization for RTAX2000S/256CQFP
**********************************************************
Resource                          Used    Avail   Utilization
----------------------------------------------------------
IOs                               618     -           -
Modules                           7804    32256    24.19%
Sequential modules                2691    10752    25.03%
Combinational modules             5113    21504    23.78%
Global HCLKs                      0       4         0.00%
Global RCLKs                      0       4         0.00%
RAM Blocks                        9       64       14.06%
Mathblocks                        0       0         0.00%
----------------------------------------------------------
```

Fig. 5. SpaceFibre Single Virtual Channel Microsemi RTAX2000 Utilisation

The SpaceFibre IP core has been designed to support the testing of the SpaceFibre standard. It has not been designed for

speed or size. A version of the SpaceFibre IP core targeted for high performance and small size in flight qualified FPGAs is currently being developed by STAR-Dundee Ltd. This IP core is designed to support instrument interfacing with SpaceFibre using existing flight proven FPGAs and SerDes devices.

IV. A Radiation Tolerant SpaceFibre Device

A radiation tolerant SpaceFibre interface device has been developed by University of Dundee, STAR-Dundee, Ramon Chips, ACE-IC, IHP, Airbus DS and SCI within the Very High Speed Serial Interface (VHiSSI) European Commission Framework 7 project [5]. The VHiSSI chip integrates a complete SpaceFibre protocol engine, together with the physical layer interfaces, in a radiation tolerant chip manufactured by a European foundry. A block diagram of The VHiSSI device is shown in Fig. 6.

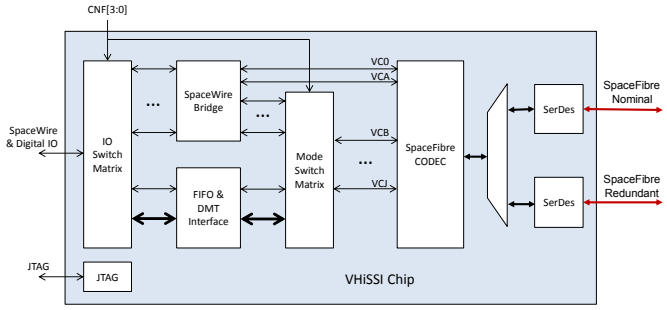

Fig. 6. VHiSSI Chip Block Diagram

There are five main functions within the VHiSSI chip:

- SpaceWire Bridge
- FIFO, DMA, Memory and Transaction Interface
- SpaceFibre Interface
- SerDes
- IO Switch Matrix
- Mode Switch Matrix

The SpaceWire Bridge provides a bridge between SpaceWire and SpaceFibre with up to 11 SpaceWire interfaces being available. The SpaceWire Bridge includes a seven port SpaceWire router which allows routing between three SpaceWire ports, three Virtual Channel (VC) buffers of the two SpaceFibre interfaces and a device configuration port. Configuration of the VHiSSI chip can be carried out over any SpaceWire interface connected to the embedded SpaceWire router or over VC0, VCA and VCB of the SpaceFibre interface. The SpaceWire Bridge is connected to the IO Switch Matrix and to the Mode Switch Matrix.

The FIFO and DMA, Memory and Transaction (DMT) Interface provides various types of parallel interface into the VHiSSI chip for sending and receiving data over the SpaceFibre interfaces. The various parallel interface functions have been designed with specific application scenarios in mind and between them are able to operate with many types of local host system, including FPGAs and processors. The parallel interface is also designed to use a small number of pins, so that the VHiSSI chip can fit into a small (100 pin) package

The SpaceFibre Interface has 11 virtual channels. VC 0 is intended primarily for VHiSSI device and local system configuration and monitoring and is connected to the embedded SpaceWire router. The other VCs have programmable VC numbers and so are referred to by letters. VCA and VCB are connected to the embedded SpaceWire router. The other VCs are connected directly to a SpaceWire interface, or to the parallel interface, depending on the mode of operation. Each VC supports full SpaceFibre QoS which can be configured independently for each VC.

VC0 and VCA are directly connected to the embedded SpaceWire router. The other SpaceFibre VC buffers are connected to the Mode Switch Matrix which connects them to either the SpaceWire Bridge or the parallel interface. The SpaceFibre interface is connected via a multiplexer to either the nominal or redundant SerDes and CML transceiver.

The SerDes converts parallel data words from the SpaceFibre interface into a serial bit stream and vice versa. On the receive side the bit clock is recovered from the serial bit stream by the SerDes. The SerDes includes integral CML transceivers.

The IO Switch Matrix connects either the SpaceWire LVDS, SpaceWire LVTTL or parallel interface signals from the FIFO and DMT interface to the digital IO pins of the VHiSSI chip. Configuration is static and determined on exit from device reset.

The Mode Switch Matrix connects either the SpaceWire Bridge or FIFO and DMT interface (parallel interface) to the VC buffers of the two SpaceFibre interfaces. Configuration is static and determined on exit from device reset.

The digital logic for VHiSSI was designed by STAR-Dundee Ltd. with system architectural design and project management being carried out by University of Dundee. AirbusDS provided inputs to the VHiSSI requirements. The back end design was carried out by Ramon Chips. ACE-IC designed the SerDes parts of the chip. Test vectors were prepared by STAR-Dundee and SCI with inputs from other partners. The chip was manufactured by IHP. The resulting VHiSSI chip is shown in Fig. 7.

Fig. 7. VHiSSI SpaceFibre Chip

Initial testing of all chips was carried out at IHP with support from STAR-Dundee and SCI. The VHiSSI chip can be seen on the right hand side of Fig. 8. mounted on a chip tester.

Fig. 8. VHiSSI in Chip Tester at IHP

The chip tester was able to carry out basic testing of the VHiSSI chip, but full-speed functional testing had to be carried out using dedicated test boards and test equipment. Four test boards were designed:

1. SpaceWire LVDS test board, for testing VHiSSI in the SpaceWire bridge mode with five LVDS SpaceWire interfaces and one LVTTL interface. This board is also being used for SEU radiation testing of VHiSSI.
2. SpaceWire LVTTL test board, for testing the SpaceWire bridge mode with eleven SpaceWire LVTTL interfaces.
3. FMC interface test board, for testing the parallel, FIFO and DMT, interface modes of operation.
4. Radiation test board, for testing the total ionising dose characteristics of the VHiSSI device.

A block diagram of the SpaceWire LVDS test board is shown in Fig. 9.

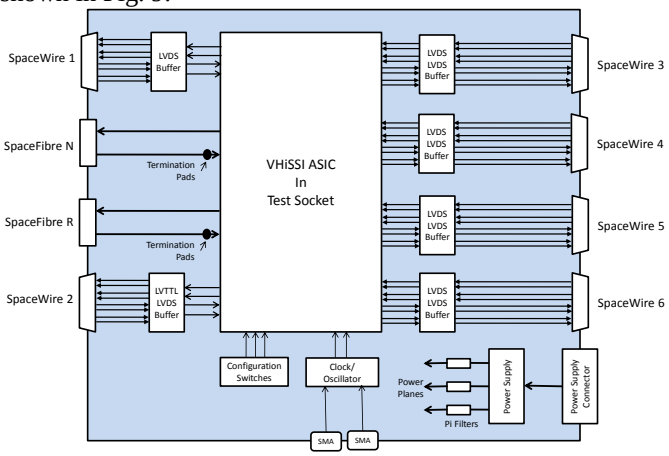

Fig. 9. Block Diagram of VHiSSI SpaceWire LVDS Test Board

The VHiSSI chip is mounted in a specially designed wide bandwidth test socket which can support the SpaceFibre 2.5 Gbits/s serial data rate. The VHiSSI chip is directly connected to the nominal and redundant SpaceFibre interfaces. The SpaceWire interfaces are connected via LVDS buffers to SpaceWire micro-miniature D-type connectors. The LVDS buffers on the board are only necessary for SpaceWire interface 1 which has an LVTTL interface to the VHiSSI chip. The others are there simply to protect the VHiSSI chip during testing since in the SpaceWire LVDS mode they have LVDS interfaces on the VHiSSI chip. A crystal oscillator, configuration switches and power supply circuitry are included on the test board. Latch up protection circuitry is also included within the power supply circuitry for the SEE radiation testing.

The VHiSSI chip was tested using a STAR-Dundee STAR Fire unit, to send and receive SpaceFibre packets and broadcast codes from VHiSSI and to monitor the link during lane initialization and error recovery operations. The STAR-Fire unit is described in section V.

The block diagram of the VHiSSI SpaceWire LVDS illustrates how simple it is to build a SpaceWire to SpaceFibre bridge using the VHiSSI chip, with very few additional components being required especially when external LVDS buffers are not used.

A photograph of the VHiSSI SpaceWire LVDS test board is shown in Fig. 10.

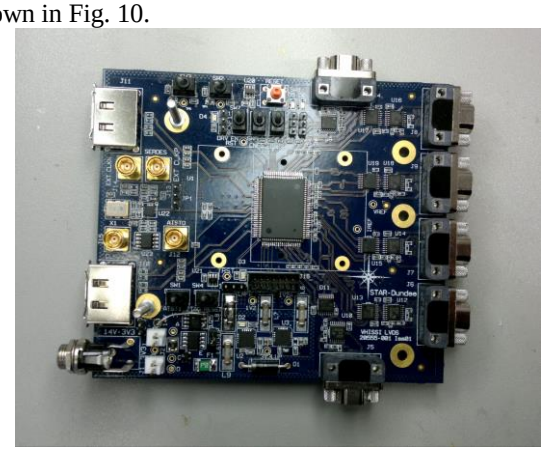

Fig. 10. VHiSSI SpaceWire LVDS Test Board

The radiation test board for VHiSSI is shown in Fig. 11.

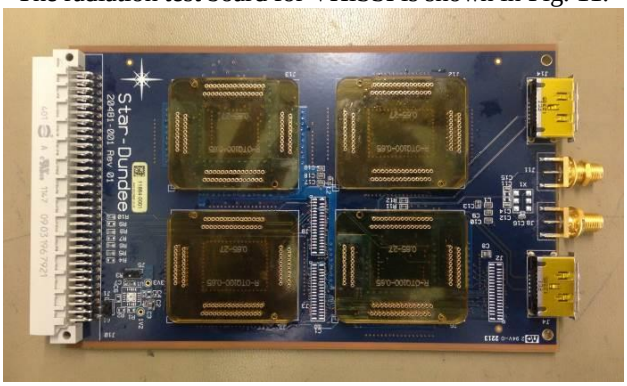

Fig. 11. VHiSSI Radiation Test Board

Four chips are tested together two powered and two not powered with one of the powered devices also clocked.

The VHiSSI chip is currently being tested in Dundee. Initial results from the testing will be available by the end of

September 2014. Radiation testing will be carried out by Airbus DS GmbH in October 2014.

V. SpaceFibre Test Equipment

STAR-Dundee has developed a range of SpaceFibre test and development equipment. The first unit, STAR Fire, was designed to support the testing of SpaceFibre and include SpaceWire to SpaceFibre bridging, pattern generation and checking for multiple virtual channels and link analysis capabilities. A block diagram of STAR Fire is shown in Fig. 12. and a photograph in Fig. 13.

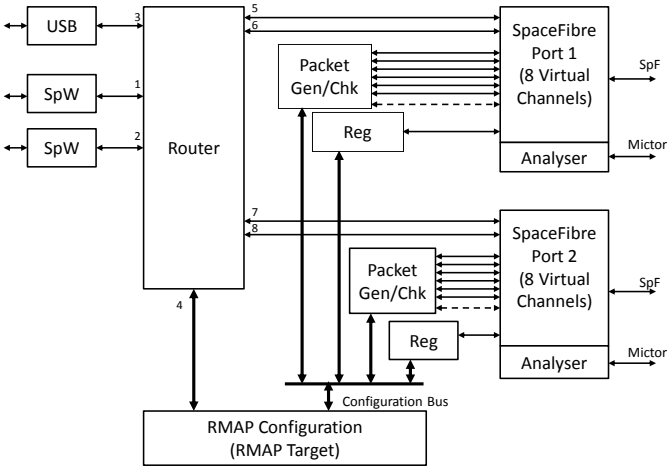

Fig. 12. STAR Fire SpaceFibre Development Kit

The STAR-Fire unit contains two SpaceFibre interface each with eight virtual channels. Two virtual channels of each SpaceFibre interface are connected to a SpaceWire router, which also has two SpaceWire ports, a USB port and an RMAP configuration port. This allows the two SpaceWire interfaces and the USB interface to send packets through either SpaceFibre interface. To test the SpaceFibre interface at full speed and to exercise and validate the bandwidth reservation, priority and scheduled qualities of service, a packet generator and checker is attached to six of the virtual channels of each SpaceFibre interface. The STAR Fire unit is configured and controlled by a Remote Memory Access Protocol (RMAP) interface attached to the SpaceWire router. This allow configuration to be performed over the SpaceWire interfaces, USB interface or the SpaceFibre interfaces. Each SpaceFibre interface has an analyser attached which can be used to record and analyse the operation of the SpaceFibre interface.

Fig. 13. STAR Fire Unit

A graphical user interface provides access to all the capabilities of STAR Fire. Part of an example analysis display is shown in Fig. 14. where the control words being exchanged in each direction are shown in colour and the four symbols that make up the left hand control code being shown in black and white.

ComaInit	LLCW	INIT3	0	INIT3	INIT2
ComaInit	LLCW	INIT3	0	INIT3	INIT2
ComaInit	LLCW	INIT3	0	INIT3	INIT2
ComaInit	LLCW	INIT3	0	INIT3	INIT2
ComaInit	LLCW	INIT3	0	INIT3	INIT3
ComaInit	LLCW	INIT3	0	INIT3	INIT3
ComaInit	LLCW	INIT3	0	INIT3	INIT3
ComaInit	LLCW	INIT3	0	INIT3	INIT3
ComaInit	LLCW	INIT3	0	INIT3	IDLE
ComaInit	LLCW	INIT3	0	INIT3	IDLE
ComaInit	LLCW	INIT3	0	INIT3	IDLE
ComaInit	LLCW	INIT3	0	INIT3	IDLE
ComaInit	LLCW	INIT3	0	INIT3	FCT +1 (1)
ComaInit	LLCW	INIT3	0	INIT3	FCT +2 (2)
ComaInit	LLCW	INIT3	0	INIT3	FCT +3 (3)
ComaInit	LLCW	INIT3	0	INIT3	FCT +4 (4)

Fig. 14. STAR Fire Analysis Display

A cPCI interface board has also been developed for SpaceFibre which is compatible with cPCI, RASTA and National Instruments PXI systems. This board can provide a number of different SpaceFibre functions including SpaceFibre interface, SpaceWire to SpaceFibre bridging and SpaceFibre Router functions. This board is expected to be available early in 2015. The STAR Fire unit is currently available from STAR-Dundee along with the SpaceFibre IP core.

VI. SpaceFibre Validation

The University of Dundee designed the lane layer of SpaceFibre with funding from ESA under the SpaceFibre contract, and the QoS and FDIR layer with funding from the European Commission (EC) SpaceWire-RT grant. The physical, multi-lane and management layers are currently being specified with ESA funding under the SpaceWire Demonstrator contract.

As SpaceFibre was being designed by the University of Dundee, various alternative designs were simulated to rapidly explore alternative designs and support design trade-offs.

In parallel with specifying the SpaceFibre standard the University of Dundee designed and tested the SpaceFibre IP core in VHDL. This was used to validate each revision of the SpaceFibre standard in a series of FPGA implementations.

To support the testing of SpaceFibre a suitable test platform was required, so STAR-Dundee Ltd. developed the STAR Fire unit. This device was used as a validation platform for the SpaceFibre IP core. Link analysis capability was included so that the exchange of information over the SpaceFibre interface could be recorded and analysed.

As the specification of the SpaceFibre standard developed formal simulations of the standard were carried out by St Petersburg University of Aerospace Instrumentation (SUAI), covering drafts C, D and E [6], and by Thales Alenia Space France, covering draft F3. These simulations identified many issues with the SpaceFibre standard which were then rectified.

NEC and Melco in Japan are both developing SpaceFibre interface devices to the specification produced by the University of Dundee. This work has provided valuable feedback on the specification and implementation of SpaceFibre.

Several ESA projects are using the Dundee SpaceFibre IP core under a Beta evaluation programme. Feedback from these beta sites has been used to improve the SpaceFibre standard and the SpaceFibre VHDL IP core and related documentation.

To raise the TRL of SpaceFibre a spaceflight engineering model is being developed by Airbus Defence and Space in the frame of the ESA SpaceFibre Demonstrator project. This design uses already flight proven components (RTAX2000 and TLK2711-SP).

The VHiSSI radiation tolerant SpaceFibre interface device was developed by University of Dundee and partners within the Very High Speed Serial Interface (VHiSSI) European Commission Framework 7 project. This device has been manufactured and is currently being tested.

Axon is working on an open specification for SpaceFibre cable and connectors, which has been referred to in the current draft specification of the SpaceFibre standard. The cables and connectors have been tested using the STAR Fire unit.

VII. Conclusions

SpaceFibre is a multi-Gigabit/s data link and network technology specifically designed for spaceflight applications. It is targeted primarily at spacecraft onboard payload data-handling applications. It includes built in, very efficient, quality of service and fault detection, isolation and recovery techniques, which simplify the use of SpaceFibre enormously; providing substantial system level benefits without requiring the implementation of complex performance limiting software protocols. SpaceFibre is backwards compatible with SpaceWire at the packet level allowing easy bridging between SpaceWire and SpaceFibre, so that existing SpaceWire devices can be incorporated into a SpaceFibre network and take advantages of its performance and QoS and FDIR capabilities.

SpaceFibre has been designed, reviewed and validated through analysis, simulation and hardware implementation, in a series of stages with feedback from each validation cycle feeding into the design. This has resulted in a mature well tested standard which will be released to ECSS for formal standardisation at the end of 2015. The TRL is already at TRL 5 with an implementation designed in flight proven radiation tolerant FPGA and SerDes devices. It will be raised to TRL 6 with application demonstrations in the near future. An experimental radiation tolerant SpaceFibre interface has been designed and manufactured and is currently undergoing tests.

Acknowledgment

The research leading to these results has received funding the European Space Agency under ESA contract numbers 4000102641 and 17938/03/NL/LvH from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement numbers 263148 and 284389. We would also like to thank Martin Suess the ESA project manager for the SpaceFibre related activities for his help, advice and guidance.

References

[1] S. Parkes, A. Ferrer and A. Gonzalez, “SpaceFibre Standard”, Draft F3 September 2013, available from http://space-env.esa.int/indico/confLogin.py?confId=32 (last accessed 15th Feb 2014).

[2] S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, “SpaceFibre: Multiple Gbits/s Network Technology with QoS, FDIR and SpaceWire Packet Transfer Capabilities”, International SpaceWire Conference, Gothenburg, June 2013.

[3] S. Parkes, “Never Mind the Quality, Feel the Bandwidth: Quality of Service Drivers for Future Onboard Communication Networks”, paper no. IAC-10.B2.6.6, 61st International Astronautical Congress, Prague 2010.

[4] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 1, European Cooperation for Space Data Standardization, July 2008.

[5] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, R. Ginosar, T Liran, G Sokolov, G Burdo, N Blatt, P Rastetter, M Krstic, A Crescenzio, “A Radiation Tolerant SpaceFibre Interface Device”, International SpaceWire Conference, Gothenburg, 2013.

[6] V. Olenev, I. Lavrovskaya, and I. Korobkov, “SpaceWire-RT/SpaceFibre Specification and Modelling”, International SpaceWire Conference, Gothenburg, 2013.

SpaceFibre/SpaceWire-RT implementation experience and evolution trends

SpaceFibre, Long Paper

Elena Suvorova, Irina Lavrovsakaya, ValentinOlenev, YuriySheynin
Saint-Petersburg State University of Aerospace Instrumentation (SUAI)
Saint-Petersburg, Russian Federation
sheynin@aanet.ru, suvorova@aanet.ru

***Abstract*—**

The paper considers experience of SpaceFibre/SpaceWire-RT implementation and analysis. Overheads are estimated and gaps in the SpaceFibre draft for implementation are considered, its refinement is proposed.

The problems in the Retry level are discussed. In case of repeating a frame the virtual channels characteristics (priorities, timeslots) are not fully followed; it could corrupt required QoS of the traffic. Another problem with the Retry level is shown in a case study of streaming data traffic. For such type of traffic late frames transmission (could be not only useless but harmful for the target traffic. To deal with the problems a modification of Retry is proposed.

Another open problem is in disconnection of a link at the Lane and Encoding levels when there is a need to change virtual channels (VC) logical numbers. When one needs to switch on or change a logical number of a single VC the link disconnection and restart is done. Thus data transfer would be stopped for all the VC of the link. It leads to considerable delays in tuning logical structuring of networks, excessive delays in virtual channel transfers when other VC in the same controller are retuned, excessive complication and resources in processor-less nodes implementations. To deal with the problem a modification of link flow control crediting is proposed.

I. Introduction

The SpaceFibre standard supports several quality of service (QoS) classes:

- guaranteed packet delivery;
-priorities;
-guaranteed throughput;
- scheduling;
- best effort.

Support for these classes of service is provided in the data link at the QoS layer. (The SpaceFiber protocol stack is shown in fig. 1). The QoS layer includes the Virtual channels sublayer, the Framing sublayer and the Retry sublayer, fig. 2.

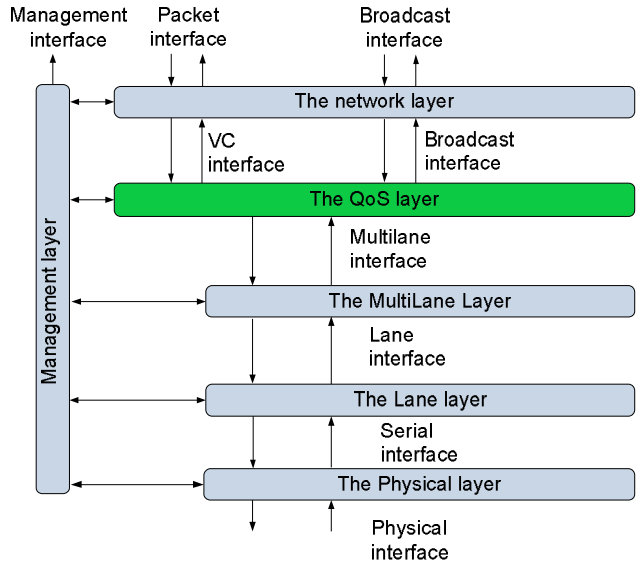

Fig.1. The SpaceFibre protocol stack

However, there are some problems and requirements, which are not supported in the current SpaceFibre standard draft:

- problem of fulfillment the data flows characteristics (priority, reserved bandwidth, timeslots list for scheduling) in case of frames retransmission;
- no packets transmission without guaranteed delivery for some types of traffic;
- no reconfiguration of virtual channels logical numbers without disconnection on the Lane and Encoding layers.

In some cases a single error that occurs in the channel does not lead to breaking the connection at the Lane and the Encoding layer. Currently specified in the SpaceFibre procedure of frames retransmission may violate the QoS characteristics and constraints that are specified for data flows characteristics. At the VC layer it can lead to quite noticeable delays of high-priority traffic transmission, and, when scheduling is used, to frames transmission out of the assigned timeslots.

For most types of streaming data traffic there is no requirement of guaranteed delivery, for example for video traffic. Loss of some data in this traffic is not critical for system functionality as a whole. In typical videostream networking (e.g. ARINC 818) frame fragments with errors are not repeated; it is neither needed, nor possible for conventional equipment to implement. Created by this traffic network load is very high. The retransmissions of such traffic may lead to network congestions, impossibility to deliver other traffic with the QoS constraints to destination terminal nodes, e.g. command traffic, which transmission is much more critical to the system functionality.

Reconfiguration of virtual channels logical numbers may be necessary due to changes in the network operation mode, to tasks and applications migration between nodes, to new equipment attachment, etc. In SpaceFibre, to reconfigure VC logical numbers one needs to disconnect the link. Disconnection leads to data transmission impossibility via all the other VCs of the data link during reconfiguration of the single VC and connection recovery (about 50 us). This may result, for example, in very essential delay of command traffic.

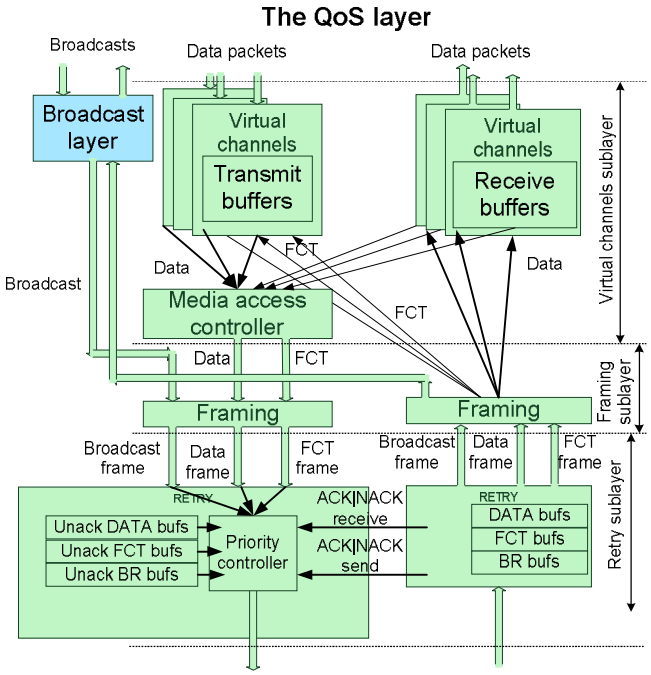

Fig. 2 The QoS sublayer structure

II. The QoS layer of the SpaceFibre standard

The QoS layer includes the Virtual channels sublayer, the Framing sublayer, the Retry sublayer, fig. 2.

A. *The Virtual channels sublayer*

The main functions of the Virtual channels sublayer are:

- the data flow control functions (credit based mechanism);
- partitioning of packets into data blocks, each of which is further placed in a separate frame;
- support of priority mechanism;
- providing of guaranteed throughput;
- scheduling.

The Virtual channels sublayer has a separate buffer space for every virtual channel (VC), which includes the buffers for storing the transmitted data and the buffers for storing the received data. The Destination VC sends to the Source VC information about the available buffer space. The Source VC may send amount of data that corresponds to this free buffer space.

The Virtual channel partitions data into blocks before transmission to the Framing sublayer that will place them into separate frames.

Transmission of the next data block may start if the Virtual channel transmission buffer contains the end of packet or contains 256 Nchars (the maximal size of frame data field).

The media access controller, fig. 2, arbitrates requests for data transmission from the virtual channels. The arbitration is done in correspondence with the priority levels, amount of allocated bandwidth and scheduling that are assigned to every VC.

B. *The Framing sublayer*

The main functions of the Framing sublayer:

- packing data blocks, FCT, Broadcasts to frames for transmission and extraction of data blocks, FCT, Broadcasts from received frames;
- scrambling and descrambling.

C. *The Retry sublayer*

The main functions of the Retry sublayer are:

- junction for transmission of data frames, Broadcast frames, command frames correspondingly to their priorities;
- separation of the received frames into the data frames flow, Broadcast frames flow, command frames flow;
- implement mechanisms for guaranteed delivery.

To ensure guaranteed delivery, at the Retry sublayer on the source side:

- A serial number and CRC are assigned to every frame.
- A frame is stored in the retry buffer for transmitted and unacknowledged frames; the frame is stored in this buffer until ACK will be received.
- When ACK is received, all the frames with sequence numbers less or equal to the specified in the ACK are considered successfully transmitted and are deleted from retry buffer.
- When NACK is received all the frames with sequence numbers more than the specified in NACK number should be retransmitted (sequence

numbers of the frames could be changed before retransmission).

To ensure guaranteed delivery on the destination side:

- the Retry sublayer checks the sequence number, CRC and structure of the received frame;
- if there are no errors in the frame the ACK (acknowledgment of success) is sent;
- if an error is found, the NACK is sent.

If the frame has been received without errors it is transferred to the Framing layer and then pass to the Virtual channels layer (or to the Broadcast layer).

The problem of data frames retransmission in correct QoS sequence

The data flow, FCT flow and broadcast flows are distinguished at the Retry layer (the priority levels of them are kept when they are retransmitted). However, the data frames of different virtual channels are not distinguished at the Retry layer. Serial numbers of frames are generated at the Retry layer and the frames are placed in the buffers, fig.2. If a NACK is received, the unacknowledged data frames will be retransmitted in the order, in which they happened to be placed in the Retry data buffer. The assigned priorities, scheduling, allocated bandwidth in this case are not accounted by the Retry layer. It can violate QoS parameters of data frames from different VC.

The fig. 3 shows dependency between the frame retransmission time and quantity of retransmitted frames that have been placed in the Retry buffer before this negatively acknowledged frame. The frame retransmission time grows proportionally to the quantity of previous frames in the Retry buffers.

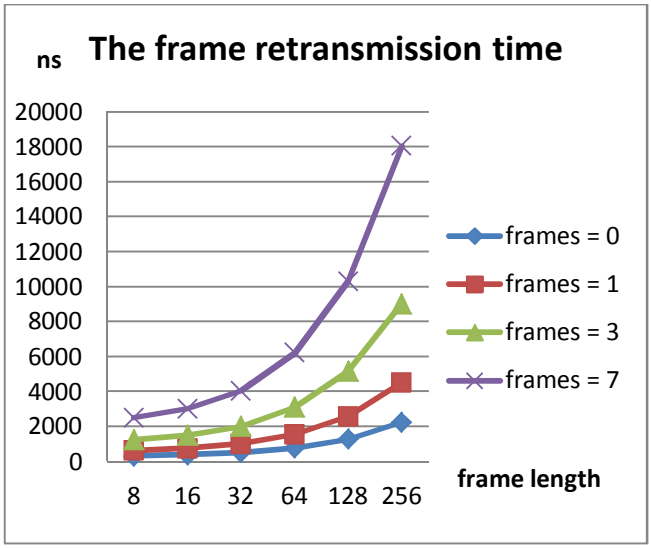

Fig.3 The dependency between the frame retransmission time and quantity of retransmitted frames that are placed in the Retry buffer before this frame

The Retry sublayer provides the guaranteed delivery service class by retransmission of all corrupted or lost data frames. This retransmission mechanism cannot be turned off for some data flows, for which the guaranteed delivery is not required.

We suggest some modifications of the QoS layer for provision of:

- rearranging frames processing between the sublayers of the QoS layer to take into account the frames' QoS attributes and parameters when the frames are retransmitted;
- additional transmission mode without guaranteed delivery for some data flows;
- resetting and reconfiguration of a single VC without interruption of information flow in other VCs.

III. The Modifications of the Virtual Channels and the Retry Sublayers

At the Retry sublayer there is no information about the required characteristics of frames, their QoS attributes and parameters. Therefore in frames retransmission it is not possible to apply processing algorithms for QoS support at this layer. We suggest moving the retransmission functionality from the Retry sublayer to the Virtual channels sublayer and the Broadcast layer. At these layers retransmission of frames may be arranged in accordance with their priorities, the list of timeslots, allocated bandwidth. The modified QoS protocol sublayers structure is represented in fig. 4. The suggested modifications:

- separate numbering of data frames and FCT frames;
- separate numbering of data frames and Broadcast frames;
- separate the acknowledgement and retransmission for different Virtual channels, for Broadcasts channels;
- move the acknowledgement and retransmission schemes d from the Retry to the Virtual channels sublayer;
- move the acknowledgement and retransmission for Broadcast frames from the Retry sublayer to the Broadcast layer.

Within the suggested approach the receiver interprets the frame numbers with reference to the type of flow (Data, Broadcast, FCT) and for the Data and FCT flows – with the reference to the logical VC number also. Some functions of the Retry sublayer go from the Retry sublayer to the VC sublayer and Broadcast sublayer:

- data frames and FCT sequence numbers will be generated and controlled by the VC sublayer (;
- sequence numbers of Broadcast frames will be generated and controlled by the Broadcast sublayer;
- ACK and NACK frames are generated separately: for the Data and FCT flows they are generated by the VC sublayer, for Broadcasts - by the Broadcast sublayer;

To support these new features:

- include additional information in the ACK, NACK and RETRY frames – the attribute of belonging the ACK/NACK to the data flow, FCT flow or Broadcast flow;
- include the VC number in Data and FCT frames;
- include in the FULL frame the attribute of belonging to data flow, FCT flow or Broadcast flow, for the Data and FCT flows the VC number also;
- frames, which belong to a data flow without guaranteed delivery (No-retry frames), should be transmitted with the sequence number 0.
- IDLE frames should be transmitted with the sequence number 0.

The ACK, NACK, RETRY FULL commands should be transmitted between the VC and the Retry sublayers via the Framing sublayer, fig.4.

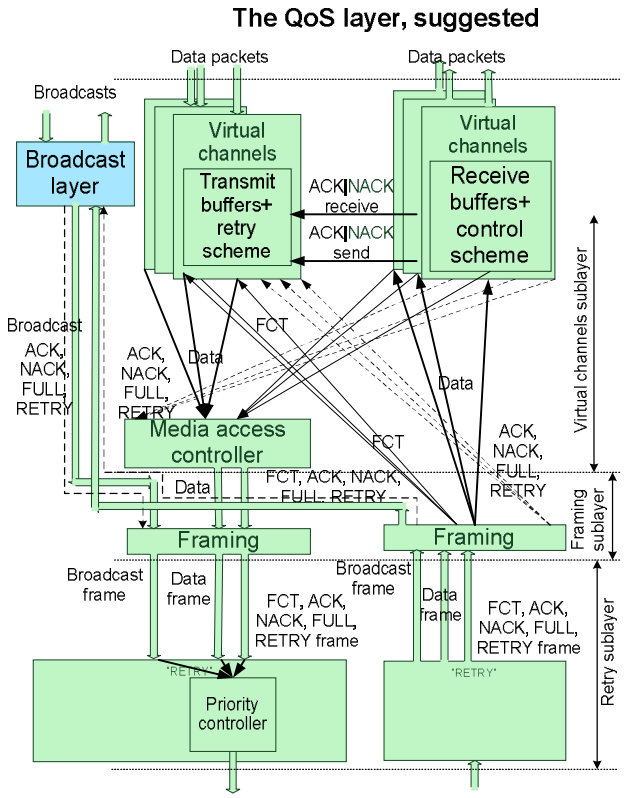

Fig . 4 The suggested variant of the QoS sublayer structure

In the modified sublayers the frame retransmission can be organized only for data flows with the guaranteed delivery requirement. If guaranteed delivery is not required for a data flow, its frames are not checked in the receiver of the Retry sublayer; the receiver does not send acknowledge for such frames. For these frames the frame sequence number is not used and we recommend setting it to 0.

IDLE frames do not belong to any flow. If an error occurs when an IDLE Frame is received, it does not cause frames retransmission for any flow. Therefore the serial number of IDLE frames is not used in the receiver; we recommend to set its value to 0.

We propose to change the format of the ACK, NACK, RETRY and FULL frames. These frames are generated not by the Retry sublayer (as in the basic variant) but by the VC sublayer and by the Broadcast sublayer. We add to them information about flow type (Broadcast, FCT, Data) and the VC number (for Data and FCT flows), fig. 5-8.

COMMA	ACK	SEQ_NUM	FLOW_TYPE
VC_NUM	ACK_NACK_type =ACK	D0_0	CRC

Fig. 5 The ACK frame format

COMMA	ACK	SEQ_NUM	FLOW_TYPE
VC_NUM	ACK_NACK_type= NACK	D0_0	CRC

Fig. 6. The NACK frame format

COMMA	RETRY	FLOW_TYPE	VC_NUM

Fig. 7 The RETRY frame format

COMMA	FULL	SEQ_NUM	FLOW_TYPE
VC_NUM	D0_0 (reserved)	D0_0 (reserved)	CRC

Fig. 8 The FULL frame format

The encoding of the FLOW_TYPE field:

D0_0 – Broadcast;
D0_1 – FCT;
D0_2 - Data.

Other possible values are reserved.

The suggested QoS sublayers modification positively effects implementation complexity. The transmitter buffers of the Retry sublayer are not needed due to suggested modification (its role play the buffers on the VC layer). Therefore hardware cost of an implementation is decreased by 15 – 20% (Concrete value in general case depends on the Retry buffer size in the basic variant of an implementation.)

Frame numbering separation from the junction controller of data, broadcast and command flows, which includes priority control, allows decreasing of ratio between the QoS layer operating frequency and data transmission frequency in the serial channel. It is important parameter if a SpaceFibre port is implemented, for example, in 180 nm technology. Such technologies are actively used today for aerospace equipment when thinner design rules do not meet space operation requirements.

Implementation of suggested mechanisms leads to some additional channel throughput overheads. To estimate

overheads assume that transmission of one frame (the maximal size one) corresponds to transmission of one FCT and one ACK in the opposite direction. Overheads of FCT and ACK transmission in SpaceFibre are about 3% of the channel throughput. In the modification, the length of ACK (NACK) and FCT grow in 2 times, so maximal overhead of its transmission grows from 3% to 6% of the channel throughput.

For data without guaranteed delivery requirement the suggested algorithms allow to decrease data link transmission time. Let the ratio between QoS processing frequency and data channel transmission frequency be 1/10. The ratio between frame transmission time without CRC control and with them is represented on fig. 8. These charts show that timing gain is 10 - 15% when frames length is 64 - 256 bytes; elimination of unnecessary for this traffic frames retransmission gives additional gain also.

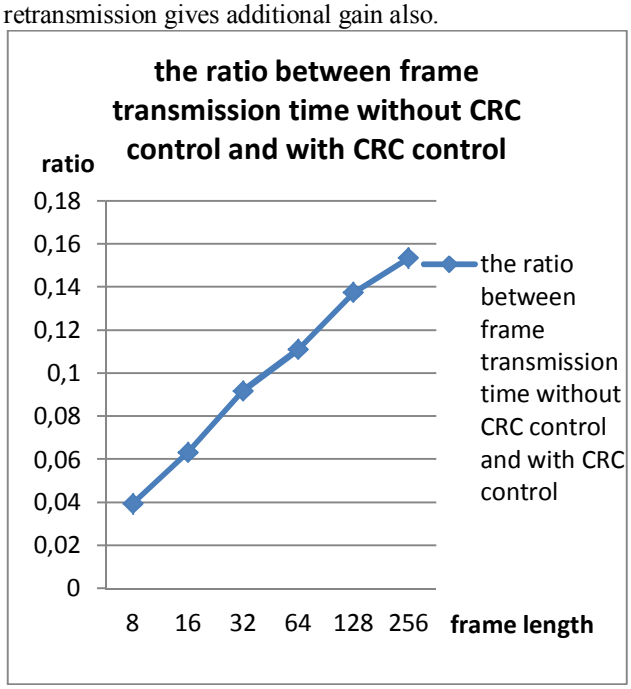

Fig. 9 Ratio between frame transmission time without CRC control and with CRC control

Implementation of the suggested mechanisms allows reducing transmission delay of high priority frames or frames that should be transmitted in specific timeslot, in case of channel errors.

The charts of the ratio between high priority packet transmission time with current SpaceFibre and with the suggested modification used are represented in fig.9. In this investigation we supposed that the high priority packet length is 64 bytes. We assume that transmission error does not cuase disconnection at the Lane or Encoding layer. Different quantities of frame buffers at the Retry layer: 2, 4, 8; and the frame length in the buffers from 8 to 256 Bytes, are considered. These charts show that when frames in the Retry buffers are short and quantity of retry buffers is small the proposed method allows to reduce time in 2 – 4 times. If the quantity of buffers is big and frames are long enough timing gain goes up to 18 times.

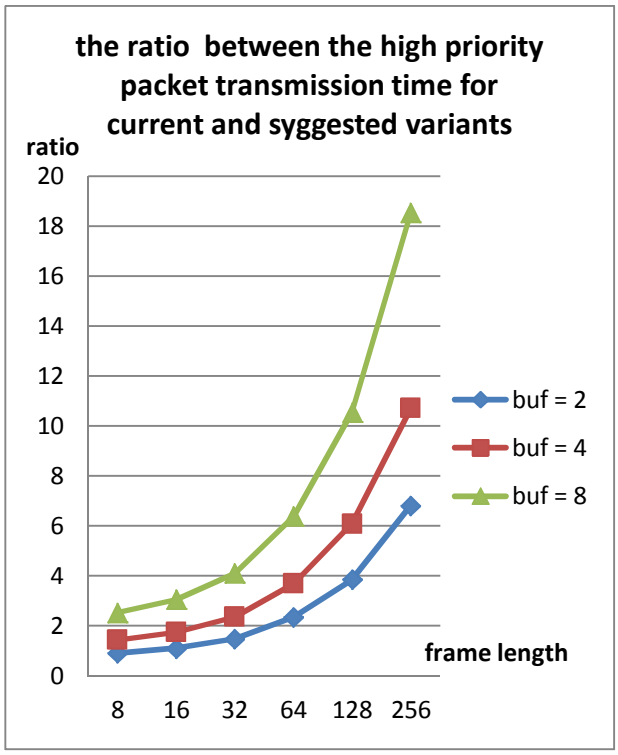

Fig. 10. Ratio between high priority packet transmission time with the current SpaceFibre retransmission mode and with the suggested modification

IV. Logical Numbers Reconfiguration and Reset for Separate VC without Disconnection

User may need to change logical numbers of a VC in a SpaceFibre network reconfiguration for starting new applications in the network nodes or migration of applications between nodes, adding new devices, changing of a node equipment operation mode, etc.

In the current SpaceFibre draft it is not possible to reset a single VC. For example in case of a VC data buffer overflow (such event is considered in standard) the user should reset all virtual channels. Thus data transmission for others VC of this data link will delay until connection recovery (about 50 us).

In the current SpaceFibre standard draft any changes of VC logical numbers without the connection break is impossible. The new VC logical numbers on different sides of the connection will be configured with some variance in time. As a result, one side can start FCT sending before another side is configured; these FCT will be rejected as invalid. So if we need to reconfigure a single VC, we should break connection at the Lane layer. The connection recovery time is about 50 us. For this time data transmission from all the other VC of this data link will be impossible. This delay can significantly affect parameters of the data flows transmitted via other VC, may violate special requirements and QoS constraints for these VC.

To deal with the problem we suggest:

- to slightly change the FCT sending rule: after the VC logical number configuration the VC should send only

one FCT; all other FCT should be send only after receiving of one FCT from the other side of the channel (the first FCT is used not only for credit but for connection establishment on the VC sublayer);

- add the new reset command– "Single VC Reset"; this command is analog to the flush command, but it resets only one VC – the VC, for which user needs to reset or to change logical number.

This modification of the FCT transmission mechanism affects only the VC sublayer. We propose to introduce an additional command for reset of a single concrete VC (VC_reset) to solve this problem. For this command coding could be the KCode that in current SpaceFibre standard draft is used for NACK command, but in our modification, which is described in the previous section, it is not needed more for this purpose. When the VC_reset command is received, the control logic of this virtual channel only is set to the initial state.

Fig. 11 The format of VC_reset command

Implementation of this mechanism requires some modifications at the Retry and Framing layers also, because the command should go via these layers. Processing of this command should be done at the Virtual channels layer.

Implementation of the suggested mechanisms requires little hardware cost (less than 1% of the SpaceFibre port implementation). It does not lead to additional overhead of data channels bandwidth.

V. CONCLUSION

The suggested modifications of SpaceFibre standard draft can extend its functionality and improve implementation and application.

Transfer of all functionality for data QoS support to the Virtual channels sublayer allows to retransmit frames in full correspondence with the QoS traffic parameters (such as priority, scheduling). AS we show, it isn't ensured fully in the current SpaceFibre draft. This modification allows also decreasing hardware cost of an implementation due to removing buffers from the Retry sublayer. Changing the frames numbering scheme allows to decrease ratio between the QoS processing frequency and the serial link frequency. It is useful when coarse design rules are used for implementation.

The modification enables to retransmit only traffic with guaranteed delivery requirement also. For other types of traffic retransmission could be switch off, along with overheads for its implementation. It efficiently supports streaming data traffic and improves network useful bandwidth. Adding of the service class without guaranteed delivery allows to reduce transmission time, to exclude retransmission of data frames in case of disconnections and thereby decrease network load after connection recovery, and, accordingly, decrease delivery time for other traffic.

Provision of dynamic reconfiguration of a single virtual channel number and a separate reset for a single virtual channel without disconnection at the Lane and the Encoding layers provide possibility of dynamic reconfiguration for some data flows without distortion of other data flows.

REFERENCES

[1] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements. SpaceFibre Specification. Draft F3, September 2013

QoS in SpaceFibre and SpaceWire/GigaSpaceWire Protocols

SpaceFibre, Long Paper

Nadezhda Matveeva, Yuriy Sheynin, Elena Suvorova
Saint-Petersburg State University of Aerospace Instrumentation
SUAI
Saint-Petersburg, Russian Federation
nadezhda.matveeva@guap.ru, sheynin@aanet.ru, suvorova@aanet.ru

Abstract—Nowadays SpaceWire, SpaceFibre, GigaSpaceWire protocols are widely used in spacecraft design. SpaceWire is established as one of the main standards for data transmission. It is used in many Russian, European, American and Japanese spacecraft. SpaceFibre is a newly emerging standard for the SpaceWire technology standards family, which is able to operate over fiber-optic and copper cable and supports data rates up to 2 Gbit/s. GigaSpaceWire link specification is also developed for SpaceWire technology extension. It provides gigabit link technology with longer distances and galvanic isolation capability for SpaceWire networks.

Quality of service (QoS) becomes important network characteristic for prospective onboard networks. There are various approaches for QoS provision in networks. Some of them provide QoS at every data link and node inside the network, some provide QoS features at the network boundary, in its terminal nodes, some combine these approaches in a way.

The SpaceFibre follows the first approach. In every data link it has QoS services, providing priorities, guaranteed bandwidth, guaranteed data delivery, scheduled frames transmission. Implementation of these mechanisms is associated with additional overhead such as frame transmission delay, transmitting overhead information such as header and end of frame, traffic planning and dispatching, retransmission in every data link, etc. These factors lead to increasing overheads and packet transmission time, to useful bandwidth degradation.

For SpaceWire/GigaSpaceWire the second approach for QoS provision is evolving. QoS services can be implemented over the basic SpaceWire/GigaSpaceWire network interconnection, e.g. at the Transport layer, with much more economical implementation and overheads.

In the article we analyze both approaches, their feasibility and value of QoS in SpaceWire/GigaSpaceWire and in SpaceFibre networks. Networks with different topologies and traffic pattern are used to study and to evaluate the performance. Various traffic types such as the data packets, streaming data, commands will be transmitted in networks. Data delivery characteristics for SpaceFibre and SpaceWire/GigaSpaceWire networks are analyzed and compared.

Index Terms—SpaceFibre, SpaceWire, GigaSpaceWire , Quality of Service (QoS)

I. INTRODUCTION

Let us see what features for QoS have SpaceFibre, SpaceWire and GigaSpaceWire.

The SpaceFibre standard, [1], supports several classes of service at the data link layer:

- priority;
- guaranteed throughput;
- guaranteed packet delivery;
- scheduling;
- best effort.

The QoS (Quality of Service) layer of the SpaceFibre standard provides these services. Its sublayer - the Virtual channels sublayer, realizes priority, guaranteed throughput, scheduling and best effort classes of service functionality.

For each virtual channel (VC) a priority level may be assigned. When some virtual channels have data to transmit, data from the VC with the highest priority will be sent first. Unique priority level can be assigned to every VC or one priority level may correspond to some VC.

For each VC amount of bandwidth that it can use can be defined. There is a bandwidth credit counter for every virtual channel. If the VC does not transmit any data the bandwidth credit counter is incremented. If the VC transmits some data, the bandwidth credit counter is decremented (wherein takes into account amount of transmitted data and defined amount of bandwidth for this channel). If some virtual channels with the same priority level have data to transmit, first come data from the VC with the largest bandwidth credit counter value.

Another QoS mode uses scheduled frames transmission. For every virtual channel a list of timeslots, in which it can transmit data, can be defined. . Request from the VC for data transmission during other timeslots are blocked. It gives guaranteed delivery latency for VC traffic.

The SpaceFibre standard draft makes the Retry layer responsible for guaranteed data delivery service. This layer checks correctness of the received frames and retransmit frames that have been transferred with errors or lost. For these mechanisms each frame includes a sequence number and the checksum (excluding IDLE frames).

Mechanisms of constrained priority are supported in the SpaceWire/GigaSpaceWire standards. Mechanisms to support other classes of service are not provided by the core SpaceWire/GigaSpaceWire networks. However mechanisms to support different service classes may be implemented on top of these standards, at the Transport layer especially. In this paper we consider only variants that do not require implementation of some special functions in routers for SpaceWire, [2], and GigaSpaceWire, [3]:

- priorities (at the Network layer);
- the guaranteed packet delivery between the source and destination terminal nodes (at the Transport layer);
- the scheduling mechanism for providing constrained data packet delivery time (at the Transport layer).

In the paper we consider and compare:

- features and characteristics that could be provided by the priority mechanism in SpaceWire /GigaSpaceWire networks and in SpaceFibre networks;
- mechanisms of guaranteed packet delivery that is based on an acknowledgement scheme between data source and destination terminal nodes in SpaceWire/ GigaSpaceWire and retransmission mechanism in data links in SpaceFibre .
- scheduling mechanisms for guaranteed data packet delivery time for SpaceWire/GigaSpaceWire networks and in SpaceFibre data links.

II. PRIORITY MECHANISMS FEATURES

In the SpaceFibre standard draft each VC may have its own priority level. The priority level affects frames transmission order from different virtual channels to the link. The frame for transmission is selected according to its priority value. If transmission of a lower priority frame has started before the higher priority frame arrival, then higher priority frame waits until the lower priority frame transmission is finished. The SpaceFibre standard does not use frame transmission interruption. Therefore high priority frame waiting time is up to maximum length frame (256 Nchar) transmission time plus time overheads.

In SpaceWire/GigaSpaceWire a priority level can be specified for packets at the Network layer. Priority level is associated with the packet network address (logical, regional-logical). The priority level affects packet transmission order to the output port. When transmission of a packet with lower priority is started before the packet with higher priority has arrived, the higher priority packet is transmitted after completion of the lower priority packet transfer; SpaceWire/GigaSpaceWire do not use packet transmission interruption. Therefore high priority packet waiting time depends on the lower priority packet length.

The SpaceWire standard does not limit packet length and in a general case we can't estimate the high-priority packet delay in a hop. Its waiting time depends on data formats used in a specific network. If we limit maximum packet length in the SpaceWire network, we can obtain reliable estimates of high priority packets waiting time.

To estimate transmission characteristics of high priority traffic in SpaceFibre and SpaceWire/GigaSpaceWire networks consider dependency of high priority packet transmission time in one router from low priority packet size. This dependency is presented in Fig. 1; the high priority packet length is 64 bytes.

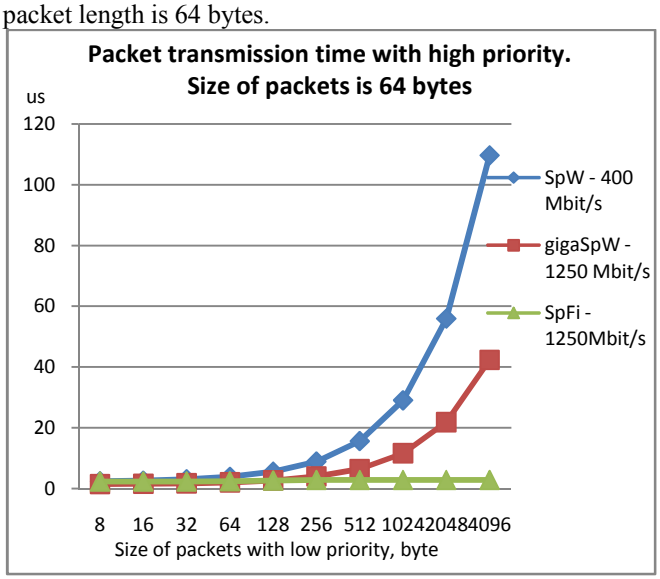

Fig. 1. Dependency of the high priority packet transmission time in one router from low priority packet size. Data rate: 400 Mbit/s in SpaceWire; 1250 Mbit/s in GigaSpaceWire; 1250 Mbit/s in SpaceFibre

The SpaceFibre provides priority level at the Frame layer. Its dependency on the Fig. 1 looks almost like straight line parallel to X axis with value 2784 ns.

High priority packet transmission time in one router for SpaceWire and GigaSpaceWire practically coincides with high priority packet transmission time in one router for SpaceFibre when low priority packet size is less than 256 bytes; after it high priority packet transmission time in one router for SpaceWire and GigaSpaceWire significantly grows.

The results show that high priority packet transmission time for SpaceWire and GigaSpaceWire networks may be close to the value for SpaceFibre networks if low priority packets size would be limited to 256 bytes. This can be achieved by appropriate fragmentation on the Transport layer in terminal nodes.

Now let us consider dependency of low priority message transmission time from the packet size. Charts for this dependency are presented in Fig. 2.; data rates are 250 Mbit/s and 312 Mbit/s.

The Figure 2 shows that message transmission time is almost the same when it is transmitted as one packet and by several packets with the size of 256 Nchar. So packets fragmentation practically doesn't worsen the message transmission time.

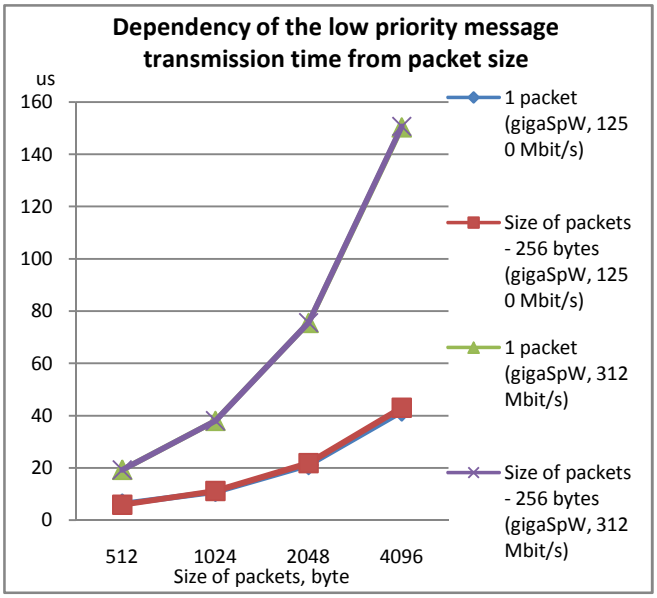

Fig. 2. Dependency of the low priority message transmission time from the packet size.

Thus for traffic with different priorities in SpaceWire/GigaSpaceWire networks practically the same transmission characteristics as in SpaceFibre can be achieved if packet's data field length would limited to 256 Nchar. It can be implemented at the Transport layer.

From the functional point of view SpaceWire/GigaSpaceWire networks are more flexible in packet priority mechanism than the SpaceFibre. In them a priority is assigned to logical and regional-logical addresses. Thus different priorities can be assigned for dozens and hundreds of packet streams in a network. In SpaceFibre priorities are assigned to a virtual channel in a data link. While in theory there could be 256 VCs in a data link, due to high hardware overheads for a VC implementation their number in a link would be limited by quite several ones (4-8 VC as an optimistic estimation). Thus only 4-8 data packet streams may have particular priorities in the entire network.

III. PACKET DELIVERY MECHANISMS

The guaranteed delivery in SpaceFibre is ensured by checking the frames transmission correctness in every data link at the Retry layer. Transmitted with errors or lost frames are retransmitted.

From a functional point of view both options allow to ensure guaranteed delivery of a packet. Difference is in where retransmission is organized – at every data link or at the network boundary, in terminal nodes. These options may have different timing characteristics and hardware costs.

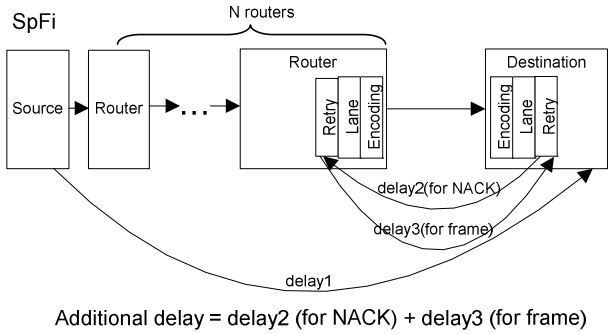

Fig. 3. The illustration of retransmission scheme in SpaceFibre data link layer and corresponding delays

The SpaceWire and GigaSpaceWire do not provide mechanisms for guaranteed packet delivery in a data link. But mechanisms for guaranteed packet delivery can be implemented in terminal nodes, at the Transport layer (the RMAP protocol is an example). Such protocol can include mechanisms for identification of packets that are lost during transmission (for example by sequence numbers), for identification of packets with errors (for example by CRC), for data packets acknowledgement and retransmission of unacknowledged packets (either not been confirmed or timed out in acknowledgement waiting).

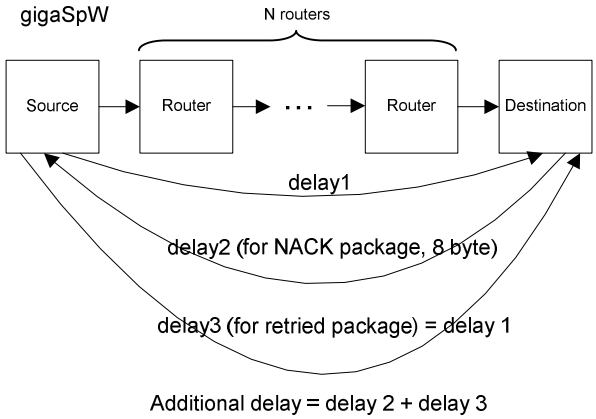

Fig. 4. The illustration of retransmission scheme between source and destination node in SpaceWire /GigaSpaceWire network and corresponding delays.

To compare timing characteristics we assume that one error occurs during the packet transmission. In a SpaceFibre network it cause a frame retransmission in the data link. In this case additional transmission time consists of the NACK transmission time and the retransmission time of the frame in the link. We assume that time of NACK formation and time of it operation at the Retry level is negligible.

In SpaceWire/GigaSpaceWire networks an error will cause a full packet retransmission from the source node. In case when one error occurs during the packet transmission, packet retransmission time depends on

- communication protocol organization;
- rules of error detection;
- timeout mechanisms and timeout values ;

Timeout values depend on network size, traffic characteristics, transmission paths of different traffics, which intersect with the considered traffic.

Let us first take the case when a packet is delivered to destination node but contains errors due to errors during transmission. In this case total time of packet delivery is calculated as the sum of transmission between source and destination time, time of packet checking, time of NACK transmission from the receiver to the transmitter and the repeated transmission time. We can assume that the packet checking time is negligibly small as it can be performed on-the-fly during packet receiving.

Dependencies between packet delivery time and number of transient routers in SpaceFibre and GigaSpaceWire networks for the one error case are presented in Fig. 5. These results are based on calculations and modeling. Plots are drawn for different packet sizes: 16, 64, 256, 1024 and 4096 Nchar. The Acknowledge packet size for a SpaceWire/GigaSpaceWire network is equal to 8 Nchar.

As one can see from the Fig. 5, the delivery time of short packets (up to 256 bytes) for a GigaSpaceWire network is less than for SpaceFibre. Further, with increasing the size of packets and with a small number of transit routers, delivery time is better for SpaceFibre. It is interesting that with increasing packet size, the number of transit routers from which SpaceFibre is better also increases. When size of packets is 1024, SpaceFibre is better when the number of routers ≤ 3. When size of packets is 4096, SpaceFibre is better after the number of routers ≤ 8.

Now consider the situation where error occurred during the packet transmission and it led to link disconnection.

In SpaceFibre the time to restore connection is 50 us after a disconnection error occurs. In this case retransmission time is increased by sum of time to restore the connection and duration of the noise.

In SpaceWire and GigaSpaceWire the time to restore connection is 19,2 us after a disconnection. To evaluate packet transmission time in this case, we assume that the acknowledgment waiting timeout (Tout) consists of packet transmission time between the source and the destination and of the NACK transmission time.

Denote sum of packet transmission time between the source and the destination and NACK transmission time as Tf. We evaluate characteristics when acknowledgment waiting timeout is equal Tf, 2*Tf and 3*Tf. We consider the worst case when disconnection happens in the final link.

There are two variants:

- Sum of noise duration and the connection restore time is less than the sum of the acknowledgment waiting timeout and the repeat packet transmission time up to the link, in which the disconnection happens. In this case packet delivery time consists of the acknowledgment waiting timeout and repeat packet transmission time between the source and the destination.

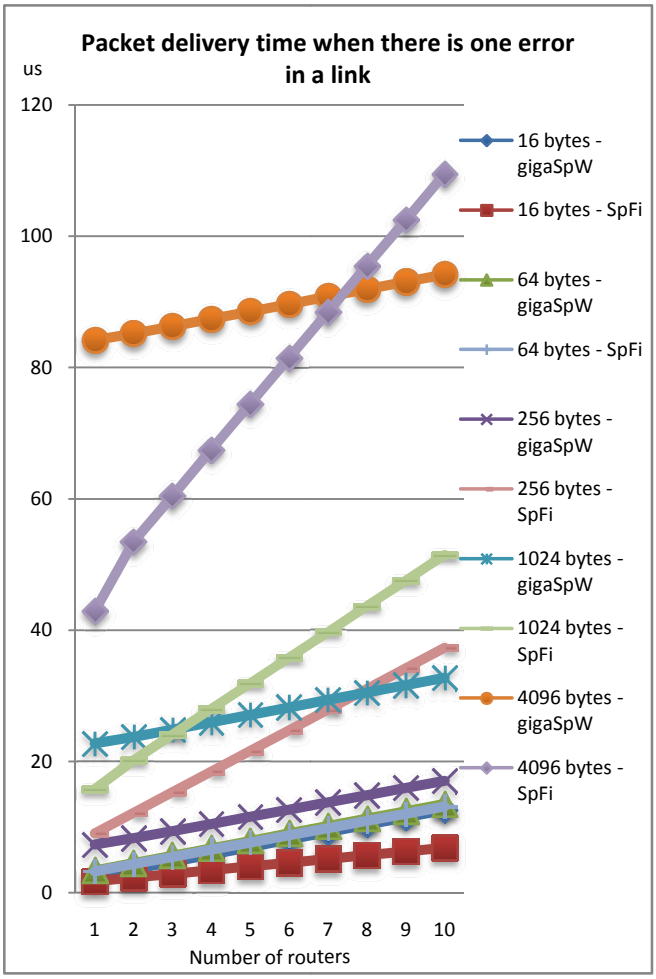

Fig. 5. Dependencies between packet delivery time and number of routers for the one error case (without disconnections)

- Sum of noise duration and the connection restore time is larger than the sum of the acknowledgment waiting timeout and repeat packet transmission time up to the link, in which disconnection happens. In this case packet delivery time consists of the acknowledgment waiting timeout, repeat packet transmission time between the source and the destination and the time to restore the connection.

Dependency between the packet delivery time and the packet size, when the noise duration is 1 us, is presented in Fig. 6. For SpaceWire/GigaSpaceWire in this case acknowledgment waiting timeout is equal to packet transmission time between the source and the destination plus the NACK transmission time.

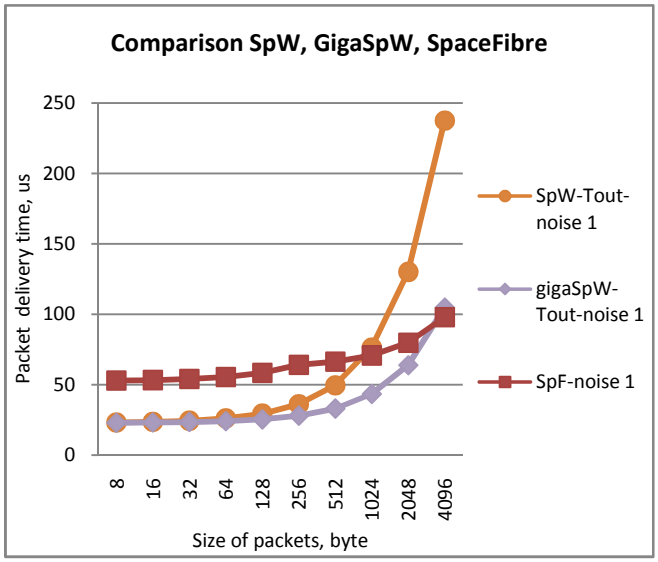

Fig. 6. Dependency between the packet delivery time and the packet size when a transmission error occurs (Tout = Tf

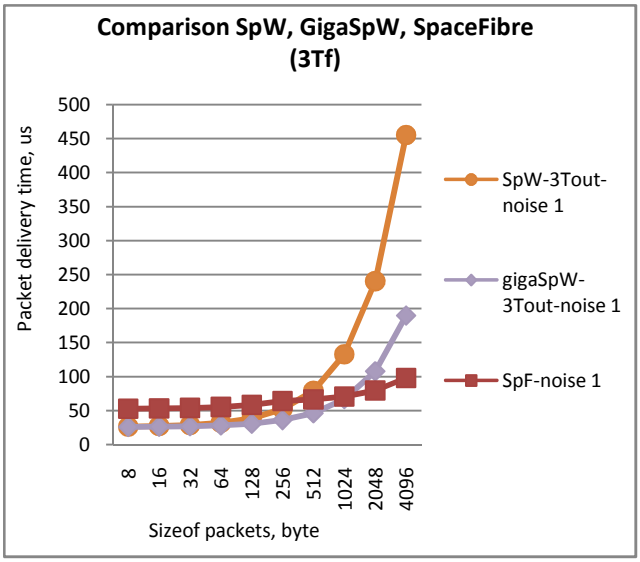

Fig. 7. Dependency between the packet delivery time and packet size when transmission error occur (Tout = 3*Tf).

As one can see from the Fig.6, packet delivery time in SpaceWire is less than in SpaceFibre if size of packets ≤ 1024 bytes. Packet delivery time in GigaSpaceWire is less than in SpaceFibre if size of packets ≤ 2048 bytes.

However this value of acknowledgment waiting timeout for SpaceFibre/GigaSpaceWire networks may be selected in cases when packet and NACK paths do not interfere with packet paths of other traffic. Namely, packet and its NACK don't wait in output ports. If packet and NACK paths interfere with packet paths of other traffic, we should consider waiting time to access an output port also in a value of acknowledgment waiting timeout.

Let us evaluate packet delivery time when acknowledgment waiting timeout is equal 3*Tf. Dependency between the packet delivery time and packet size when the noise duration is 1 us is presented in Fig. 7.

As one can see from the Fig.7, the packet delivery time in SpaceWire is less than in SpaceFibre if size of packets ≤ 512 bytes. Packet delivery time in GigaSpaceWire is less than in SpaceFibre if size of packets ≤ 1024 bytes.

Let's consider the packet transmission time in case when no errors occur during transmission and packets are not retransmitted. The plots of dependency between packet transmission time and the transit routers number are presented in Fig. 9.

As can be seen, the packet transmission time (for packets with considered length) is 1,5 times less for GigaSpaceWire than for a SpaceFibre network. The wormhole routing used in SpaceWire/GigaSpaceWire routers reduces the packet transmission time via network. The packet transmission time in a SpaceFibre network is about 1,5 times bigger due to delays associated with the full frame buffering and CRC checking in each data link. This check is made for all types of traffic, including the traffic for which guaranteed data delivery is not required by an application.

This delay is especially significant for short packets with less than 256 Nchars length. The transmission time of short packets is bigger for SpaceFibre network with 1250Mbit/s transmission rate than even for SpaceWire network with 400Mbit/s transmission rate. Short packets typically are used for command traffic therefore its delivery time is particularly important.

It is important to understand that provided in the SpaceFibre guaranteed delivery mechanism, cannot guarantee a packet delivery if there would be faulty network equipment or links. Therefore for networks with high guaranteed delivery requirements one still need to use mechanisms of packet replication at the hardware level.

If the hardware and data redundancy is used in a SpaceFibre network in combination with standard retry mechanism and recoverable connection breaks, then correct interpretation of the packet replicas that goes via a path with temporary disconnection is very difficult.

Connection recovery in a SpaceFibre link may take a long time – duration of connection procedure is 50 us. The duration of noise may be added to this time. Therefore one copy of packet can reach the destination node with a very noticeable delay (dozens – hundreds of us, dependent on duration of noise) in comparison with other copies that goes via paths without disconnections.

In systems with data duplication for redundant transmission (N replicas of one packet are sent to the network) typically packet numbering is used. The receiver terminal node determines by its number whether it has already a copy of this packet.

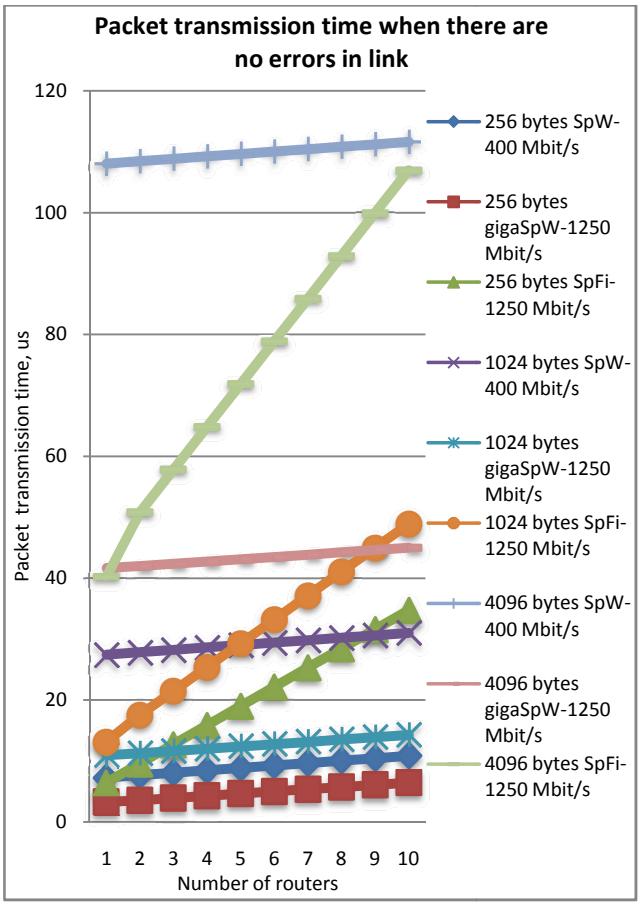

Fig. 8. Dependency between the packet transmission time and the amount of transit routers (without errors during transmission)

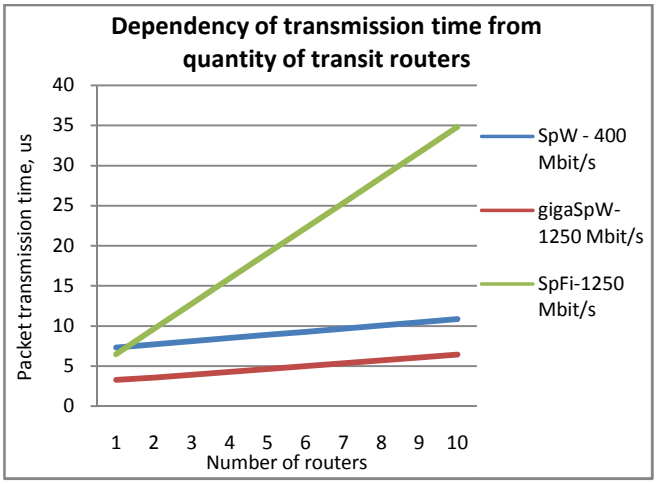

Fig. 9. The dependency of transmission time from quantity of transit routers for packets with 256 Nchars length

The packet number field in the packet structure has constrained size. In short command packets size of this field is typically 3 – 8 bits. Correspondingly the length of the cycle after which numbers will be repeated is small, especially when commands go often. So it could be very difficult to determine by the command sequence number whether the late command is belated, or goes on time or goes with some advance.

IV. Guaranteed Packet Delivery Time

To ensure guaranteed packet delivery time a scheduled packet transmission is used. A list of timeslots for data transmission is assigned for every application, every packet source in a terminal node. Timeslots are linked with destination addresses of the packets. The data transmission path for different packet traffics, which are assigned to one timeslot, should not share data links.

It can be implemented in SpaceWire/GigaSpaceWire networks, [4]. Terminal nodes are responsible for data transmission only in allowable for them timeslots. However, a terminal node (an application in this terminal node) may still start data transmission out of allowable timeslot due to failure/error of time synchronization or due to internal malfunctions (for example distortion of the bits in the timeslots table). It may increase packet delivery time, violate the time constraints for the data packets that are transmitted in time but cross with the packet that runs out of its timeslot. The scheduling and data transmission in corresponding timeslots must be made by the trusted component of the terminal node – by special network controller (not by software) to deal with this problem.

Another source of a packet transmission out of its timeslot is data link disconnection. For example if the packet should be transmitted to an router output port that is currently under connection recovery, the packet will stop for a time until the connection is set; after it the packet will continue its transmission. Meanwhile the time slot could be finished and the packet would run in another, in alien timeslot.

In packets delivery scheduling one needs to take into account such situation, to set appropriate margins in the time schedule. To determine the margins the network topology, transient faults and errors probability, which can lead to link disconnections, should be considered and took into account. With appropriate margins we can eliminate, with certain probability, sending packets in wrong time slots.

In SpaceFibre the timeslots control is performed in each data link of a router and a terminal node. Scheduling is associated with virtual channels, not with packets. For data flows transmission in predetermined timeslots, data flows should be assigned to different virtual channels and the timeslots table should be assigned to corresponding virtual channels in every data link.

As the data transfer is controlled in every data link a data transmission out of its timeslot will be quickly stopped in the nearest network node in case of incorrect behavior of a terminal node. Even if a terminal node transmits the data packet out of its timeslot (and this transmission has not been blocked in the SpaceFibre output port of the terminal node itself) the packet goes to the next SpaceFibre router. It will be received in the SpaceFibre router input port, go to an output port, where its further transfer will be suspended until a corresponding timeslot.

The Figure 10 represents plots of the packet delay due to a transmitted out of its timeslot traffic.

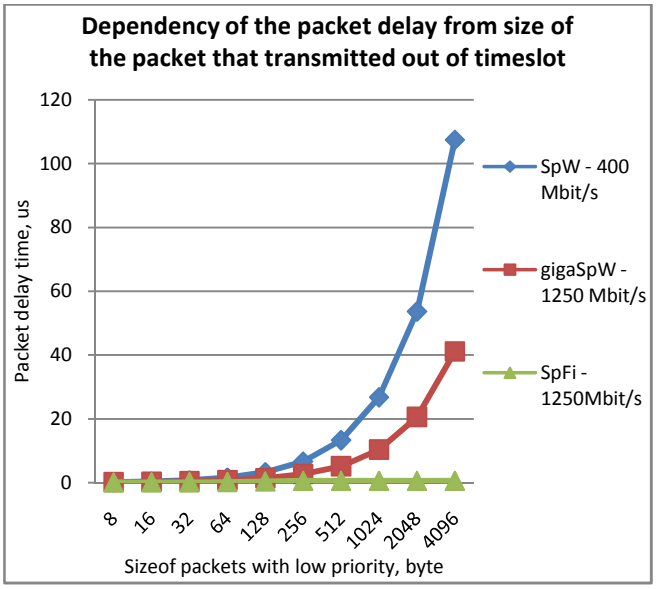

Fig. 10. Dependency of the packet delay in one router from the size of the packet that is transmitted out of his timeslot

If we need to transmit traffic with guaranteed delivery time and traffic without this requirement, the one part of time slots can be for traffic with guaranteed transmission time and remaining time slots to other traffics. The transmission paths of the traffic without guaranteed time requirement can share interconnection paths in assigned for them all time slots.

We should take into account also that in SpaceFibre timeslotting is associated with virtual channels, not with packets. As in the priorities case, the quantity of virtual channels in a SpaceFibre data link is constrained by its hardware cost. Typical virtual channels quantity per data link is 4 or, rarely 8. The specified in the SpaceFibre standard draft quantity of 256 VC is practically impossible for implementation in VLSI. Therefore in SpaceFibre there is very limited number of objects for scheduling. Applications' packets flow scheduling and selection of data transmission paths will be essentially complicated in comparison with SpaceWire/GigaSpaceWire networks. The quantity of data flows that can be planned to time slots is very limited.

V. Conclusions

As has been shown abode, the QoS may be implemented both in SpaceFibre and in SpaceWire/GigaSpaceWire networks. While SpaceFibre strives for QoS in every data link, SpaceWire/GigaSpaceWire could implement QoS at the boundaries of the network, in its terminal nodes, at the Transport layer.

If we constrain the packet length in SpaceWire and GigaSpaceWire network by value of 256 Bytes (equal to the SpaceFibre data frame size) the timing parameters of high priority traffic transmission in these networks and SpaceFibre are similar. This constraint does not affected essentially transmission time of big data objects, which in this case would be sent by multiple packets. Basically, the question goes down to where the data objects (messages) will be sliced into pieces – in terminal nodes (as packets) or in every data link (as frames).

The data transmission time when network error occur in SpaceFibre will be better than in SpaceWire/SpaceWire GigaSpaceWire network with retransmission of lost or incorrect packet between the source and destination nodes, for example at the Transport layer. On the other hand, impossibility of turning off for retry mechanism in SpaceFibre leads to essential growing of data packets transmission for traffic without guaranteed delivery requirement (e.g. video data streams). Potential incorrect interpretation of the packets that arrive too late due to waiting of connection recovery in network with packets duplication is another problem.

For traffic with guaranteed delivery in SpaceWire/GigaSpaceWire networks same characteristics could be reached as for SpaceFibre networks if all data flows will be transferred strictly in the assigned timeslots. If can happen that some traffic is transmitted out of its timeslots (as result of malfunctions of terminal nodes or disconnections on links) then SpaceFibre operation will be more reliable. It checks the schedule in every data link and will stop invalid in time transmission in the first network node on its.

Traffic parameters for SpaceWire/GigaSpaceWire networks can be similar to the SpaceFibre ones when SpaceWire packet length is constrained by 256 bytes.

In general, in timing characteristics for QoS traffic both SpaceFibre and SpaceWire/GigaSpaceWire area balancing in their gains in relation to network topology, error probability, size and features of target data items. In many cases they could be made rather similar.

The SpaceFibre advantages are in QoS mechanism immersion in every data link that makes them more reliable in case of network components malfunctioning. Drawbacks of the SpaceFibre approach to QoS are much higher implementation costs and longer latencies in packets delivery.

The SpaceWire/GigaSpaceWire QoS approach is considerably cheaper in implementation, gives lower latencies, and may operate over conventional SpaceWire/GigaSpaceWire network backbone. However, without control of packets transmission QoS rules and assignments inside the network backbone, it may be more sensitive to errors and network components malfunctioning. What could be included in a SpaceWire router node for more reliable QoS network operation, without sacrificing the native SpaceWire feature – compactness and simplicity, is a good subject for further research.

Acknowledgment

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under contract n° 14.578.21.0022.

REFERENCES

[1] *S. Parkes, A. Ferrer, A. Gonzalez, C. McClements. SpaceFibre Specification. Draft F3, September 2013*

[2] *ECSS-E-ST-50-12C, European Cooperation for Space Standardization, "SpaceWire – Links, nodes, routers and networks", 31 July 2008*

[3] *165. E. Yablokov, Yu. Sheynin, E. Suvorova, A. Stepanov, T.Solokhina, Ya. Petrichcovitch, A. Glushkov, I. Alekseev, "GigaSpaceWire – Gigabit Links for SpaceWire Networks", SpaceWire-2013. Proceedings of the 5th International SpaceWire Conference, Gothenburg 2013. Editors Steve Parkes and Carole Carrie. ISBN 978-0-9557196-4-6, Space Technology Centre, University of Dundee, Dundee, 2013, pp. 28-34.*

[4] *D. Raszhivin, Yu. Sheynin, A. Abramov, "Deterministic Scheduling of SpaceWire Data Streams", SpaceWire-2013. Proceedings of the 5th International SpaceWire Conference, Gothenburg 2013. Editors Steve Parkes and Carole Carrie. ISBN 978-0-9557196-4-6, Space Technology Centre, University of Dundee, Dundee, 2013, pp. 141-144.*

Poster Presentations

Implementation of a RMAP Bootloader for the Solar Orbiter RPW Experiment

SpaceWire Missions and Applications, Poster Paper

Philippe Plasson, C. Cuomo, T. Gadeaud, A. Gaget, L. Gueguen, L. Malac-Allain, E. Revert
Laboratory of Space Studies and Instrumentation in Astrophysics (LESIA)
Observatoire de Paris, CNRS, UPMC, Université Paris-Diderot
5 place Jules Janssen, 92195 Meudon, France
Philippe.Plasson@obspm.fr

***Abstract*— This paper presents the implementation of a RMAP bootloader which has been developed in the context of the RPW (Radio Plasma Waves) experiment on the Solar Orbiter mission. It describes the different steps of the RMAP boot mechanism, the performances of this process, the main advantages of such an approach and its integration as a reusable software block in the GERICOS (GEneRIC Onboard Software) framework.**

***Index Terms*—RMAP, Bootloader, RPW, Solar Orbiter, LEON3-FT, Flight software**

I. Introduction

The Radio and Plasma Waves (RPW) experiment [1] is one of the ten instruments of the ESA Solar Orbiter mission which will be launched in 2017. The RPW instrumentation is a sophisticated plasma/radio wave receiver system providing in-situ measurements of both electrostatic and electromagnetic fields and waves in a broad frequency range. The RPW consortium is led by CNES (RPW management) and LESIA (PI institute). It includes the scientific and technical participations of the following labs and institutes: LPC2E (Orleans), LPP (Palaiseau), IRF (Uppsala), IAP (Praha), IWF (Graz).

In this paper, after presenting the electrical and software architecture of the instrument, we describe how the RMAP protocol [2] has been used to implement a remote boot process of the RPW subsystem software. We also give the various justifications which have led to this technical choice. Some results about the performance of the RMAP boot process are detailed. Finally, we show how the RMAP boot loader modules have been developed and qualified as generic and reusable pieces of software.

II. RPW Instrument Electrical and Software Architecture

A. RPW Electrical Architecture Overview

1) RPW System

The RPW instrument is divided in three subsystems located in different parts of the Solar Orbiter spacecraft:

- The MEB (Main Electronic Box) is located with in-situ systems.
- The SCM (Search Coil Magnetometer) is located on the S/C boom.
- The Antennas are located on the three sides of the S/C.

2) RPW MEB

The RPW Main Electronic Box is made up of several electronic boards. Four RPW sub-systems embed a LEON3-FT processor and thus a flight software:

- the first one is the Data Processing Unit (DPU) which is in charge of the communication with the spacecraft via a SpaceWire interface;
- the three others are analyzer boards in charge of the electrical field and magnetic field signal acquisition and processing.

The RPW MEB contains also:

- a Bias Unit board driving the currents to the electric antennas;
- a Low-Voltages Power Supply (LVPS) and Power Distribution Unit (PDU).

The figure below shows the RPW MEB architecture:

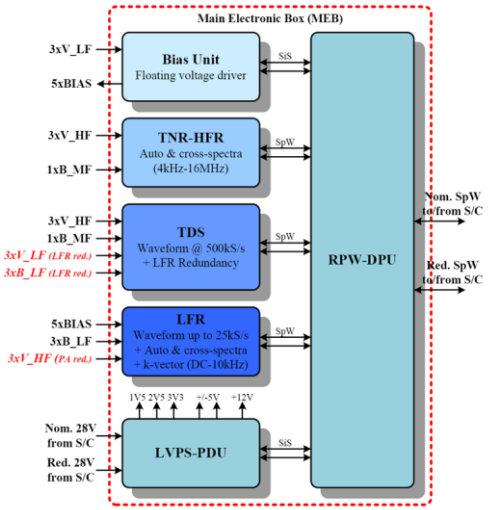

Fig. 1. RPW MEB Architecture

3) RPW DPU

The RPW DPU is based on the Aeroflex LEON3-FT UT699 processor. It handles four SpaceWire interfaces, one toward the spacecraft, the three other toward the analyzer boards.

The RPW DPU embeds an EEPROM (4 MB) for storing the software images, a SRAM memory module (64 MB) allocated to the software execution and science data processing and a PROM (64 KB) for storing the DPU boot software.

In addition to the processor, a FPGA is used to generate the control signals for the memory management and the synchronous serial interfaces to the RPW Bias Unit and to the RPW Power Distribution Unit.

They are two DPU, one nominal and one redundant, the redundancy scheme being a cold redundancy.

4) RPW Analyzer Boards

The first analyzer board is a Low Frequency Receiver (LFR) which is designed to produce waveforms, averaged spectral matrices and basic parameters from LF electromagnetic waves in the range from quasi-DC to 10 kHz.

The second analyzer board is a Time Domain Sampler (TDS) which consists in a medium frequency wave analyzer that processes analogue signals from electric field antennas and search coils at sampling rates up to 524288 samples per second.

The Third analyzer board is a Thermal Noise and High Frequency Receiver (TNR-HFR) which provides electric power spectral densities from 4 kHz up to 16.4MHz and magnetic power spectral densities from 10 kHz up to 500 kHz.

The three RPW analyzer boards (LFR, TDS and TNR-HFR) are based on a LEON3-FT processor synthetized in a RTAX FPGA. They communicate with the DPU thanks to a point-to-point SpaceWire link. They use as SpaceWire link controller a GRSPW core with RMAP support.

B. RPW Software Architecture Overview

The RPW Instrument contains five flight software: the DPU application S/W, the LFR flight S/W, the TDS flight S/W, the TNR-HFR flight S/W and the DPU boot S/W.

1) DPU Application S/W

The DPU Application Software (DAS) is a complex software managing lots of various interfaces and implementing the standard PUS services but also a set of services specific to the RPW experiment. The DAS is responsible for managing the RPW modes, switching on/off the RPW sub-units, configuring and commanding the RPW sub-units, verifying and executing the RPW telecommands received from the S/C, monitoring the RPW sub-units, reporting housekeepings and events, supporting the FDIR mechanisms, distributing the time to the analyzer boards, processing, compressing and packetizing the science data sent by the analyzer boards, performing science event detection. The DAS communicates with the spacecraft using the CCSDS protocol over the SpaceWire link [3].

2) LFR, THR and TDS Flight S/W

The main role of the analyzer flight software is to acquire and pre-process the raw data provided by the sensors. The pre-processing includes event detection, data reduction, data selection and lossy compression. The communication protocol used between the DPU and the analyzers for exchanging TC and TM is the CCSDS protocol.

3) DPU Boot S/W

To manage the maintenance and the boot of the DPU application software, a standard and well proven architecture based on the use of a separate boot software has been implemented. The DPU boot software is a critical and low complexity software stored in PROM. The DPU application software, that is stored in the DPU EEPROM, is loaded in the working memory and started upon the reception of a dedicated TC packet by the boot software. The boot software implements also the memory management PUS service and is able to patch or fully change in EEPROM the application software.

4) Analyzer Flight S/W Boot and Maintenance

The approach chosen for managing the boot of the DPU Application S/W is a well proven approach, used for many years in the context of lots of space missions. However, this approach has a relatively high cost in terms of development. A boot software is never a trivial software: in addition to the boot mechanism itself, it shall implement a subset of the PUS services [4], as the service 1 (telecommand verification), the service 3 (housekeeping reporting), the service 5 (event reporting), the service 6 (memory management). A boot software shall be stored in PROM to avoid any corruption or unexpected erasing of the software image during the mission. The consequence is that this kind of boot software cannot be changed or patched during the flight. A software failure, not seen during the validation, which would occur for example in the module managing the communication with the spacecraft or in the module managing the boot process itself could have dramatic consequences for the mission and could result in the loss of the instrument. That's why the criticality level of the boot software is the level B according to the ECSS-E-40 standard [5]. The effort for qualifying a level B software is huge in terms of validation, code coverage analysis, design justification and quality rule verification. Developing such a boot software is not always possible for the instrument teams which prefer focus their efforts on the development of the application software which is directly linked to the scientific return of the mission.

All these arguments have pushed the RPW team to find another solution for tackling the issue of the analyzer flight S/W boot process and maintenance management.

In order to simplify the overall electronic and software architecture of the RPW instrument and to suppress the risks and costs inherent to the development of a level B software, the boot process and maintenance management of the three analyzer software has been delegated to the DPU application software. A solution based on the capability offered by the SpaceWire / RMAP technology has been studied.

III. RMAP Bootloader Mechanism

A. RMAP Bootloader General Principle

The boot process of each analyzer flight software is performed remotely, over the SpaceWire link, by the DPU application software using the RMAP protocol. Each analyzer board integrates a RMAP hardware controller, which allows the DPU to access to any area of its memory, including the processor registers. The RMAP protocol offers all the mechanisms allowing to have remote memory write and read operations highly reliable. In this sense, the RMAP protocol is very suitable for implementing a remote boot mechanism for the flight software.

Thanks to the RMAP protocol, the DPU can remotely configure the registers of the LEON3-FT processor analyzer boards, load a software image in the SRAM analyzer board and start it without intervention of any local boot software. The boot process of the analyzer software is entirely under the responsibility of the DPU application software.

The image of each analyzer flight software is stored in the EEPROM of the DPU. In addition to being responsible for the analyzer boot process, the DPU application software is also responsible for the maintenance of these three software images. During the flight, the various software images can be patched or entirely replaced by a new one directly in the DPU EEPROM by the DPU application software using the standard PUS memory management service (PUS service 6). Each flight software executable image is stored in EEPROM as a sequence of several data/opcode segments and one end segment, each segment being protected by a checksum. The size of each segment is limited to the maximum size (204 bytes) of one telecommand in order to simplify the code upload process and the code maintenance: each memory load TC (PUS service 6,2) allows to upload exactly one data/opcode segment. In case of failure of the EEPROM at the DPU side, the RMAP boot process allows to retrieve the analyzer software images directly from the DPU SRAM instead from the EEPROM. A set of GSE software tools have been developed to generated automatically from the SREC (S-Record format) files provided by the compilation chain a sequence of telecommands allowing to upload in the EEPROM of the DPU any analyzer S/W image.

With this approach, there is no need to have, at the analyzer board level, a boot software whose the development and the qualification would have been costly and whose a failure during the flight would be critical for the mission.

In the same way, there is no more need to have EEPROM parts or PROM parts at the analyzer board level: this simplifies the hardware design and reduces the number of potential failures. The overall development cost (hardware and software) is also clearly minimized.

B. RMAP Boot Process Steps

1) RMAP Boot Process Overview

The RMAP boot process is divided into the following steps:

- Identify the start address of the analyzer flight S/W in the DPU source memory (EEPROM or SRAM). The flight S/W start address is a parameter of a telecommand packet or can be retrieved in the RPW operational context maintained by the DPU application S/W in case of boot operation triggered after an internal decision taken in the context of the FDIR (Failure Detection and Isolation Recovery) mechanisms.
- Check in the DPU source memory (EEPROM or SRAM), where the flight S/W is stored as a sequence of opcode and data segments, the completeness and correctness of each software segment using checksums.
- If the flight S/W image is complete and correct in the DPU source memory, configure the registers of the analyser LEON processor before loading the application image itself.
- Copy, using RMAP write commands, all the software segments from the DPU source memory (EEPROM or SRAM) to the analyzer executable memory (SRAM).
- Check in the analyzer executable memory (SRAM), using RMAP read commands, that the flight S/W has been correctly deployed.
- If the flight S/W has been correctly deployed, start the flight software.

The success of each step depends on specific criteria: if a failure is detected, the boot process is stopped and an event report TM packet is generated to notify the ground segment of the error.

2) Boot Process Starting

The RMAP boot process of the analyzer S/W is started upon the reception of a telecommand from the spacecraft (ground telecommand or OBC telecommand). The telecommand contains a parameter allowing to choose the location in EEPROM (or SRAM) of the software image to be booted. The RPW EEPROM has been sized to be able to store up to 2 software images for each of the four RPW flight S/W. A second parameter allows to enable or disable the verifications which are performed during the various steps of the boot process.

3) Software Image Integrity Verification

This second step consists for the boot manager in checking that each data / opcode segment stored in EEPROM and forming the software image pointed by the logical address given in the telecommand are not corrupted. The verification is based on the computation of a XOR checksum on the data / opcode block and to the comparison of this XOR checksum with the XOR checksum contained in the trailer of the segment. During this step, the boot manager checks also that the destination address and the size of the data / opcode block contained in each segment header are in authorized ranges. Each segment shall be immediately followed by another segment. The verifications ends with success if the boot manager finds an end segment and that all the verifications performed for the previous segments have been successful.

4) Remote Processor Configuration

In this step, the DPU application S/W takes the control of the remote processor (LEON3-FT inside the analyzer board).

The DPU application S/W puts the remote LEON3-FT processor in the debug state. To do that, it sends a RMAP write command toward the analyzer board for configuring the Debug Support Unit (DSU) Control Register and for setting the Break-Now (BN) bit of the DSU Break and Single Step Register of the remote processor. It has to be noted that after the initial powering-on sequence, the LEON3-FT processor of each analyzer board goes by itself in the debug mode without executing any instructions: this is done by asserting DSUEN and DSUBRE signals at reset time [6].

Then, before loading the S/W image, the boot manager performs on the remote LEON3-FT processor the setting listed below by accessing its various registers using RMAP write commands:

- Set up analyzer memory configuration: the analyzer memory controller shall be initialized before the RAM can be accessed. This shall be accomplished by writing to the MCFG[1:3] registers over RMAP.
- Disable interrupts
- Disable the watchdog
- Clear the IU register files
- Clear the FPU register files
- Set up the Y, PSR, WIM, FSR registers
- Set up GPIO if needed
- Set up AHBSTAT if needed
- Set up the PC and TBR registers to the entry point address of the analyzer software
- Set up the NPC registers to the entry point address plus 4 bytes
- Set up the stack pointer
- Configure the timers

5) Software Image Deployment

Once the remote LEON3-FT processor is configured, the code deployment phase itself can start. The content of each data / opcode segment is copied from the local EEPROM to the remote SRAM by using RMAP write commands.

Just after having copied a segment, the boot manager reads back, using a RMAP read command, the segment written in the analyzer memory. Then, it compares the content of the read segment with the source content stored at the DPU level. If the comparison fails, the boot process is stopped and an error event report packet is generated. This verification has been implemented to make the boot process fully reliable. However, it has to be noted that the RMAP protocol includes already a verification mechanism, based on a CRC computation, of the integrity of both the write command header and data. In case of corruption of the data during the transfer, the analyzer RMAP controller will detect it and reply with a negative acknowledgment reporting the cause of the error ("invalid data CRC error"). The CRC implemented in the RMAP protocol is a 16-bit CRC highly reliable (99.998%): the verification based on the read back operation of the written blocks could be optionally skipped without taking too much risk.

6) Remote Software Starting

When all the data / opcode segments are copied and checked, the boot manager has to release the remote processor from the debug state by writing 0 to the DSU control register and by clearing the Break-Now bit of the Break and Single Step Register. The remote application is then started and can perform its own initializations.

7) RMAP Boot Process Ending

If all the different steps of the boot process have been successful and as soon as the first housekeeping packet is received from the analyzer board, the DPU application S/W produces a progress event report TM packet notifying that the boot process is a success and giving the version number of the booted software.

8) Error Management and Failure Reporting

The following errors can cause the failure of the boot process:

- At a given address in the DPU memory (EEPROM) or after a list of valid segments, there is no other valid segment.
- A calculated checksum over a segment is different from the checksum value at the end of this segment.
- At a given address, there is only an end segment.
- The destination address in a segment is invalid.
- The size of a segment is invalid.
- The content of a segment copied in the analyzer memory is different of the content of the segment in the DPU memory.

All these errors cause the aborting of the boot process and the generation by the DPU application software of an error event report TM packet giving information useful for the diagnosis.

The errors which can occur at the SpaceWire level or at the RMAP level are also reported by the DPU application S/W and are normally recovered thanks to a retry mechanism. If a RMAP command fails due to any reason (EEP, Invalid Data CRC, Too much data…), the RMAP software driver implemented in the DPU application S/W will repeat it again. The boot manager knows that a command has been successfully executed because all the RMAP commands are acknowledged by the receiver, the Reply bit of the RMAP write commands being set to 1.

IV. RMAP BOOTLOADER PERFORMANCE

The RMAP bootloader performance, in terms of duration of the boot sequence, depends on the following parameters:

- The CPU clock of the board hosting the RMAP boot loader.
- The SpaceWire data rate.
- The size of the remote software image.

Concerning RPW, the CPU clock of the DPU is 25 MHz and the SpaceWire data rate is 10 Mbps.

The duration of the RMAP boot process which has been measured for each RPW analyzer flight S/W is given in the following table:

TABLE I. RMAP BOOT PROCESS DURATION

S/W	Size (KB)	Boot process duration (s)
LFR	260	8
TDS	95	5.5
TNR-HFR	211	7.5

The CPU load and the output data rates reported by the DPU Application S/W during the RMAP boot process show that, with the RPW configuration, the main bottleneck is the CPU resource which reaches 100%. The SpaceWire link occupation rate never exceeds 4% during the boot process. With a processor clocked at 50 MHz, it should be possible to reduce by a factor of two the boot duration (less than 4 seconds for a 260-KB software). With a processor clocked at 100 MHz, the boot duration for a 260-KB should be lower than 2 seconds. By skipping the read back operation of the written blocks, with the same configuration, the boot duration should be lower than 1 second.

The figures below show the CPU load and the SpaceWire transmission rate during the RMAP boot process of the three RPW analyzers.

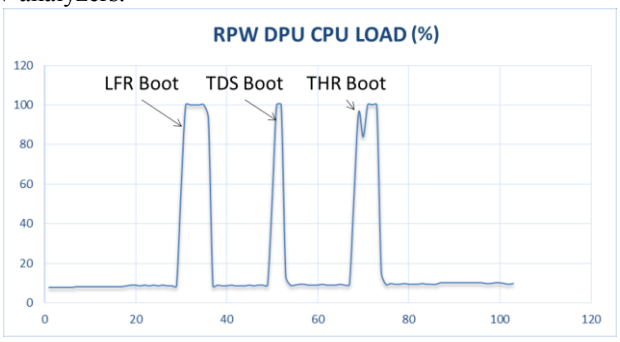

Fig. 2. CPU load during the RMAP boot process

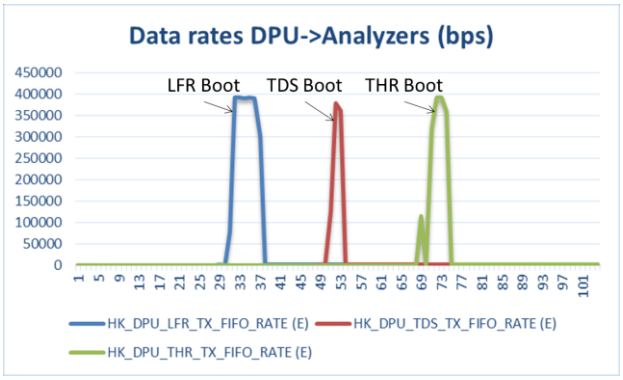

Fig. 3. SpaceWire transmission data rate during the RMAP boot process

V. Design and Qualification of the RMAP Bootloader

The RMAP bootloader developed for the RPW instrument has been designed to be a fully reusable software block. The RMAP bootloader modules have been developed in C++ using a coding standard which is compliant to the ECSS-E-40 requirements and which is derived from the Lockheed Martin Joint Strike Fighter standard Air Vehicle C++ coding standard [7]. Its object-oriented design makes it easily re-usable in the context of other projects and missions. The RMAP bootloader modules are part of the LESIA GERICOS (GEneRIC Onboard Software) framework. The GERICOS framework offers a set of C++ libraries allowing to build a flight software by using a collection of generic, re-usable, interoperable and space-qualified software bricks implementing various things as a RTOS object-oriented abstraction layer, SpaceWire and RMAP drivers, the main PUS services...

The qualification of the RMAP bootloader modules has been carried out according the requirements of the ECSS-E-40. All the code of the RMAP bootloader modules has been verified thanks to quality tools as Logiscope (metrics and coding rules) or Polyspace (static analysis tool).

The development and the integration tests have been performed using a DPU breadboard and specific home-made simulators allowing to produce any RMAP or SpaceWire errors. Several tests have been carried out for proving the robustness of the RMAP bootloader to transient SpaceWire errors and to RMAP failures.

The final validation of the RMAP bootloader modules has been performed on the RPW EM boards. The RMAP boot process and its implementation are now validated. The result of this validation has been submitted to the RPW CDR review group.

VI. Conclusion

The RMAP boot loader developed for the RPW instrument is now fully operational. It has been integrated in the RPW DPU application S/W and is currently extensively used in the context of the RPW AIT. No major issues have been encountered during the various tests carried out at AIT level or at software development level. The approach can be now considered as successful.

This approach takes benefit from the RMAP protocol and its close integration with the LEON-3FT processor. The gains, in terms of risk reduction and cost saving, relatively to the development of a standard boot software, have been clearly confirmed. Thanks to this technology, the RPW analyzer flight software are fully reconfigurable during the flight: this is crucial for the success of the mission.

The RMAP boot loader modules have been designed as a set of generic software bricks, integrated in the GERICOS framework, which could be easily reused in the context of other space instruments.

In the context of the PLATO 2.0 project, studies are going to be carried out in order to assess if the concept of RMAP bootloader could be used. With the large number of DPU boards making up the on-board data processing system, the benefits could be huge for the PLATO payload team.

Last but not least, we advocate that the use of such an approach could be also envisaged, by the space agencies and the main prime contractors of the domain, at spacecraft level for managing the boot of the payload DPU themselves. Such a generic approach would reduce the development cost of the scientific payloads.

References

[1] Solar Orbiter Radio and Plasma Waves Instrument (RPW). http://lesia.obspm.fr/Solar-Orbiter.html

[2] ECSS Standard, “SpaceWire - Remote memory access protocol”, ECSS-E-ST-50-52C, Feb 2010.

[3] ECSS Standard, “SpaceWire - CCSDS packet transfer protocol”, ECSS-E-ST-50-53C, Feb 2010.

[4] ECSS Standard, “Ground systems and operations - Telemetry and telecommand packet utilization”, ECSS-E-70-41A, January 2003.

[5] ECSS Standard, “Software”, ECSS-E-ST-40C, March 2009.

[6] Aeroflex Gaisler, “GRLIB IP Core User’s Manual”, April 2014.

[7] Lockheed Martin Corporation, “Joint Strike Fighter Air Vehicle, C++ Coding Standards”, 2RDU00001 Rev C, December 2005

The study and proposal for improvement the multi-lane operation of SpaceFibre protocol

SpaceWire standardisation, Poster Paper

Yu Otake[1], Kohei Hosokawa[1], Yasuhiro Sota[1],
Takahiko Tanaka[1], Hiroki Hihara[2]

[1]NEC Corporation Tokyo, Japan
[2]NEC TOSHIBA Space Systems, Ltd. Tokyo, Japan
y-otake@bp.jp.nec.com

***Abstract*— SpaceFibre protocol, which is developed by European Space Agency as a standard protocol to communicate between payloads in satellite network, adopts multi-lane technique to increase transfer rate by using multiple physical links. To realize high transfer rate in which the multi-lane technique is needed, e.g. 20Gbps, FPGAs or ASICs devices to process the SpaceFibre protocol needs to increase operating frequency or to extend processing data width at one clock cycle. The later way is generally adapted for giga bps class transfer system since the operating frequency of FPGAs and ASICs for space systems is relatively lower than these devices for commercial systems.**

It is important to consider a length of data to be processed when extending the processing data width at one clock cycle. In the SpaceFibre protocol, it is effective to design processing device with 32-bits data width, because the minimum unit of transmitting/receiving data width is 32-bits. For example, however, if the width of data bus is assumed to 64-bits, it is possible that frame boundaries between leading data frame and following data frame is appeared in the middle of data bus. In this case, two frames have to be processed at the same time and it causes increase of complexity for processing and increase of circuit size. To mitigate the complexity, we change the frame length to be aligned with the utilized lane number at the framing layer. In this paper, we report the method for alignment of frame length and its evaluation results.

***Index Terms*— SpaceFibre, Multi-lane, throughput, implementation**

I. INTRODUCTION

The SpaceFibre protocol in Fig. 1 is currently developed, and applies various techniques such as virtual channels, quality of services and retransmission [1]. Furthermore, the SpaceFibre protocol has a multi-lane transmission as an optional feature which can enhance transfer rate by using several physical links simultaneously (hereafter, a "physical link" is called by a "lane"). However, implementing the multi-lane transmission into actual devices has some problems. Although details of those problems are described in chapter II, continuous data might be unable to be processed in one clock cycle for aerospace devices. Therefore, we describe a framing method to adjust a number of words in a data frame to a multiple of a number of used lanes for multi-lane transmission so as to process a data frame efficiently by a circuit in chapter III. We also show evaluation results of data throughput with our method in chapter IV, because the data throughput with our method decreases from that with the standard specification of the SpaceFibre protocol.

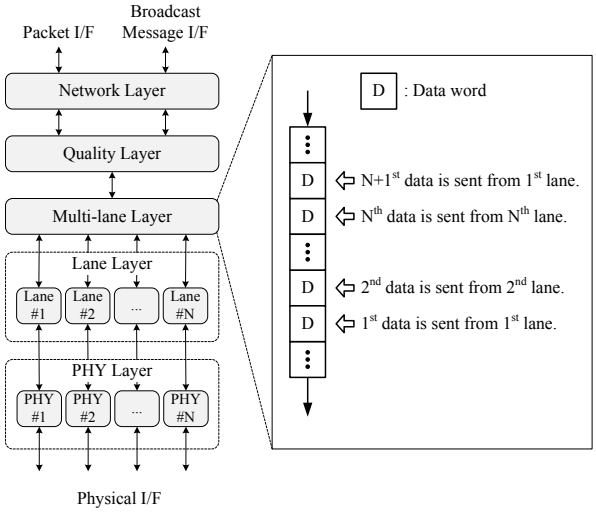

Fig. 1. Overview of multi-lane transmission of the SpaceFibre protocol

II. PROCESSING PROBLEM IN MULTI-LANE TRANSMISSION

The minimal meaningful unit of the SpaceFibre protocol is 32-bit data (one word). For example, framing control words, such as Start Data Frame (SDF) and End Data Frame (EDF) to configure one data frame, Start Broadcast Frame (SBF) and End Broadcast Frame (EBF) to configure one broadcast frame and Start Idle Frame (SIF) to configure one idle frame, are 32-bit data. Similarly, other control words such as Flow Control Token (FCT), ACK and NACK are also 32-bit data. Therefore, it is efficient to design a processing device for the SpaceFibre protocol with 32-bit data width. If a processing device is designed with 32-bit data width, a required frequency is 62.5 MHz to achieve 2.0 Gbps throughput which is an initial transfer rate of the SpaceFibre protocol by using only a single lane. This operating frequency is feasible for an aerospace device [2][3].

In multi-lane transmission of the SpaceFibre protocol, all words are distributed and transmitted in parallel by using the multiple lanes. In multi-lane transmission, at transmitter side, the first word shall be sent over the lane with lowest number(lane #1), the next word shall be sent over the lane with the next number of the lowest number(lane #2), and so on, as

shown in Fig. 1. Multiple words, which is the same as the number of used lanes, are transmitted simultaneously to a receiver side. Besides, each transfer rate of multiple lanes is the same as that of sigle lane. Therefore, the multi-lane transimission of the SpaceFibre protocol can enhance transfer rate in proportion. In contrast, the receiver gathers received words from every lane and re-construct the transmitted words from these receirved words.

Since the transfer rate of each lane does not depend on the number of used lanes, the operating frequency of lane layer is the same as that of single lane. However, upper layers such as quality layer and multi-lane layer require higher throughput than the lane layer. To achieve the higher processing speed, these layers should operate at higher frequency, or should expand internal data-bus of these layers in order to process multiple words simultaneously. For example, when 10 lanes are used and the transfer rate of each lane is 5.0Gbps which is the maximum number of lanes supported in the SpaceFibre protocol and the maximum transfer rate will be supported long-term, the former way needs 1.5625 GHz with 32-bit data-bus. It is difficult for aerospace devices to achieve this operating frequency. Therefore, it is necessary to adopt the latter way to implement the multi-lane transmission.

The required operating frequency of the latter way in the quality layer is the same as that for single lane transmission. Fig. 2 shows an example of data format of the SpaceFibre protocol. When N lanes are used, N words are located from the least significant word (LSW) to the most significant word (MSW). For the latter way, N words on a line in Fig.2 are processed simultaneously to increase the data throughput. Words located at LSW are sent over the lowest number lane, and words located at MSW are sent over the highest number lane as shown in Fig. 2.

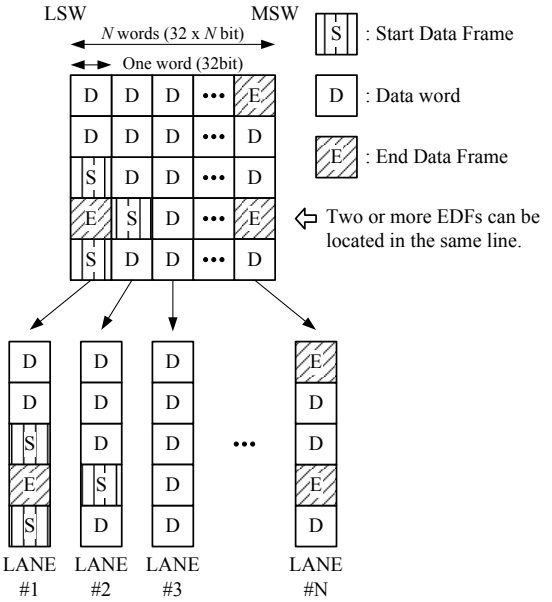

Fig. 2. Word placement for multi-lane transmission

The frame length of the SpaceFibre protocol is independent of the number of used lanes in the specification. Therefore, two or more EDFs can be located at the same line which should be processed at the same time. In this case, receiver should verify two or more data frames at the same time since these EDFs include frame sequence numbers and CRCs used for frame verification. However, it is difficult to process multiple data frames in parallel because sequential EDFs have order dependency in their frame sequence numbers. This makes the operating frequency lower in a processing device. This is a problem to implement the multi-lane transmission into the devices for aerospace since the operating frequencies of the devices are slow.

Additionally, multiple CRC calculators are needed to calculate CRCs of multiple data frames in parallel at the same time. Although these parallel circuits are not always required, they have to be prepared to calculate multiple CRCs only when multiple frames are in the same line. This makes the circuit area increase in a processing device. According to the reference [4], a circuit size of CRC-32 calculator with 256-bit data width becomes over 2000 LUT's. This complexity is approximately equal to a half of the SpaceFibre protocol using single lane transmission. Thus, it is not preferred to implement two or more large CRC calculators for the aerospace devices whose circuit area is smaller than commercial devices.

III. OUR FRAMING METHOD FOR MULTI-LANE TRANSMISSION

To solve the problems of multi-lane transmission described in chapter II, it is effective to limit the number of frames by one in N continuous words to be processed at the same time when N lanes are used. This avoids two or more frames verifying at the same time and calculating in parallel. Compared to the SpaceFibre protocol, this method can increase the operating frequency and reduce the circuit area.

A. Detail specification of our framing method

We propose to add/modify the following three rules into the specification of the SpaceFibre protocol for multi-lane transmission (see Fig. 3).

Rule 1: Quality layer inserts Fill words to make a frame length included one SDF and one EDF to a multiple of N if N lanes transmission is used, whereas the Fill word contains of four Fill codes (K27.7) in this paper. Thus, SDFs are located at LSW and EDFs are located at the MSW.

Rule 2: Multi-lane layer sends SDF, SBF, SIF and FCT over the lowest number lane (lane #1) \and sends EDFs over the highest number lane (lane #N).

Rule 3: Multi-lane layer sends a control word whose length is one word such as FCT, ACK or NACK over the lowest number lane (lane #1), and sends the Padding words over the rest lanes (lane #2-#N). The Padding word should be assigned a new K-code to distinguish a control word inserted at the multi-lane layer from the other control words inserted at the other layers.

By the first rule, two or more words which include the frame sequence number are not sent at the same time in multi-lane transmission, because the frame length is set to a multiple

of the number of used lanes. Therefore, the problem to verify two or more frame sequence numbers at the same time can be avoided. By the second rule, two frames are not sent at the same time. Therefore, the problem to implement two circuits such as CRC calculators to process two frames in parallel can be avoided. This second rule can reduce the circuit area in a processing device. By the third rule, for control words other than data frames, verifying multiple frame sequence numbers and calculating multiple CRCs at the same time can also be avoided because the rule makes one control word, such as FCT, ACK or NACK, to N words by inserting the N-1 Padding words. In the case, the Padding words should be removed in the multi-lane layer at the receiver side.

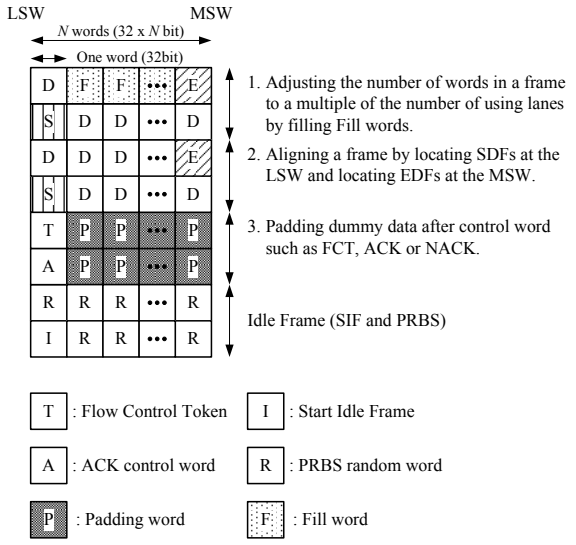

Fig. 3. Our framing method for multi-lane transmission

B. Optional rules for our framing method

Our framing method inserts the Fill words and the Padding words to limit the number of frames to be processed at the same time by one. Therefore, compared to the standard specification of the SpaceFibre protocol, data throughput of our method must be decreased by inserting these words. Thus, we add/modify the following two optional rules to the standard specification of the SpaceFibre protocol.

Optional Rule 1: Quality layer makes a frame length including one SDF and one EDF to the maximum number of a multiple of N which is less than or equal to 64 words for N lanes transmission.

Optional Rule 2: Minimum interval of sending ACK is 256ns in spite of the number of used lanes.

The first optional rule can reduce a number of the Fill words inserted by our framing method. In the specification of the SpaceFibre protocol, if there are 64 or more words of data without EOP in virtual channel buffer, only 64 words of data are read out from the buffer to make a data frame by adding a SDF and an EDF. Then, the frame length is 66 words which include 64 words of data, the SDF and the EDF. In our framing method, the Fill words should be inserted when 66 words is not a multiple of the number of used lanes. In other words, the data throughput is decreased by inserting the Fill words to make a data frame if the number of used lanes is not a submultiple of 66. By the first optional rule, the Fill words are not needed to make a data frame if there are data more than maximum frame length in the virtual channel buffer, except for last data frame. Then, the decrease of the data throughput by inserting the Fill words can be reduced.

The second optional rule can reduce the number of the Padding words inserted by our framing method. When N lanes is used with our framing method, one ACK word is sent over the lowest number lane and the N-1 Padding words are sent over rest lanes. Thus, compared with the standard specification, the bandwidth for sending ACK becomes multiples of N in our framing method. In the specification of the SpaceFibre protocol, the minimum interval of sending ACK is 16 words in order to restrict the increase of the bandwidth for sending ACK. This interval equals to 256 ns in a single lane transmission if data transfer rate is 2.0 Gbps. The bandwidth for sending ACK can be the same as the single lane transmission in the standard specification to apply this second optional rule to our method. Then, the decrease of the data throughput by inserting the Padding words can be reduced.

IV. Evaluation of Data Throughput with Our Method

Our framing method proposed in chapter III makes implementation of the multi-lane transmission easily by adjusting a frame length to the number of used lanes. On the other hand, the data throughput must be decreased because the Fill words and the Padding words are inserted to adjust a frame length. Therefore, we evaluate the data throughput with our framing method in computer simulations.

A. Simulation conditions

To measure the data throughputs of the SpaceFibre protocol and our methods, we built a C-language simulation environment. In our method, four packet transfer models, such as (1) without any optional rules, (2) with the first optional rule, (3) with the second optional rule, and (4) with the two optional rules, are evaluated. The packet is transmitted in one-way and the length of packet is set from 1 to 1024 words in random, and there is no error at any physical links (Bit error rate is 0). The throughput of data and control words are measured by counting the received data or the received control words.

B. Simulation results

Figure 4 shows that the usage ratio of the bandwidth which is occupied by data and control words when 10 lanes are used. The left circle graph shows the usage ratio with the standard specification of SpaceFibre Draft-F3 protocol, where the usage ratio of the data is 95.5%. But this is an ideal data throughput for multi-lane protocol since it is difficult to be implemented by actual LSI and FPGA for aerospace as pointed in chapter II. The right circle graph shows the usage ratio of our framing method without any optional rules described at chapter III. The data throughput is decreased from 95.5 % to 78.3% by the insertion of the Fill words and the Padding words.

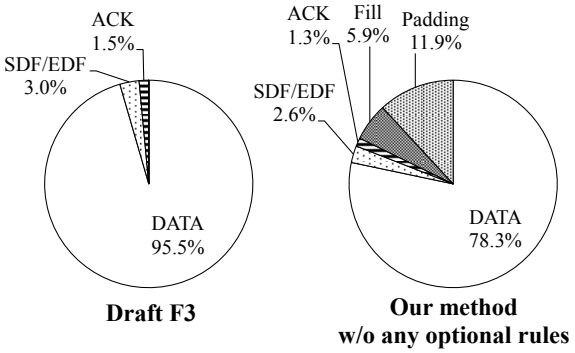

Fig. 4. Usage ratios of Draft-F3 and our method without any optional rules.

Next, the usage ratios of our methods with the first optional rule, with the second optional rule, and with two optional rules are shown in Fig. 5 when 10 lanes are used. In the case of our method with the first optional rule, the usage ratio of the Fill words can be decreaced from 5.9% to 0.7%. The usage ratio occupied by ACK or the Padding words is the same as the case without any optional rules. In the case of the second optional rule, the usage ratio of the Padding word can be decreaced from 11.9% to 4.3%. In addition, the usage ratio of the ACK words can be decreaces from 1.3% to 0.5%. Finally, in the case with two optional rules, the usage of data achieves about 91.8%, and the data throughput with two optional rules can increase from 78.3% to 91.8%.

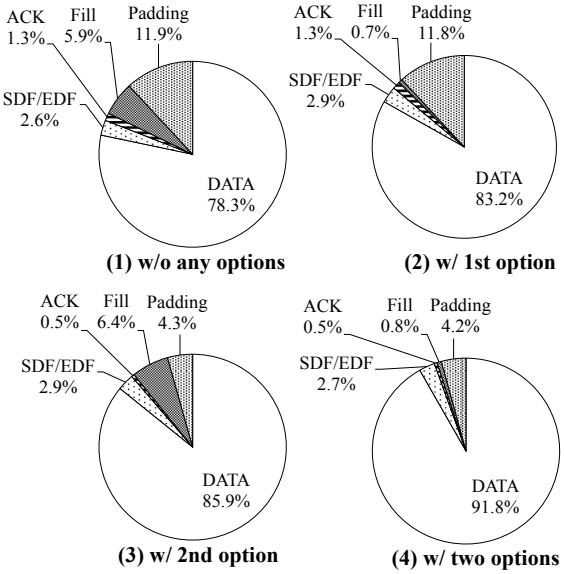

Fig. 5. Usage ratios of our method with/without optional rules

Next, maximum throughput of our methods are shown in Fig. 6 when a number of used lanes is changed. Figure 6 shows that the maximum throughputs of our methods are decreaced by insertion of the Fill words and the Padding words with the increasing of the number of used lanes. However, Figure 6 also shows that the optional rules can mitigate the deceasing of the data thoughput. Especially, the method with the two optinal rules can acheive over 90% data throughput in spite of a number of used lanes.

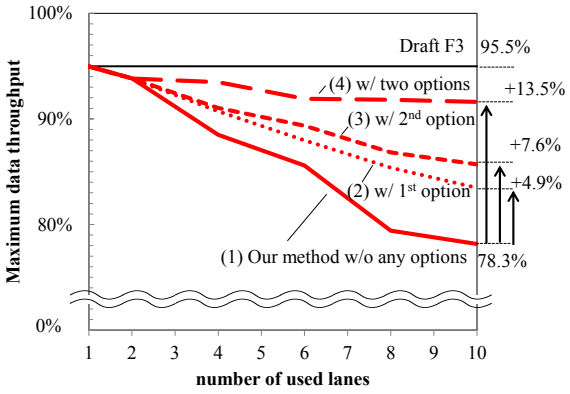

Fig. 6. Data throughput of our methods for the number of used lanes

V. SUMMARY

In order to make circuit implementation of the SpaceFibre protocol using multilane transmission easy, we studied the problems about multi-lane transmission and proposed the framing method to solve it.

In the specification of the SpaceFibre protocol, two or more EDFs can be received at the same time from different lanes. This makes it difficult to implement the SpaceFibre protocol into an aerospace device because sequential EDFs have order dependency and the aerospace device generally operates at low frequency. To solve this problem, we change a framing method for multi-lane transmission by adjusting the frame length to a multiple words of a number of used lanes, and by sending the first word of a frame from the lowest number lane an sending the last word of a frame from the highest number lane.

To evaluate the effect of our method, we measured the data throughput in simulation because the data throughput may be decreased by inserting the Fill words and the Padding words in our method to adjust the placement of data or control word. The simulation result shows the data throughput in our method without the optical rules is decreased by 18% compared with the case of the standard specification of the SpaceFibre protocol. However, the maximum data throughput can achieve over 90% at any number of lanes by applying two optional rules which changes the maximum frame length and the interval of sending ACK.

REFERENCES

[1] Steve Parkes, “SpaceFibre Standard Draft F3”, University of Dundee.

[2] Yu Otake, Kohei Hosokawa, Yasuhiro Sota, Takahiko Tanaka and Hiroki Hihara, “Performance evaluations and proposal to improve next-generation SpaceFibre protocol”, proceedings of international SpaceWire conference 2013, pp271-275.

[3] Steve Parkes, “A Radiation Tolerant SpaceFibre Interface Device”, proceedings of international SpaceWire conference 2013, pp123-128.

[4] M. Walma, “Pipelined cyclic redundancy check (CRC) calculation,” in ICCCN’07: Proceedings of 16th International ConferConference on ComputerCommunications and Networks, 2007, pp365-370.

SpaceFibre Based Spacecraft Network Case Study

Networks & Protocols, Poster paper

Yuriy Sheynin, Elena Suvorova, Nadezhda Matveeva
Saint-Petersburg State University of Aerospace Instrumentation
SUAI
Saint-Petersburg, Russian Federation
sheynin@aanet.ru, suvorova@aanet.ru, nadezhda.matveeva@guap.ru

Alexey Khakhulin, Igor Orlovsky, Dmitry Romanov
Rocket and Space Corporation Energia after S.P. Korolev
RSC Energia
Korolev, Moscow area, Russian Federation
Alexey.Hahulin@rsce.ru, Igor.Orlovsky@rsce.ru, Dmitry.Romanov@rsce.ru

Abstract*—SpaceFibre based spacecraft network is used as a case study of the real network, which SpaceWire based implementation was analyzed in [1]. The network structure has tree-based structure, includes dozens of different data sources (such as Telemetric System, Radiation Monitoring System, Control Block of Onboard Complex) and some data processing nodes (such as Central Computing Machine, Computer of the Engine Bay, Cosmonaut Consoles).*

We consider implementation of this network based on SpaceFibre technology with its advantages: quality of service mechanisms (guaranteed throughput, scheduling), possibility of galvanic isolation on physical layer.

For this example we evaluate parameters for SpaceFibre network implementation, compare the results with reachable parameters for SpaceWire based network. The analyzed network includes some typical data flows – video information, measurement information from different sensors, command information. For different data types flow parameters and timing requirements are essentially differ.

I. THE EXAMPLE: A FRAGMENT OF A SPACECRAFT NETWORK

Here we show an example of the real network which was analyzed with the simulator. This is a fragment of a spacecraft network shown in Fig. 1. The physical interconnections between components (terminal nodes and routers) are represented by the thick black lines. The thin black lines correspond to videotraffic transmission paths. This traffic is the main part of the communication system's load.

The marked by hatching terminal nodes are sources or destinations of command traffic of CBOC. The packet transmission time of this traffic is most important for the whole system correct functionality.

Detailed characteristics of the network traffic are given in Table 2. All abbreviations described in Table 1.

The videotraffic enters the network permanently. The traffic of other types is generated every 200 ms. The traffic between CBOC and CCM, OREC has highest priority. Every CBOC sends/receives one packet with data length 64 Bytes every 200 ms. Four CBOC are connected to every RRV router; five CBOC are connected to every REB.

Characteristics of the considered network should meet the following requirements:

1. Latency of packets between CBOCs and CCM is ≤ 10 ms.
2. Latency of video frames is ≤ 100 ms.

We evaluate the reachable characteristics for considered network fragment implementations based on SpaceWire, gigaSpaceWire and SpaceFibre standards:

- maximal packet transmission time from CBOC;
- videoframes transmission time;
- maximal available throughput.

In connection with the expected operating conditions for the SpaceWire network we consider the variant, in which the transmission rate is limited to 125 Mbit/s.

The variants with SpaceFibre and gigaSpaceWire transmission rates 1250 Mbit/s and 125 Mbit/s are reviewed also.

For simulation we use the routers with parameters:

- internal frequency of the router is 125 MHz;
- the width of switch matrix channels is 32 bit;
- the packet header transmission time is 7 clocks of processing frequency.

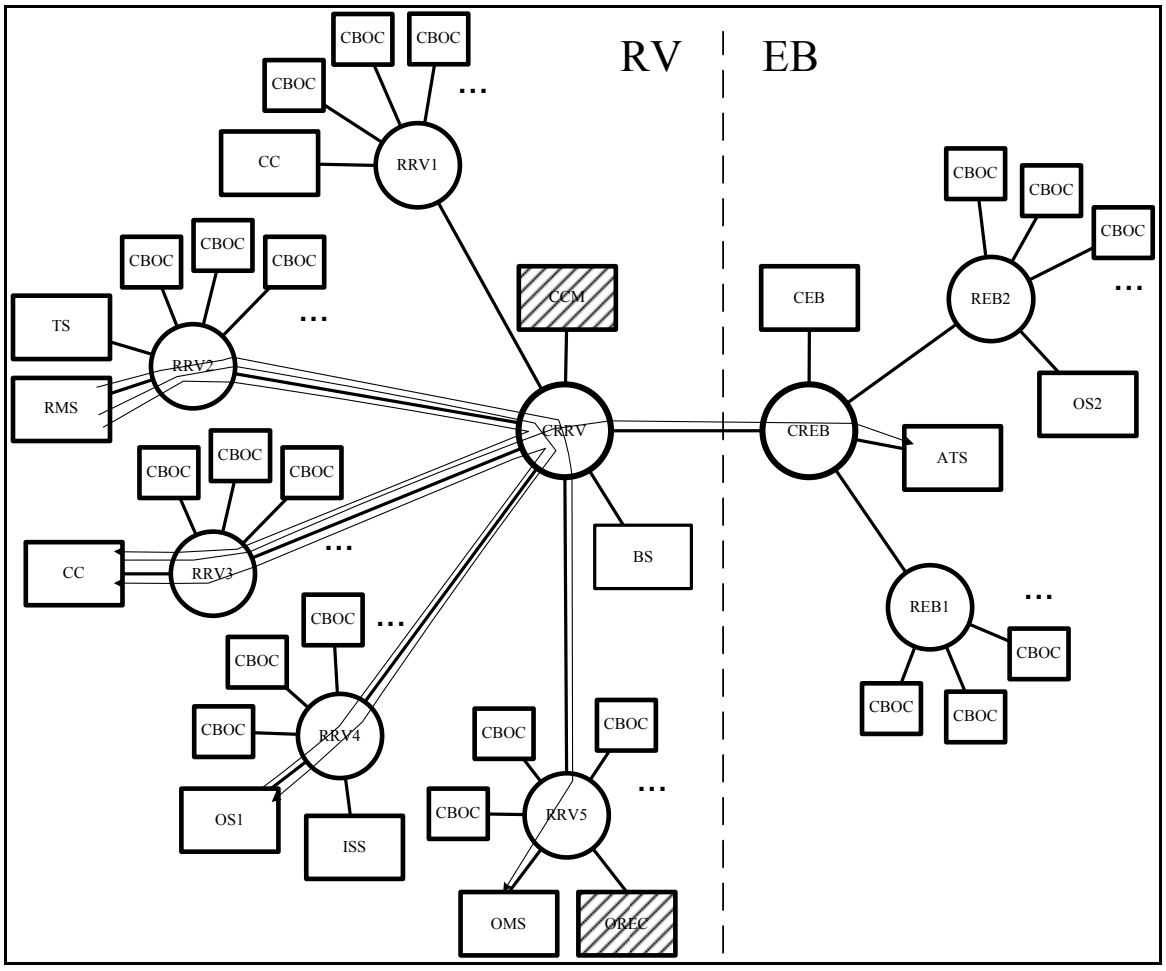

Fig. 1. Fragment of spacecraft network (rectangles are terminal nodes and rounds are routers)

TABLE I. ABBREVIATIONS

ATS	Automated Test System
OREC	Onboard Radio Engineering Complex
OS	Onboard Systems
CBOC	Control Block of Onboard Complex
RV	Re-entry Vehicle
EB	Engine Bay
CEB	Computer of the EB
RRV	Router of RV
REB	Router of EB
ISS	International Space Station
CC	Cosmonaut Consoles
OMS	Onboard Measurement System
RMS	Radiation Monitoring System
BS	Bearing System
CS	Communication System
TS	Telemetric System
CCM	Central Computing Machine
CRRV	Central RRV
CREB	Central REB

TABLE II. NETWORK TRAFFIC, IN MBIT/S

Senders \ Receivers	ATS	OREC	CBOC	CEB	ISS	CC	OMS	TS	CCM
ATS	-	-	-	-	-	-	-	-	-
OREC	-	-	-	-	-	2	-	-	5
OS	-	-	-	-	-	1	1	-	3
CBOC	-	-	-	-	-	-	0.5	-	0.08
CEB	-	-	-	-	-	0.1	2	-	2
ISS	-	-	-	-	-	25	3	-	2
CC	25x3	-	-	-	-	-	2	-	0.2
OMS	5	-	-	-	3	0.1	-	-	5
RMS	-	-	-	-	-	1	1	-	1
BS	-	-	-	-	-	-	0.1	-	-
CS	-	-	-	-	-	-	0.2	-	-
TS	-	25	-	-	25	25	0.1	-	0.01
CCM	5	5	0.08	2	2	5	1	0.01	-

II. THE TRANSMISSION TIME BETWEEN CBOC AND CCM, CBOC AND OREC

The path between CBOC and CCM includes 2 or 3 transit routers (depend on placement of CBOC in system). The path between CBOC and OREC includes from 1 to 4 transit routers.

Let's evaluate data packets transmission time form CBOC with maximal length for our example. The data packets length is 64 Bytes.

The minimal transmission time of these packets in SpaceWire network is 2,3 us when transmission rate is 400 Mbit/s and 6,4 us when transmission rate is 125 Mbit/s.

The minimal transmission time of these packets in SpaceFibre network is 2,2 us when transmission rate is 1250 Mbit/s and 14,8 us when transmission rate is 125 Mbit/s.

The minimal transmission time of these packets in gigaSpaceWire network is 1,3 us when transmission rate is 1250 Mbit/s and 6,9 us when transmission rate is 125 Mbit/s.

These values are essentially less than user constrains (10 ms). But this transmission time is reachable only in case when data packet from CBOC doesn't wait the output ports in transit routers.

Let's evaluate the maximal data packet transmission time form CBOC to OREC in SpaceWire, gigaSpaceWire and SpaceFibre network with transmission rate 125 Mbit/s.

The data packets from CBOC have the highest priority layer, all CBOC send its packets practically in the same time. Therefore in worst case the data packet from CBOC should wait in output port queue of transit routers until all packets from other CBOC, which goes along the same path, and one packet with low priority (that transmission can happen to start before the first packet from CBOC goes to the router) would be transmitted. For considered data path this low priority packet in worst case would be videotraffic packet from RMS to OMS.

Figure 2 shows the dependence between the maximal data transmission time (CBOC ->OMS) and the low priority traffic (videotrffic) packet length. We consider the packet lengths from 512 to 4096 Bytes.

For the SpaceFibre network we evaluate transmission time for network variations with different quantity of CBOC: from 30 to 60. In SpaceFibre the data transmission time for high priority traffic depends not from the low priority traffic packet length but from the frame length for SpaceFibre network. Thus the corresponding to SpaceFibre diagrams, are straight lines that go parallel with the X axe.

These graphs show, that in all cases the transmission time is less than 10 ms (the user defined constraint). The data transmission time for SpaceWire and gigaSpaceWire network is less than for SpaceFibre, when the low priority traffic packet size is 2 – 4 times bigger than the SpaceFibre frame size.

III. EVALUATION OF MAXIMAL VIDEOFRAMES DENSITY FOR SPACEFIBRE AND GIGASPACEWIRE NETWORKS

Let's evaluate the maximal reachable videoframes density for considered example with SpaceFibre and gigaSpaceWire based networks.

In this network video data goes counter each others, Fig. 1, Table 1. Therefore in each direction of SpaceFibre network should be translated one video data flow and ACK (NACK) and FCT flows from video data flow that goes to opposite direction.

In each direction of gigaSpaceWire network the video data flow and FCT for It go in opposite directions.

In the SpaceFibre network video data flow can be transmitted with big packets (packet size equal to video frame size – from 1 to 2 Mbytes). These packets would be sliced in a data link into frames with maximal size (256 Nchars). On each such frame will have to transfer one ACK and one FCT from an opposite video flow. Thus the overhead of data transmission are 5,88 %. So the maximal reachable throughput for video data flow is 941 Mbit/s.

We now consider the transfer of the same traffic by the gigaSpaceWire network. We assume that for video flow

transmission packets with data field size 256 Nchars and network header size 1 Nchar are used. This packet size is selected to provide required transmission time for high priority traffic.

In the gigaSpaceWire standard the credit size can be adjusted. We assume that the size of one credit is 128 Nchars (the maximal possible value). The size of the Credit command is 1 K-Code.

For these parameters the overhead is 1,53 %. Correspondingly the maximal reachable throughput for videotraffic transmission is 984 Mbit/s.

When the videoframes transmission path does not meet with command traffic, and therefore we can transmit videoframes with big packets. If we transmit frames by packets with 1 Mbyte size, the overhead is 0,78 %. For packets with 2 Mbytes size the overhead is 0.77 %.

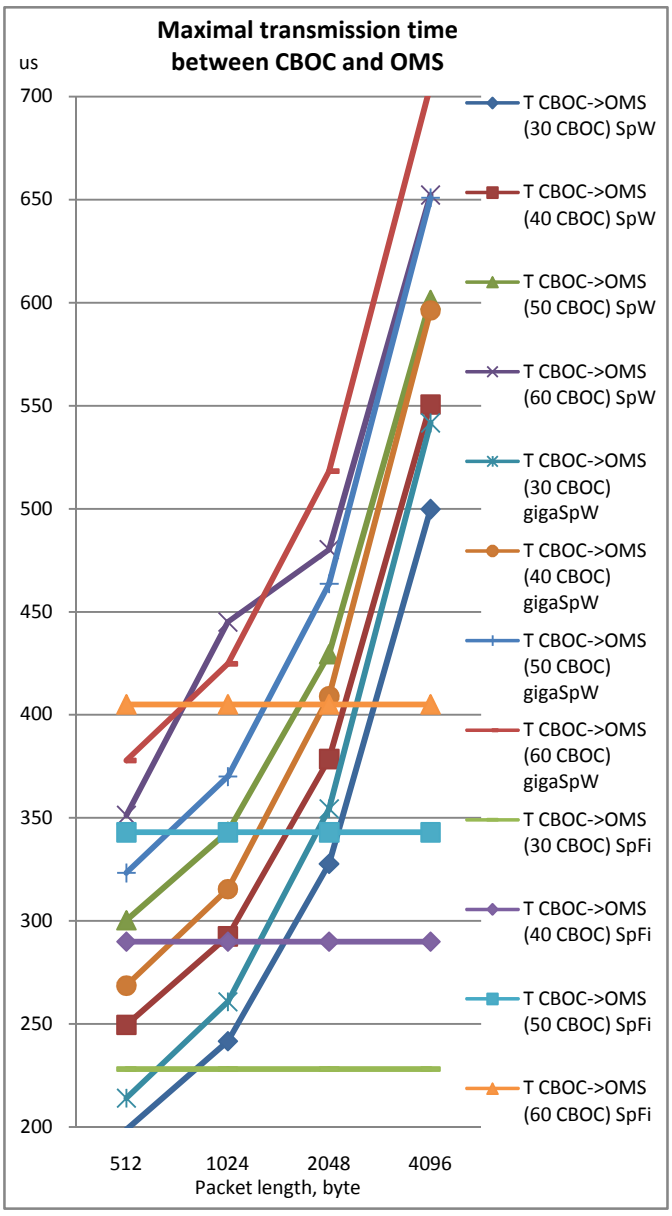

Fig. 2. Dependency between the data packets length and the maximal transmission time between CBOC and OMS (videostream)

IV. Evaluation of the Videoframes Transmission Time

To evaluate transmission time of the big data objects – videoframes with size 1 Mbyte, 1.7 Mbytes, 2 Mbytes for SpaceFibre and gigaSpaceWire network with transmission rate 1250 Mbits/s consider that in gigaSpaceWire network the videoframe is divided to packets with 256 Nchars data field size.

The graph (Fig. 3) of dependency between the videoframe transmission time and its size for gigaSpaceWire and SpaceFibre network when other traffic isn't transmitted via the network.

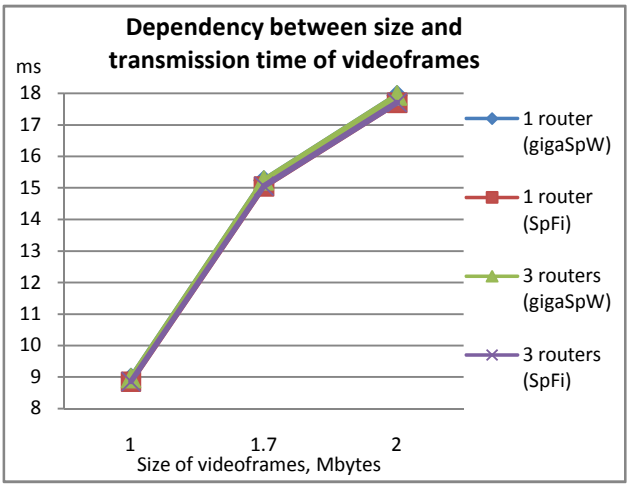

Fig. 3. Dependency between the size of videoframes and its transmission time in the network that includes 1 or 3 routers

This figure shows that parameters for gigaSpaceWire and SpaceFibre are practically same.

With transmission rate 1250 Mbit/s the transmission time of biggest videoframes (2 Mbytes size) is less than 18 ms.

Let's evaluate transmission time of videoframes in the considered network with command traffic from CBOC when the videotraffic and traffic from CBOC are crossed in the CRRV router.

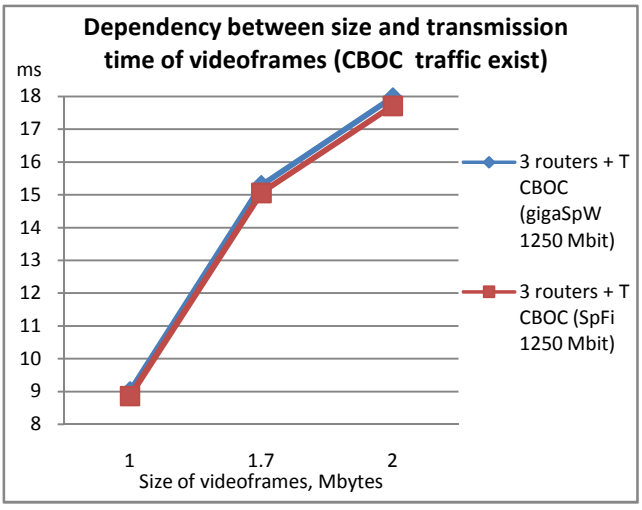

Fig. 4. Dependency between the size of videoframes and its transmission time in network with 3 routers and traffic from CBOC

In the gigaSpaceWire network QoS mechanisms are not provided. But the curves in the Fig. 4 show, that the videoframes transmission time is practically the same for gigaSpaceWire and for SpaceFibre.

It was reached thanks to the traffic logical organization: the videoframes were divided into packets with data field sizes equal to the SpaceFibre maximal frame size.

The number of terminal nodes in this example is less than 224. Therefore the SpaceWire address field size of these packets is 1 for this example.

V. Conclussion

In this article we consider different implementations of the onboard network: the SpaceWire based network, the SpaceFibre based network and the gigaSpaceWire based network. We estimated reachable timing parameters and throughput for different traffic types in these variants of network implementation.

SpaceFibre with its QoS features provides ready-made backbone networking for spacecrafts with mixture of high-priority command traffic and low priority intensive streaming data (videostreams as an example); the user packet delivery latency constraints are consistently met for both traffic classes.

Unlike SpaceFibre, SpaceWire and gigaSpaceWire network shave no QoS support mechanisms. But we show that for SpaceWire and gigaSpaceWire network the same timing characteristics as for SpaceFibre can be reached when packet sizes for low-priority traffic is selected correctly. The packet delivery latency constraints are consistently met also. For more complicated mixture of traffic types, stricter latency constraints and priority requirements SpaceWire and gigaSpaceWire networks could be supplied with QoS features at the Transport layer.

Acknowledgment

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under contract n° 14.578.21.0022.

References

[1] A. Eganyan, L. Koblyakova, E. Suvorova. "SpaceWire network simulator," in proceedings of the 3rd International SpaceWire conference, St.Petersburg, Russia, 2010, pp. 403-406.

[2] L. Koblyakova, Yu. Sheynin, D. Raszhivin. "Real-Time Services in Networked Embedded Systems," in 7th Conference of Open Innovations Framework Program FRUCT, SPb, Russia, 2010, pp. 64-67.

[3] A. Eganyan, E.Suvorova, Y. Sheynin, A. Khakhulin, I. Orlovsky, "DCNSimulator – Software Tool for SpaceWire Networks Simulation," in proceedings of the 5th International SpaceWire Conference, Gothenburg 2013, pp. 216-221.

The Remote Virtual-Channel Transfer Protocol

SpaceWire Networks and Protocols, Short paper

Wang Zhen
No.1 Dep.
Shanghai Institute of Satellite Engineering-SISE
Shanghai, China
simonlover121@163.com

Dong Yaohai
Science and Technology Dep.
Shanghai Aerospace Bureau
Shanghai, China
dongyaohai@hotmail.com

***Abstract*—The Remote Virtual-Channel Transfer Protocol (RVTP for short) is proposed to transfer CCSDS AOS Virtual Channel Frames across a SpaceWire Network. In order to transfer more efficiently and reliably onboard data in large volume and at high data rate, RVTP is designed to encapsulate a CCSDS AOS Virtual Channel Frame into a SpaceWire packet at the initiator node. At the target node, RVTP provides error detection and handling services.**

***Keywords*—SpaceWire RVTP, CCSDS, FY-4 Series Mission.**

I. INTRODUCTION

RVTP is designed to encapsulate a CCSDS AOS Virtual Channel Frame into a SpaceWire packet [1] which is transferred from an initiator to a target across a SpaceWire network. The RVTP provides error detection as a received packet can be checked if it complies with the protocol at the target. But it does not provide any means for ensuring successful delivery of the packet, neither is it responsible for the content of the packet being a CCSDS AOS Virtual Channel Frame [2].

Fig. illustrates the location of the RVTP in a typical onboard protocol stack. The RVTP provides a unidirectional data transfer service from a single source user application to a single destination user application through a SpaceWire network.

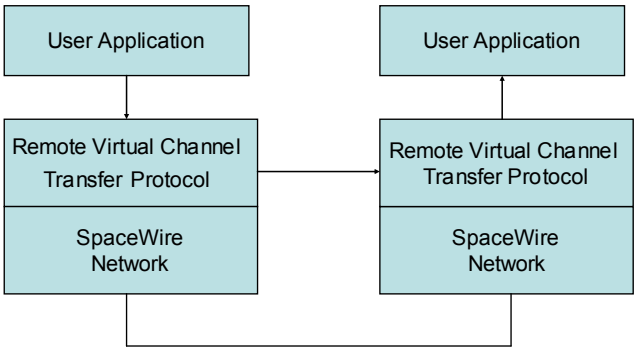

Fig.1. Protocol configuration

II. PROTOCOL FEATURES

RVTP is suitable for high-speed data transferring between remote instruments and communication system. It provides the capability to transfer AOS virtual channel frames with fixed size between end nodes of a SpaceWire network. When a RVTP SpaceWire packet is received at the target, error detective mechanism works. If the AOS virtual channel frame is right, it will be extracted, and a synchronous header (1ACFFC1D) is inserted to form a Channel Access Data Unit (CADU) which can be transferred in communication system.

There are three main innovations of RVTP shown as follows:

- The RVTP is based on Virtual Channel which is firstly proposed.
- The RVTP packets are with fixed length which makes the data transfer delay predicable in a SpaceWire network.
- The RVTP provides FDIR function at the target node to facilitate fault location and recovery autonomously.

III. SERVICE PARAMETERS

The RVTP provides users with data transfer services. The point at which a service is provided by a protocol entity to a user is called a Service Access Point. A Service Access Point of the RVTP is identified by a SpaceWire logical address and each service user is also identified by a SpaceWire Logical Address.

Implementations may be required to perform flow control at a Service Access Point between the service user and the service provider. However, the RVTP does not recommend a scheme for flow control between the user and the provider.

The end-to-end quality-of-service provided to service users is the one that is provided by the underlying SpaceWire network. The RVTP does not provide any mechanisms for guaranteeing a particular quality-of-service; it is the responsibility of implementing organizations to ensure that the end-to-end performance of a particular service instance meets the requirements of its users.

The service parameters are as follows.

A. Virtual Channel Frame

The Virtual Channel Frame parameter, intended as the service data unit transferred by the Remote Virtual Channel Transfer service, shall be the AOS Virtual Channel Frame.

B. Frame length

The value of the Virtual Channel Frame length shall be of a fixed size.

The frame length is selected depending on the bandwidth of communication channel. Users can select a suitable frame length for a particular mission.

C. Status code

The Status code parameter shall be used to indicate the validity of the Virtual Channel Frame to the receiving service user. It shall take one of the following values [4]:

- 0x00 indicates that the Virtual Channel Frame is ok;
- 0x01 indicates Virtual Channel Frame arrived terminated by EEP;
- 0x02 indicates Channel ID was illegal.

D. Target SpaceWire Address

The Target SpaceWire Address parameter shall be used to define the path to the Target when SpaceWire path addressing is being used.

E. Target Logical Address

The Target Logical Address parameter shall be used to define the logical address of the Target that is to receive the Virtual Channel Frame.

F. Virtual Channel Data Unit

The RVTP packet contains an integrated Virtual Channel Data Unit (VCDU) with that is defined in [2].

VCDU, shown in Fig.2, is contained by VCDU Header and VCDU Data Zone.

VCDU Header (6 Octets)								VCDU Data Zone	
Channel ID			VC Frame Count	Signaling Field				Frame Insert Zone	Frame Data Field
Frame Version Number	SCID	VCID		Replay Flag	VC Frame Count Usage Flag	RSVD. Spare	VC Frame Count Cycle		
2bits	8bits	6bits	3 Octets	1bit	1bit	2bits	2bits	16bits	Fixed Length

Fig.2. Virtual Channel Data Unit

VCDU shall follow, without gap, the Protocol Identifier.

G. Bitstream Protocol Data Unit

The RVTP uses Bitstream Protocol Data Unit (B_PDU), which is defined in CCSDS 732.0-B-2 AOS Space Data Link Protocol, to form Virtual Channel Data Units.

B_PDU, shown in Fig.3, shall be divided by B_PDU Header and B_PDU Bitstream Data Zone.

B_PDU Header		B_PDU Bitstream Data Zone
RSVD. Spare	Bitstream Data Pointer	
2bits	14bits	Fixed Length

Fig.3. Bitstream Protocol Data Unit

B_PDU shall follow, without gap, the VCDU Header.

IV. Protocol Format

The complete format of the RVTP packet is shown in Fig.4.

First byte transmitted

	Target SpW Adress	·····	Target SpW Adress
Target Logical Adress	Protocol Identifier	Channel ID (MS)	Channel ID (LS)
VC Frame Count (MS)	VC Frame Count	VC Frame Count (LS)	Signaling Field
Frame Insert Zone (MS)	Frame Insert Zone (LS)	B_PDU Header (MS)	B_PDU Header (LS)
B_PDU Bitstream Data (First byte)	B_PDU Bitstream Data	B_PDU Bitstream Data	B_PDU Bitstream Data
B_PDU Bitstream Data	·····	·····	B_PDU Bitstream Data
B_PDU Bitstream Data	B_PDU Bitstream Data (Last byte)	EOP	

Last byte transmitted

Fig.4. RVTP packet format

A. Target SpaceWire Address field

The Target SpaceWire Address field shall comprise zero or more data characters forming the SpaceWire address which is used to route the RVTP packet to the target.

SpaceWire path addressing and regional addressing may be used.

The Target SpaceWire Address field shall not be used when a single logical address is being used for routing the Virtual Channel frame to the target.

B. Target Logical Address field

The Target Logical Address field shall be an 8-bit field that contains a logical address of the target.

- The Target Logical Address field is normally set to a logical address recognised by the target.
- If the target does not have a specific logical address then the Target Logical Address field can be set to the default value 254 (0xFE).
- A target can have more than one logical address, but a logical address indicates one target, in other words, different target has different logical address in a SpaceWire Network.

C. Protocol Identifier field

The Protocol Identifier field shall be an 8-bit field that contains the Protocol Identifier complied with the provisions of the related ECSS standards [3].

D. Channel ID field

The Channel ID shall be a 16-bit field that contains Frame Version Number, Spacecraft ID(SCID), Virtual Channel ID(VCID).

- The Frame Version Number shall be a 2-bit field that identifies the data unit as a VC Transfer Frame; it shall be set to '01'.
- The Spacecraft Identifier shall be an 8-bit field that is assigned by CCSDS and provide the identification of the spacecraft which is associated with the data contained in the VC Transfer Frame.
- The Virtual Channel Identifier shall be a 6-bit field that is used to identify the Virtual Channel.

E. VC Transfer Frame Count field

The Virtual Channel Transfer Frame Count shall be a 24-bit field which contains a sequential binary count (modulo-16,777,216) of each Transfer Frame transmitted within a specific Virtual Channel.

The purpose of this field is to provide individual accountability for each Virtual Channel, primarily to enable systematic Packet extraction from the Transfer Frame Data Field.

F. Signaling field

The Signaling shall be an 8-bit field that contains Replay Flag, Virtual Channel Frame Count Cycle Use Flag, Reserved Spares, Virtual Channel Frame Count Cycle.

The Replay Flag shall be a one-bit field that is to discriminate between real-time and replay Transfer Frames when they both may use the same Virtual Channel. When it is set to '0', it means that it is a real-time Transfer Frame, otherwise, it is a Replay Transfer Frame.

The Virtual Channel Frame Count Cycle Use Flag shall be a one-bit field that indicates whether the VC Frame Count Cycle field is used. When it is set to '0', it means that the VC Frame Count Cycle field is not used, otherwise, it means that the VC Frame Count Cycle field is used.

The Reserved Spare shall be a 2-bit field that is reserved for future definition by CCSDS and shall be set to '00'.

The Virtual Channel Frame Count Cycle shall be a 4-bit field. If used, the Virtual Channel Frame Count Cycle Use Flag shall be set to '1'. Each time the Virtual Channel Frame Count returns to zero, the VC Frame Count Cycle shall be incremented. If not used, the Virtual Channel Frame Count Cycle Use Flag shall be set to '0' and this field shall be set to 'all zeros'.

G. Frame Insert Zone field

The Frame Insert Zone shall be a 16-bit field that can be used to insert some special information according to user application, such as time, secret key.

The Frame Insert Zone shall exist in every Transfer Frame transmitted within the same Physical Channel, including Idle Transfer Frames.

H. B_PDU Header Field

The B_PDU Header shall be a 16-bit field that contains Reserved Spare and Bitsream Data Pointer.

The Reserved Spare shall be a 2-bit field that is currently undefined by CCSDS; by convention, it shall therefore be set to the reserved value of '00'.

The Bitsream Data Pointer shall be a 14-bit field that discriminates between idle user data and valid user data within B_PDU Bitstream Data field. The locations of the bits in the B_PDU Bitstream Data field shall be numbered in ascending order. The first bet in this filed is assigned the number 0. The Bitstream Data Pointer shall contain the binary representation of the location of the last valid user data bit within B_PDU Bitstream Data field.

I. B_PDU Bitstream Data Field

The B_PDU Bitstream Data Field shall be fixed-length that follows, without gap, the B_PDU Header.

The Bitstream Data field shall contain either a fixed-length block of the user Bitstream Data (possibly terminated with idle data at a location delimited by the Data Pointer), or Idle Data (a fixed-length project-specified 'idle' pattern).

J. EOP character

The end of the RVTP packet shall be indicated by an EOP character.

V. PROTOCOL ACTION

The normal sequence of actions for a RVTP packet transfer is illustrated in Fig.5.

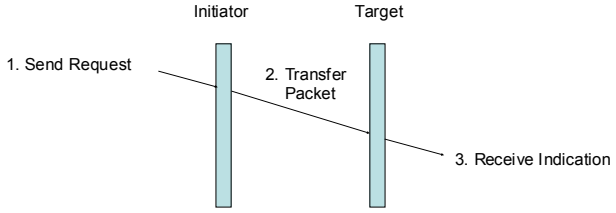

Fig.5. RVTP Packet Transfer

A. Send request

The RVTP packet transfer shall begin when an initiator user application requests to send a RVTP packet (Send Request).

The initiator user application shall pass the following information to the initiator:

- Target SpaceWire Address
- Target Logical Address
- Channel ID
- VC Transfer Frame Count
- Signalling field
- Frame Insert Zone
- B_PDU

B. Transfer packet

In response to the send request the initiator shall encapsulate the Virtual Channel Frame into a SpaceWire packet as described in Part IV and send it across the SpaceWire network to the target (Transfer Packet).

C. Receive indication

When a RVTP SpaceWire packet is received at the target, error detection and recovery mechanism works as follows.

- Protocol identifier error. When a SpaceWire packet is received at the target and the Protocol Identifier field is not indicated to be a RVTP packet, the packet shall be discarded.
- Channel ID error. If the Remote Virtual Channel Transfer Protocol packet arrives at the target with the Channel ID field set to an invalid value (the value is not expected and pre-assigned), the target user application should be informed that there is a Channel ID Error at

the target. In this case, the Virtual Channel Frame shall be extracted from the SpaceWire packet and passed to the target user application.

- Frame Count discontinuous error. If the Virtual Channel Frame Count received at the target is not sequential within a specific Virtual Channel, the target user application should be informed that there is a Frame Discarded Error with the specific Channel ID. In this case, the Virtual Channel Frame shall be extracted from the SpaceWire packet and passed to the target user application.
- Virtual Channel Frame Length error. If the RVTP packet arrives at the target with the Virtual Channel Frame Length shorter than the predesigned value, the target user application should be informed that there is a Shorter Frame Error with the specific Channel ID. In this case, the Virtual Channel Frame shall be extracted and inserted idle data subsequently till the Frame Length equal to the predesigned value. Otherwise, If the length longer than the predesigned value, the target user application should be informed that there is a Longer Frame Error with the specific Channel ID. In this case, the Virtual Channel Frame shall be extracted till the Frame Length equal to the predesigned value and the redundant data shall be discarded.

VI. Conclusion

RVTP has been applied in FY-4 series mission which is the first Chinese Space mission using SpaceWire for onboard data transfer in China. Through mass of engineering tests, RVTP is proved to be more efficient and reliable. Besides, it can make a unified design at the target.

References

[1] ECSS-E-ST-50-12A, Space Engineering - SpaceWire - Links, nodes, routers and networks. 24 January 2003.

[2] CCSDS 732.0-B-2, AOS Space Data Link Protocol. Blue Book. July 2006.

[3] ECSS-E-ST-50-51C, Space Engineering - SpaceWire protocol identification. 5 February 2010.

[4] ECSS-E-ST-50-53C, Space Engineering - SpaceWire - CCSDS packet transfer protocol. 5 February 2010.

Active Optical Cable for SpW applications

Poster session, Short Paper

Julián Blasco, Olga Navasquillo, David Cano
DAS Photonics
Valencia, Spain
jblasco@dasphotonics.com

Mª Angeles Esteban
Airbus Defence and Space
Madrid (Spain)

***Abstract*— DAS Photonics and Airbus Defence and Space (Spain) have been working for more than six years in the concept of an Active Optical Cable (AOC) for copper SpaceWire cable substitution. The main advantages that AOC offers are significant mass and size saving, better flexibility and routing of the cable and immunity to EMI.**

***Index Terms*—Active Optical Cable, low mass, SpaceWire, high speed, fiber optic**

I. Introduction

Communication harness constitutes a major part in mass and volume of current satellite and onboard equipment. The main problems of the typical copper or coaxial cables used are the high mass and some problems derived of the technology, like low immunity to EMI or difficult to be routed. In order to assure the harness to be free of EMI or noise, it is needed to increase the diameter of the cables, increasing in consequence the volume/mass.

The limitations of the copper cables, as base for satellite harnessing, are used as main arguments to switch the actual technology from copper to optical fibre to be used for payload and potentially platform applications.

The first and most obvious benefit of harness reduction is a saving in the mass of the spacecraft. This could reduce launch cost significantly, may make the spacecraft easier to balance prior to launch, and reduces the fuel required to manoeuvre the spacecraft after launch. Moreover, harness mass savings could allow additional payloads to be flown, increasing the spacecraft capability.

Another benefit of optical cables is a decrease in the cable diameter, making it easier to route through the spacecraft. In addition, it does not cause or is affected by EMI and avoids ground loops.

DAS Photonics and Airbus Defence and Space have been working together developing an opto-electronic conversion module to use fibre optic without impacting the current IF elements in on-board equipment.

The first demonstration of the technology was performed in the Spanish Space Program and the next steps done consisted in two in-orbit validations to verify the suitability of the technology under real space conditions. Finally a GSTP was executed where first Active Optical Cables (AOC) for SpaceWire (SpW) were developed.

Mainly, an AOC consist in two transceivers that manages the electro-optical conversion of equipment data, being connected using fiber optic.

In this paper are presented the main tasks performed on the design and technology verification, as well as related results to date.

II. First Developments

The first works were focused in the validation of the optical technology intended to be used in the opto-electronic conversion modules for digital communications. The initial developments consisted in a set of optical transceivers to fit low and medium signal speed:

- **Low Speed:** this solution, with a maximum data rate of 10Mbps, covers all control buses such as MIL-STD-1553 and CAN. Also is suitable to substitute other low speed links such as TM/TC signals or even low speed clocks.
- **Medium Speed:** this solution, with a maximum data rate of 500Mbps, covers all SpW data links (with low skew/jitter) usually used from 100 to 400 Mbps. Also is suitable to substitute other medium speed such a clocks or commands.

Due to the lack of qualified optical components [1], and in order to minimize the size and mass, commercial components were used in the design of the optical transceivers.

The developed models were submitted to several environmental and mechanical tests in order to validate the suitability of the technology for space use [2].

Fig. 1. First Active Optical Cable developed by DAS

III. In-Orbit Validations

The successful and promising results obtained in the first developments brought DAS two opportunities to validate in orbit a test bed of the AOCs.

First flight opportunity raised under the frame of the TDP8 project, framed in the Alphasat mission, where DAS delivered a flight optical board with four optical transceivers for digital communications. The experiment allowed the demonstration of the performance of 4 optical links working @1Mbps and 4 optical links working @100Mbps.

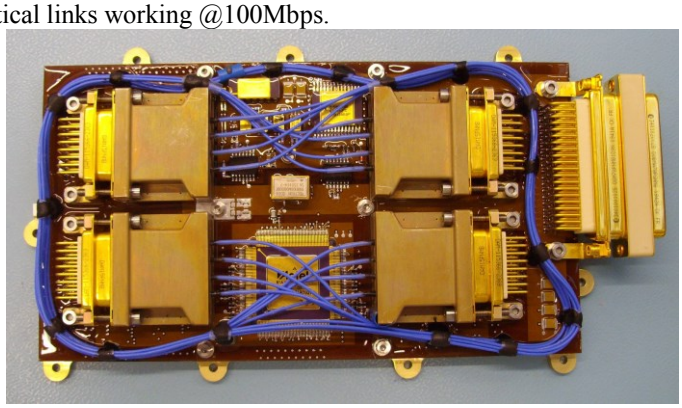

Fig. 2. Optical experiment boarded on TDP8 for Alphasat

As part of the in-orbit validation activities, the non space qualified components were submitted to a complete space assessment campaign with good results.

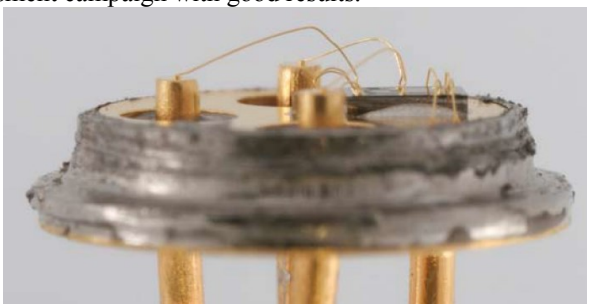

Fig. 3. Constructional analysis of photonic component

Although the satellite suffered some delays, finally was launched in 2013. At this moment DAS is receiving telemetry from the experiment with no detected errors in the optical links or visible degradation in their performance.

The second flight opportunity was in Proba-V satellite, where DAS and T&G Elektro developed a test bed to validate MTP connectors and multi-fibre cables. Thanks to the good results during TDP8 activity, T&G trusted DAS to design and manufacture an experiment that allowed both companies to demonstrate the feasibility of the technology for future space applications.

This flight opportunity consisted in single equipment with 4 optical channels SpW compatible working at 100Mbps and interconnected through two MTP connectors. Each optical channel was configured with different power margin between transmitter and receiver in order to check the complete losses (including MTP connector) in the channel.

No errors have been detected in any channel except for the channel_2, but since the distribution is centered in each equipment switch on, it seems that the error is produced due to a bad start of the equipment. The start procedure was changed on flight, and no errors were detected beyond this change.

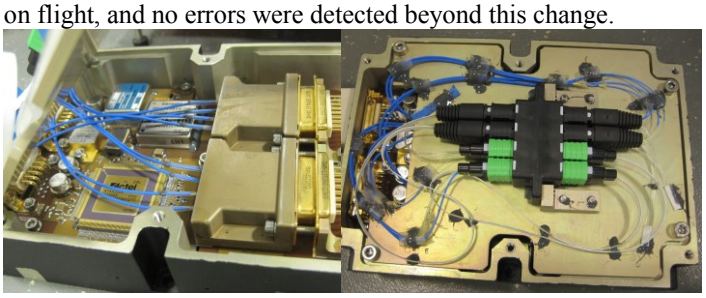

Fig. 4. Optical experiment boarded on Proba V

This experiment was executed within a very stringent schedule of 6 months. The Proba-V satellite was launched in April 2013. Collected data from the experiment telemetry allows to have more than one year of results producing representative BER information of the optical channels.

TABLE I. 1 year Proba-V experiment BER

Optical channel @100Mbps	BER values
1	1.86E-16
2 (max. power margin)	3.43E-12
3	1.86E-16
4 (min. power margin)	1.86E-16

IV. AOC for SpaceWire

With the information collected from previous activities DAS started a GSTP focused on the development of an AOC for buses and point-to-point protocols being SpaceWire one of the target applications.

Since SpW is a protocol well known and used in intra-satellite communications [3], the key points of the new AOC design was to improve such points where optic fiber has strong advantages against copper, minimizing the impact on the equipment in terms of power consumption and signal integrity. Another key point was the reduced electrical connector used in SpW. The uD-9 has very low profile and this fact constrained the mechanical design of the AOC in order to have the same size than the uD9 connector plus the backshell.

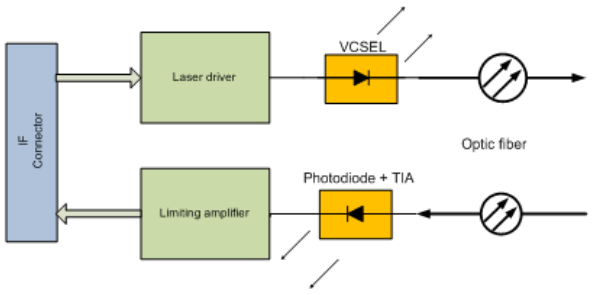

Fig. 5. SpW AOC transceiver block diagram

During GSTP execution not only the design of the AOC was performed, also a devoted components assessment campaign complementary to previous ones was executed, including life test and constructional analysis.

The GSTP activity was divided in following stages:

- Requirements identification
- Detailed Design of the optical transceivers
- Components assessment for non-qualified EEE
- Manufacturing of the AOCs
- Test campaign at module level

The results of test campaign as well as the summary of OAC performance will be presented in following paragraphs.

V. Component Assessment

Since photonic and some EEE parts used in the optical transceivers are non space qualified because of the lack of available parts for these technologies, a components assessment was needed to be performed. The obtained results along with the previous information from other activities, the viability of the use of the parts will be determined for future missions.

Previous data results from other activities were used to complement the GSTP components assessment:

- Outgassing and residual gas test (Previous data;P)
- Catastrophical Optical Damage (P)
- Thermal Vaccuum Cycling (P)
- Heavy Ions (P)
- Life test (new test)
- Thermal Conductance analysis

The life test of 1000 hours @85ºC was performed on 20 samples of each component type used in the AOC. No major degradation was observed during the life test and all components survived to the test.

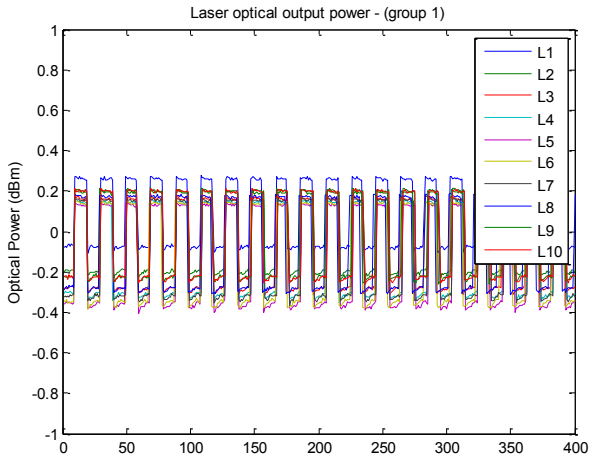

Fig. 6. Life test, output optical power of first set of 10 lasers

Also, a verification programme of the soldering process for the assembly of micro-D connector to the board inside the transceiver of the AOC was performed. Since in order to optimize the size of the optical transceiver an approach not supported by ECSS[4][5]was needed to be used.. This verification produced successful results and the process was validated for this application.

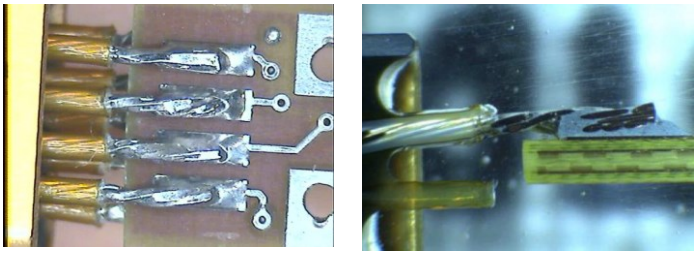

Fig. 7. uD9 soldering assement

VI. SpW AOC Test Campaign

Four complete SpW AOCs were manufactured and submitted to the test campaign. This test campaign was performed using representative values and profiles on all test in order to cover as much as typical application cases as possible for future flight missions.

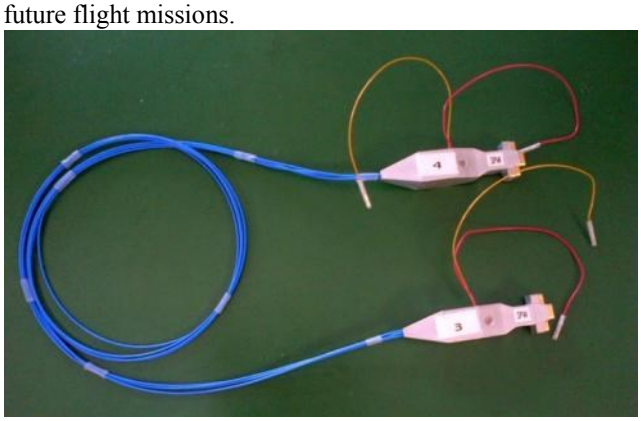

Fig. 8. SpW AOC

Test campaign at AOC level included the following tests[6]0:

- Functional tests
- Mechanical tests (vibration and shock)
- TVC: 8 cycles -40/85ºC
- TID: 150Krad @ 360rads/min
- Single Events (protons): 60,100,200MeV

None of the tests comprised in the test campaign produced destructive or detectable degradation on the performances of the AOC.

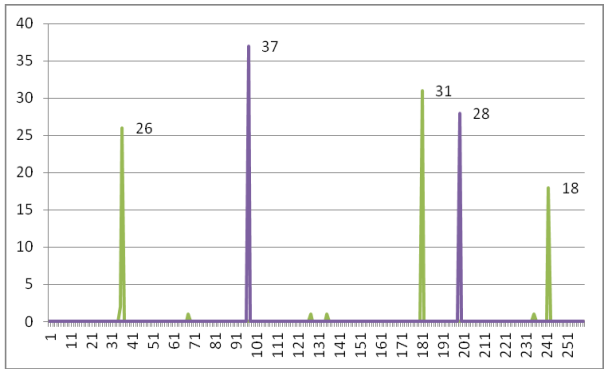

Fig. 9. Errors distribution at 200MeV, flux 1E8 p/cm2/s, fluence 1E11 p/cm2

For protons irradiation some errors were detected due to single events. This experimental data allowed inferring future

behavior in flight mission extracting the expected BER figure. For a GEO mission (15 years) the expected BER will be from 3.6E-18 to 1.7E-16 and for a LEO mission (8 years) the expected BER will be from 5E-18 to 1E-15. Both values are much lower than the requirement for a SpW communication, 1E-12.

Radiation tests such as gamma and proton were performed upon the transceivers with and without the mechanical package in order to measure the different behavior of the AOC. No important effects were measured in both tests.

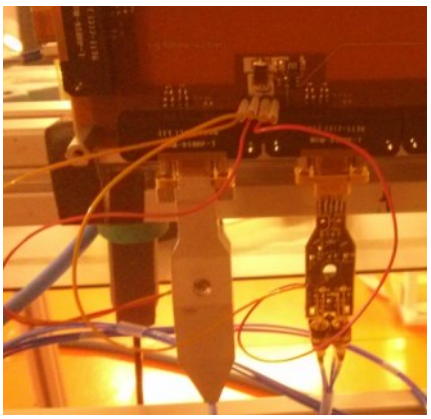

Fig. 10. TID test, transceivers with and without mechanical package

VII. SpW AOC Performances

As results of the GSTP activity was generated the first commercial version of an AOC for SpW copper cable replacement. Following table presents the current specifications of a copper cable and the measured values for the AOC for an example case of a 1 meter long cable.

TABLE II. Copper SpW cable Vs AOC

Especification	Copper cable	AOC
Mass	87 grams	<30 grams
Data rate	<400Mbps	<400Mbps by design Tested up to 380Mbps
Jitter/Skew	2000ps	190ps
Power consumption	NA	<700mW @ 200Mbps
Bending radius	>45mm	>25mm
Temperature range	-200 to +180ºC	-40 to +85ºC

The scalability of the AOC allows to improve the mass saving with longer cable lengths since the mass cost of a copper cable is around 80-100g/m but for an AOC is 4-10 g/m.

A real study of mass saving was performed comparing expected mass figures of harness and connectors.

TABLE III. Mass figures of SpW cable and AOC

		Connector mass (g)	Cable mass (g/m)
Copper cable	Min	9,5	83
	Max	23,5	100
AOC cable	Min	12	1,2 (x 4)
	Max	15	2,4 (x 4)

Using this table values the mass of a cable of 1m in copper will be around [102, 147]g whereas for optical fiber cable the mass is between [29, 39.6]g

This means a **mass saving of more than a 70% per cable.**

For longer cables the mass saving will be higher. For **10 m the mass saving could reach the 90%.**

VIII. Conclussions and Recomendations

The results from the GSTP activity were quite positive and DAS and Airbus think that the developed AOC could be a potential replacement for copper SpW cable for future applications for those scenarios where a mass reduction of current Spw harness is needed.

The major drawback of the AOC is the power consumption as well as the need of external powering the transceivers since there are no spare pins available at uD9 interface connector. Next ECSS issue will allow to use different connectors that changing minimally the mechanical package and the electrical design make possible to use a connector pin as power input to the transceiver.

At this moment DAS continues performing tests with debug equipment in order to complete all expected possibilities in future missions when connecting onboarded equipments. After these tests DAS expects to have a flight opportunity to check in real environment the SpW AOC (not only the technology as in previous flight opportunities) in order to raise TRL level of the solution.

Acknowledgment

DAS and Airbus Defence and Space want to thank the support received from ESA during GSTP execution, especially the help from the technical office, Nikos Karafolas, and also the rest of ESA departments involved in orbit validation, SpaceWire group and component selection.

References

[1] ECSS-Q-ST-60C Rev 2. Electrical, electronic and electromechanical (EEE) components.

[2] ECSS-Q-70-71A Rev.1. Data for selection of space materials and processes. Revision 1 of First issue. June 2004

[3] ECSS-E-ST-50-12C. SpaceWire - Links, nodes, routers and networks. Second issue. July 2008

[4] ECSS-Q-ST-70-08C. Manual soldering of high-reliability electrical connectios

[5] ECSS-Q-ST-70-38C. High-reliability soldering for surface-mount and mixed technology.

[6] ESA PSS-01-609. The Radiation Design Handbook.

ECSS-Q-ST-60-15C. Radiation hardness assurance- EEE components

SpaceWire to SpaceFibre Bridge

SpaceWire Components, Short Paper (Poster Presentation)

Sridhar Sundaram Natchinarkiniyan *(Author)*
Astrium GmbH
Ottobrunn, Germany
n_sri_ss@yahoo.co.in

Paul Rastetter *(Mentor)*
Astrium GmbH
Ottobrunn, Germany
paul.rastetter@astrium.eads.net

***Abstract*—This paper describes the architecture of a SpaceWire (SpW) to SpaceFibre (SpFi) bridge which merges multiple SpaceWire links from various communication nodes to the virtual channel buffers of a single high speed SpaceFibre link. The main goal is to make the bridge highly configurable and to allow each SpaceWire links to transmit data packets with full throughput and independent of its length without out blocking the network even if one is defected.**

***Index Terms*—SpaceWire, SpaceFibre, high speed serial link, bridge, flight compatible.**

I. INTRODUCTION

SpaceWire has been proven to be one of the most efficient, low latent, fault tolerant high speed serial communication interface by various missions of space communities around the world for years now. With the introduction of the SpaceFibre technology which is a spin-off from the existing SpaceWire protocol, higher data rates above 2Gbits/s and comparatively lesser cable weight is envisaged. With the escalation of highly complex network of devices for space environment, the on-board communication links should also be flexible to the devices to support both SpaceWire and SpaceFibre technologies with its maximum potential. This proposed bridge will expand the possibility of communication between instruments which support SpaceWire and instruments which support SpaceFibre without any bandwidth loss, minimized mass and expenditures for box to box communication.

II. ARCHITECTURE

The architecture of the bridge contains eight SpaceWire links which will be merged to eight virtual channel buffers of a single SpaceFibre link, of which six spacewire links will be used to transmit the data from various instruments. The remaining two SpaceWire links will be connected to some spacewire compatible micro-processer which will be used to control and configure the bridge through the Random Memory Access Protocol (RMAP). For our test the Multi-DSP Architecture (MDPA) processor from Airbus DS was used to control and configure the bridge. The SpW and SpFi IPs used in the bridge are from University of Dundee. The integration of the SpW and SpFi IPs is easy, because the SpFi protocol is designed to work with the SpW protocol, the structure of the data packets of both protocols are almost the same.

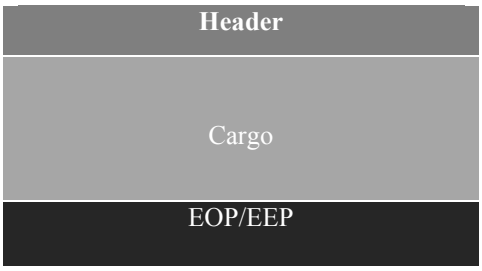

Figure (1). Packet Structure[1][5]

The data packet structure is given in the figure (1). The destination address will be the first data to be transferred from the header following the other necessary information about the packet. A single data frame of SpaceWire consists of 9-bits, MSB is the control character and the remaining 8-bits are data bits. In the SpaceFibre there is a separate 32-bit data bus and 4-bit control bus. A simple state machine can be used to convert the data and the control characters from SpaceWire to SpaceFibre format and vice versa. With this state machine one data word is transmitted in each clock cycle. The transmit clock frequency of the SpFi was set to 125 MHz which will transmit with 2.5Gbps and the SpW transmit clock frequency was set to 20MHz which will transmit with 200Mbps. The system clock frequencies for both IPs were set to 62.5 MHz. For different test cases the SpW transmit clock frequency can be variable.

The bridge was implemented with all the IPs like the RMAP controller, the Spacewire router, eight SpW links, one SpFi link and the necessary configuration and status registers for the whole design. Various FIFO interfaces were built around the IPs for proper flow control and sampling on both directions between the IPs. The difficulties of clock domain crossing within the design were solved by using dual port synchronous FIFO interfaces wherever required. The design was verified with Model Sim simulation. The test was conducted in various directions for data rate, fault tolerance and failure propagation. The design of the whole architecture is presented in figure (2).

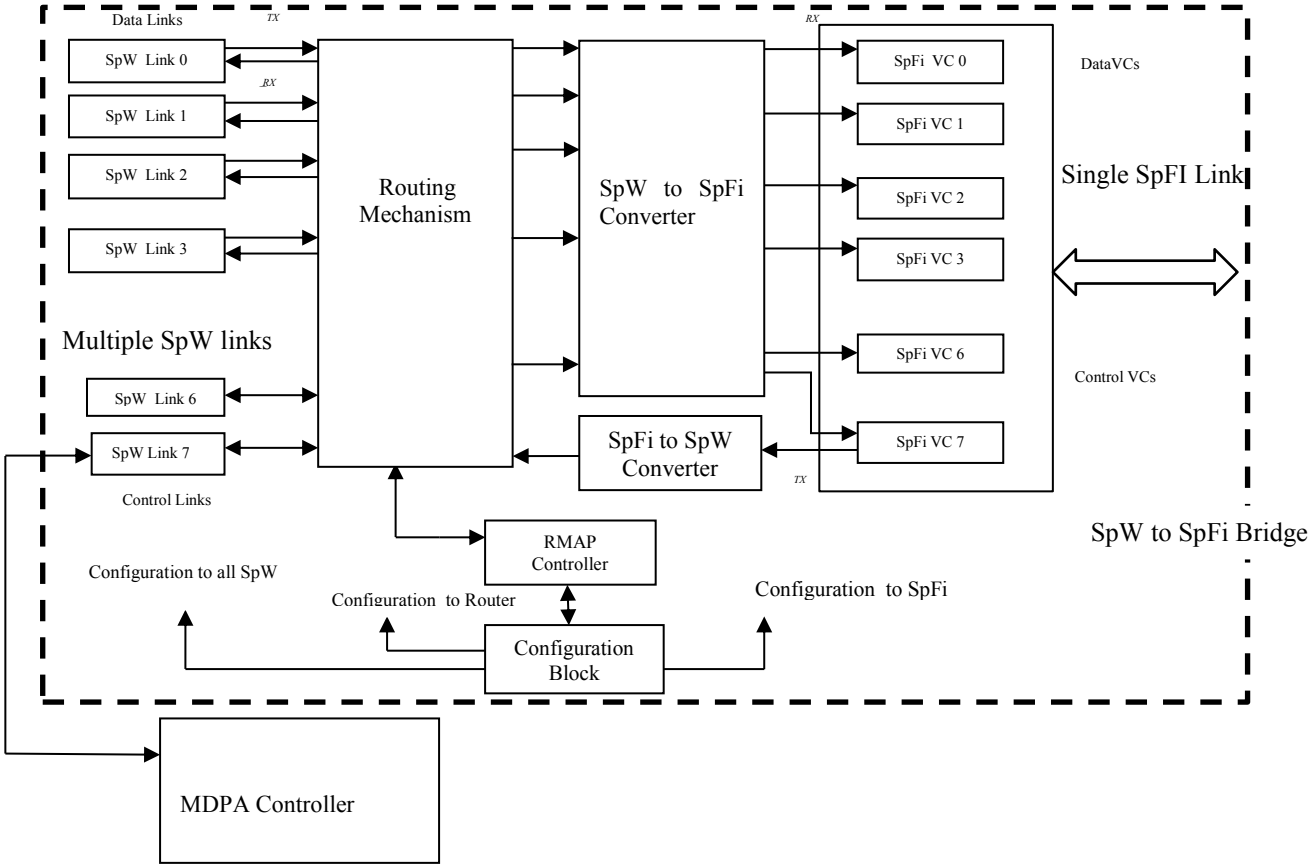

Figure (2). Architecture of the SpW to SpFi Bridge

III. ANALYSIS AND IMPLEMENTATION

The data rates of the bridge was tested for two possibilities, all SpW links transmitting in maximum data rate of 200Mbps and each links transmitting in varying data rates of maximum at 200Mbps to the minimum of 2Mbps. For the fault tolerance of the bridge various error cases were introduced in the design and verified for non-block operation of the bridge. Since the SpW router used in this bridge is working with non-blocking worm-hole routing technique, even if one link is defective and transmits for infinite time period, a timeout mechanism in the router can be implemented to recover the defective port from the software level. So each SpW link is completely independent from any other and would not affect the operation of the bridge.

The general operation of the bridge is configurable according to the implementation of the user in the design via various generics and configuration registers. For our implementation we needed some restrictions in the routing table for the communication of each sender and receiver. For example the SpW router used in the bridge contains seventeen ports, six data SpW links are considered as links coming from various instruments connected in the network and they are not allowed to communicate with each other. These data SpW have only access to transmit data to their respective virtual channel buffers of the SpFi or transmit housekeeping data via one of the two control SpW links. The RMAP controller can communicate only with one of the control SpW links. One of the eight virtual channels was used bi-directional and other virtual channels can only receive the data from their respective SpW links. The figure (3).shows the detailed routing permission table implemented in the bridge.

Components	Data SpW (To)	Ctrl SpW	RMAP	Data VC	Ctrl VC
Data SpW (From)	Not Allowed	Allowed	Not Allowed	Respective VCs(same number)	Not Allowed
Ctrl SpW	Allowed	Not Allowed	Allowed	Not Allowed	Respective VCs(same number)
RMAP	Not Allowed	Allowed	Not Allowed	Not Allowed	Not Allowed
Data VC	Not Allowed	Not Allowed	Not Allowed	Not Allowed	Not Allowed
Ctrl VC	Not Allowed	Only last one	Not Allowed	Not Allowed	Not Allowed

Figure (3). Implemented Routing Permission Table for the bridge

IV. SYNTHESIS AND HARDWARE EMULATION

The hardware implementation of the bridge was proved in Chip-It hardware emulation systems from Synopsys. For testing the design in the Chip-It systems a synthesizable test bench was designed with packet generators, packet checkers and a loopback mechanism. The random packet generators are capable of generating SpW and SpFi data with a CRC (cyclic redundancy check) attached to it before the end of packet marker. The packet checkers can receive the incoming data to check for the CRC and generate the number of packets received and the number of corrupted data in transmission. The necessary information about the packet is included in the header of the packet.

The Chip-it hardware emulation system uses UMR-Bus protocol from Synopsys to communicate with the design under test. The design under test (DUT) was connected to the APB bus as one of its slaves through a configuration register block which acts as the medium to transmit data from the external world to the bridge and vice versa. The packet generators and checkers were used to emulate the actual instruments transmitting SpW packets. A separate SpW FIFO interfaced was also attached with the APB bus for the transmission and reception of the RMAP packets from the external world through the MDPA for the configuration of the bridge. The slaves connected to the APB bus can be accessed by the UMR bus through an APB to UMR Bridge and the data to be transmitted can be given through TCL commands from a PC. The transmit and receive interface above the DUT contains SpW and SpFi IPs or dual port FIFO interfaces to pass the data from the generators and checkers to the DUT. For the first set of test in the Chip-it the FIFO interfaces were used inside the DUT and in the test bench to transmit and receive data in same data rate like the SpW links.

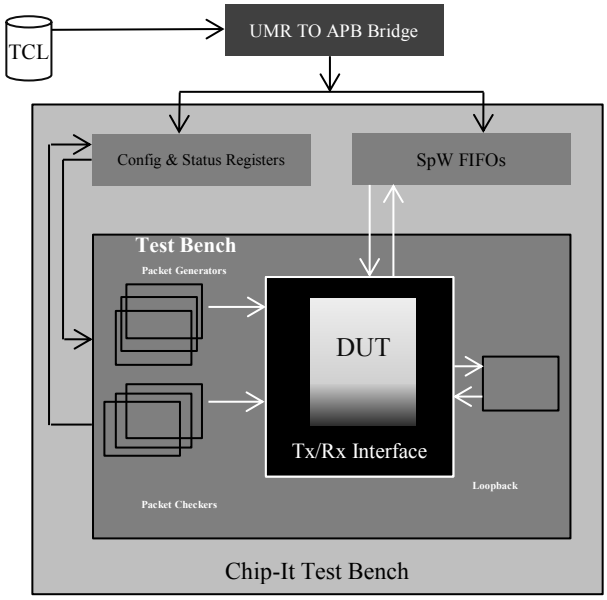

Figure (5). Hierarchy of Chip-it Test Bench

The input data for the packet generators will be written to the registers in the Chip-it test bench and the outputs from the checkers like the number of received, defective packets and various other data about the transmission can also be written in the register blocks. By accessing those register blocks with TCL commands we can know the operation of the bridge. The reason for using the FIFO interfaces instead of the SpW links is because of reduced availability of the clock buffers in the hardware. After the successful implementation of the bridge in the hardware and tested for normal operation it transmitted without any failure. Then the behavior of the bridge was observed with some failure test cases. After all these testing the bridge has been proven to be robust, fault tolerant and non-blocking even with maximum throughput. The second set of test was implemented in the Chip-it systems with three SpW links for data transfer and one for controlling the bridge through the MDPA controller. The behavior of the bridge was same like the first test but with little latency. The hierarchy of the Chip-it test bench is explained in figure (5).

The Chip-it emulation hardware contains two Xilinx Virtex-5 (XC5VLX110) FPGAs. The bridge design uses very less resources when synthesized for Virtex-5 FPGA. The area summary for Virtex-5 is in the figure (6).

Resources Used	Quantity
I/O Ports	579
DSP48s	1(64)
Non I/O Register bits	8314(12%)
Block RAMS	15(128)
LUTS	12076(17%)

Figure (6). Resource Utilization of the bridge

V. FUTURE ADD-ONS

The next step of addition in the bridge would be the timecode interface. It has been planned to implement a separate network from this bridge to transmit the timecodes from the SpW to the broadcast interface of the SpFi link.

VI. CONCLUSION

The results obtained from the implementation of the bridge shows that it is highly configurable, robust and fault tolerant which is also very easy to adapt in any network with devices supporting the SpW and SpFi standards. This architecture will comparatively reduce the total mass and costs. With the future add-ons and optimization to this current bridge, it will be compliant with most of the radiation hardened FPGA technologies.

REFERENCES

1) ECSS, "Space Engineering: SpaceWire- Links, nodes, routers and networks", EECS-E50-12C, 31 July 2008.

2) SpaceWire Codec IP, User Manual, Uod_Link_User 2.3,

12 March 2008, STAR-Dundee Ltd, www.star-dundee.com

3) SpaceWire CODEC IP, VHDL Functional Description, Uod_Link 2.3, 12 March 2008, STAR-Dundee Ltd, www.star-dundee.com.

4) ECSS, "Space Engineering: SpaceWire- Remote Memory Access Protocol", ECSS-E-ST-50-52C, 5 February 2010.

5) SpaceFibre Specification, Draft F3, 10 September 2013, Space Technology Center, University of Dundee.

6) SpaceFibre VHDL IP Core, User Manual, Doc-ref 1.4, 17th January 2013, STAR- Dundee Ltd, www.star-dundee.com.

7) CHIP-it Iridium/Copper Edition, Handbook, D-2009.12, Synopsys, Inc. www.synopsys.com.

Protocols for Deterministic Packets Delivery in SpaceWire Networks

Networks and protocols, Short Paper

Dmitry Raszhivin, Yuriy Sheynin
State University of Aerospace Instrumentation
St. Petersburg, Russia
dmitry.raszhivin@guap.ru, sheynin@aanet.ru

***Abstract*— Tasks of deterministic packet delivery in conventional SpaceWire Networks are considered. The published draft of the SpaceWire-D is considered in the general context of time division (TDMA) multiplexing in comparison with FlexRay, TTP, TTEhernet, ZigBee, etc. Advances for efficient TDMA in conventional SpaceWire networks – static and dynamic time-slot segments, variable epoch duration, multiple transport protocols support, "trusted" end-nodes are considered.**

***Index Terms*— SpaceWire, Networking, Spacecraft Electronics.**

I. Introduction

Time Division Multiplexing is a well-known and widely used in network technologies channel access method. Time division guarantees for nodes of a network predictable transmission characteristics with deterministic latency. Such industrial networks, as FlexRay, TTCAN, TTEhernet, use TDMA principles to obtain guaranteed transmission characteristics. St. Petersburg State University of Aerospace Instrumentation investigates time division multiplexing support for SpaceWire transport protocols developments in correspondence with industry requirements.

II. Time division multiplexing in communications

Time multiplexing is actively used in 2G and 3G mobile networks, as well as in some wireless personal networks, such as Bluetooth, ZigBee, Ubiquiti. However, SpaceWire developers are primarily interested in the experience of using time division multiplexing applying to wired networks.

A. TTCAN

The time-triggered CAN protocol [1] is a higher layer protocol on top of the CAN data link layer. TTCAN provides mechanisms to schedule CAN messages in a time-triggered way as well as in an event-triggered way. It allows using CAN-based networks for closed-loop control. Also the real-time performance in CAN-based in-vehicle networks increases with the use of TTCAN.

The time-triggered control and thus synchronization of the involved control units in a network are done via a reference message. All participants of the TTCAN network identify the reference message by its identifier. As soon as the first bit of the frame (Start of Frame: SOF) is recognized, the local time unit is synchronized. The accuracy of the local time units depends only on the physical signal propagation of the bus line and is thus is neglectable. Individual TTCAN participants are configured to know when to send their frames after having received the reference frame.
The time between two reference frames is called the basic cycle. Basic cycles are not always identical in order to be able to transmit messages at different periodic frequencies. The system matrix comprises several basic cycles and is repeated indefinitely until the vehicle network is turned off.

B. FlexRay

FlexRay [2] is a fast, deterministic and fault-tolerant bus system for automotive use, based on the experience of Daimler-Chrysler with the development of prototype applications and the developed by BMW byteflight communication system.

FlexRay works according to the TDMA principles. However, the fixed allocation of the bus bandwidth to the FlexRay components or messages by means of fixed time slots has the disadvantage that the bandwidth is not fully exploited. For this reason FlexRay subdivides the cycle into a static and a dynamic segment. The fixed time slots are situated in the static segment at the beginning of a bus cycle. In the dynamic segment the time slots are assigned dynamically.

In order to implement synchronous functions and optimize the bandwidth by means of small distances between two FlexRay messages, distributed components in the communication network require a common time base (global time). For clock synchronization, specific FlexRay messages tagged as synchronization messages are transmitted in the static segment of the cycle. With the aid of a special algorithm, the local clock-time of a component is corrected in such a way that all local clocks run synchronously to a global clock.

C. TTP/C

The TTP/C [3] frame consists of one byte header, up to 236 bytes of the payload and 3-byte CRC field. TTP/C implements the time-division multiplexing approach based on MEDL (Message Descriptor List), which shall be loaded into

every node. The MEDL contains predefined static data to control when a message shall be sent on or received from the communication channels.

Each MELD string consists of the following main fields: the global time when the message shall be sent or received; the memory location of the message intended to be sent or received; the attributes field that includes the message type (input or output), the message length, etc.

To determine whether a node is operating correctly the membership service is used. Each bit in the membership field corresponds to a particular cluster node. When a node is allocated for data transmission at a particular time unit, all other nodes in the cluster analyze the input data so as to determine whether this node operates correctly.

D. TTEhternet

TTEthernet [6] is implemented on the basis of Ethernet. It provides time-triggered communications and global clock synchronization as well as a fault-tolerant operation mode. TTEthernet offers three types of traffic classes: Time-Triggered, Rate-Constrained and Best-Effort.

TTEthernet implements the TT class of QoS by the combination of resource reservation in space and time-division multiplexing. Each TTEthernet device in the network shall send TT frames only at predefined points of time to avoid collisions. On the other hand, frames, which are transmitted over different paths, can be sent to the network at the same time. To support this scheme TTEthernet implements clock synchronization mechanism.

In order to prevent error propagation from failed components the fault-tolerant TTEthernet network configuration deploys two independent channels for each connection. Safety-critical TTEthernet controllers shall be able to transmit and receive messages using two communication channels simultaneously.

In order to detect a failure of nodes within a cluster, TTEthernet provides membership service similar to TTP/C.

III. SUGGESTIONS

To develop a Transport protocol conforming space industry demands, [5], we propose flexible epoch with static and dynamic segments, guarding port operation and redundant time master operation.

A. Epoch ogranization

Each new time code indicates the beginning of a new time slot. Number of time slots in the epoch can vary from 2 to 64. The question is – how the node should determine the slot number basing on the time-code value.

Let a network member know a total number of time-slots in the epoch. It increments a slot's counter on receiving of the next valid time-code. This counter is reset to zero when the maximum value is reached. The disadvantage of this method is loss of transparency that is pawned in SpaceWire-D draft: the time-slot number is equal to the received time-code value.

Router synchronization problem arises if two routers were turned on at different time moments, as it is often done in real equipment. When the second router receives the first (his) time code, it would be treated as the start point for the first time-slot. At the same time this time-code would be subsequent for the first router that started before the second one, and the routers come out of synchronization. The problem is shown in Fig. 1. : the router K1 is the time-master, it distributes time-codes, the epoch consists of 4 time-slots. The second router K2 turns on two slots later and receives a time-code with value “3”. However, it can’t unambiguously determine a place in the epoch of the current time-slot using the incoming time-code value, because time-code value is not equal to time-slot number.

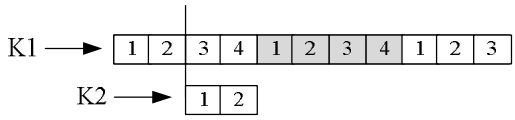

Fig. 1. Time-slots counter

Restriction on the multiplicity is another approach to epoch organization. Let’s limit the value of time-slots in the epoch by the values that are the power of two: 2, 4, 8, 16, 32, and 64. All routers should know the value of power *n,* for two time-slots n=1, for 4 slots n=2, for 8 slots n=3 etc. All network participants can identically determine time-slot number while receiving new time-code if value *n* is pre-defined for them.

B. Port Guardian

A lot of network technologies, described above, suppose port guardian mechanism in order to protect the network from faulty nodes that try to transmit data at appropriate time-slots. Often such a "watch dog" is implemented as a separate device or chip in order to increase fault tolerance. Port guardian guarantees that the node would not transmit data during wrong time-slots and eliminates «babbling idiot» problem. Port guardian mechanism is supposed to be included to SpaceWire routers or nodes in order to improve network fault-tolerance of a deterministic SpaceWire network.

The Fig. 2. shows a SpaceWire network with time division multiplexing support. Network routers, marked as «Net guard», store scheduling table and permit data transfer for nodes only at proper time moments. The central part of the router is the standard SpaceWire router.

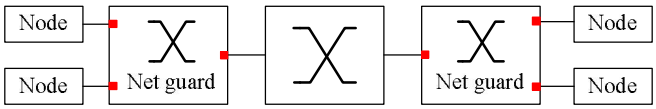

Fig. 2. Network with network guardians

The Fig. 3. shows «Net guard» router's port. This router has the ability to block transmission that violates the predetermined during configuration time scheduling table.

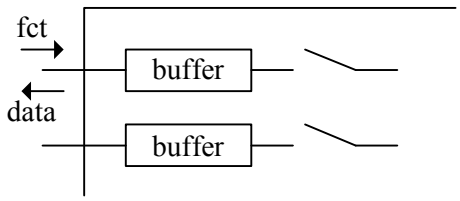

Fig. 3. Port guardian

C. Shadow master

An important task for the time division multiplexing protocol is to provide a fault-tolerance mechanism for the time markers distribution [4]. Loss of a single time-code with «master-slave» time synchronization leads to two time-slots loss; full time-master failure leads to the absolute network closedown.

Shadow master is one of possible solutions to increase time distribution fault-tolerance. This backup master can send time-codes in addition to the primary time master. It should be noted that this is a violation of the normative part of SpaceWire standard, that says that only one channel interface in the whole network should actuate an active tick signal. The Shadow master continuously checks the status of the primary time master by controlling the validity of incoming time code (Fig. 4.). If the shadow master does not receive valid time code within a certain predefined time, it would start time codes distribution itself.

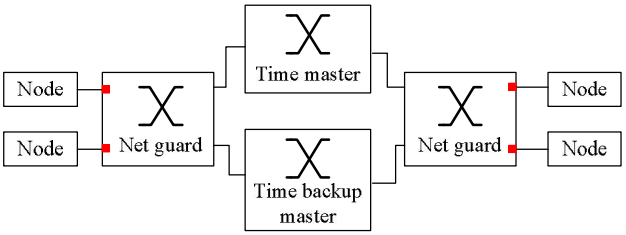

Fig. 4. Time backup master

D. Dynamic and static segments

It is necessary to integrate scheduled traffic and event-triggered traffic to effectively utilize psychical resources of the network. An epoch is divided into two parts for it; those parts are used for data transmission of scheduled or event-triggered traffic (Fig. 5.).

Scheduled data transmission goes during static segment, flow control manages epoch division into slots. During dynamic segment all TDMA mechanisms are switched off, network runs at “classical” SpaceWire mode. The last slot in the epoch is designed to clean the routers' buffers of data, which has not been sent, to transmit an EEP or EOP symbol and to prepare the conversion to the static segment.

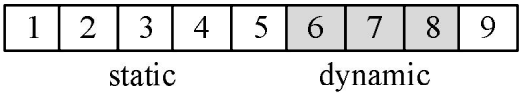

Fig. 5. Static and dynamic segments

IV. CONCLUSION

The paper gives an overview of several network technologies, that use time division multiply access for deterministic data transmission. Several mechanisms are suggested for organization of deterministic packet delivery protocol in SpaceWire networks. It could be used for scheduling in new Transport protocol developments based the requirements of space industry and in further developments of the SpaceWire-D or its successor protocols. .

REFERENCES

[1] Fuehrer, T., Mueller, B., Hartwich, F., and Hugel, R., "Time Triggered Communication on CAN (Time Triggered CAN-TTCAN)," SAE Technical Paper 2001-01-0073, 2001

[2] A. Hanzlik, “A Case Study of Clock Synchronization in FlexRay”, Research Report 31/2006 Technische Universitat Wien, Institut fur Technische Informatik, 2006

[3] H. Kopetz, “Real-Time Systems. Design Principles for Distributed Embedded Applications”, Kluwer Academic Publishers, Boston, 1997.

[4] W. Steiner, R. Maier, D. Jameux, A. Ademaj “Time-triggered services for SpaceWire”, Proceedings of the 2nd International SpaceWire Conference, 2008

[5] Lavrovskaya I., Olenev V., Korobkov I., Dymov D., “Analysis of the Transport Protocol Requirements for the SpaceWire On-board Networks of Spacecrafts”, Proceedings of the 15th Conference of Open Innovations Association FRUCT, Saint-Petersburg, Russia, 21-25 April 2014

[6] A. Ademaj, H. Kopetz, P. Grillinger, et al., “Fault-Tolerant Time-Triggered Ethernet Configuration with Star Topology”, available on-line

HANDS：A Heterogeneous Aerospace Network Architecture For Disaggregated Satellites based on SpaceWire

SpaceWire networks and protocols, Short Paper

Wei Han, Baosheng Wang, Baokang Zhao*, Jing Tao, Zhu Tang
College of Computer,
National University of Defense Technology
Changsha, Hunan, P.R. CHINA
{weihan, bswang }@nudt.edu.cn

***Abstract*—Fractionated spacecraft such as system F6, has the potential to significantly enhance the adaptability and survivability of space capabilities, while also shortening the development time for complex space systems. It has become an important trend in the development of small sized satellite.**

SpaceWire has been widely adopted in satellite for its high throughput in communication and simplicity in design, and it is believed that SpaceWire will be deployed as a common standard in small satellites. However, it lacks the capability to communicate in the fractionated spacecraft scenario.

In this paper, we propose HANDS, a Heterogeneous Aerospace Network architectures for Disaggregated Satellite based on SpaceWire. HANDS consists of three parts. First, satellites communicate with their fixed identifier and routing among them could be realized by identifier. The introduction of identifier assures the zero-loss packet during handover process. Second, Egress Router(ER) on satellites maintains reachable information of onboard equipment and shields the difference of the equipment's location. The introduction of ER helps to keep the SpaceWire communication standard in single satellite and strengthens the scalability of the network. Third, the equipment's address is coded globally, which benefits the networking between equipment. We also perform careful analysis and discussion on characters of HANDS and show the benefits of this architecture.

***Index Terms*—SpaceWire, Fractionated spacecraft, network architecture, identifier.**

I. INTRODUCTION

System F6 program[1], lead by DARPA, is proposed to develop and demonstrate the enabling technologies for fractionated spacecraft architectures. As is shown in Fig 1, the fractionated spacecraft are a set of disaggregated satellites whose function inherit from a single large satellite. Function of traditional satellite is divided into several independent parts and each of them is realized on certain fractionated satellite. These spacecrafts are wirelessly-interconnected and capable of seamlessly sharing a variety of resources such as computation, storage and so on. Such an architecture has the potential to significantly enhance the adaptability and survivability of satellite in aerospace, while also shortening the development time for complex space systems. More security policies and fault tolerance scheme could be realized on satellites and the reliability could be improved at the same time.

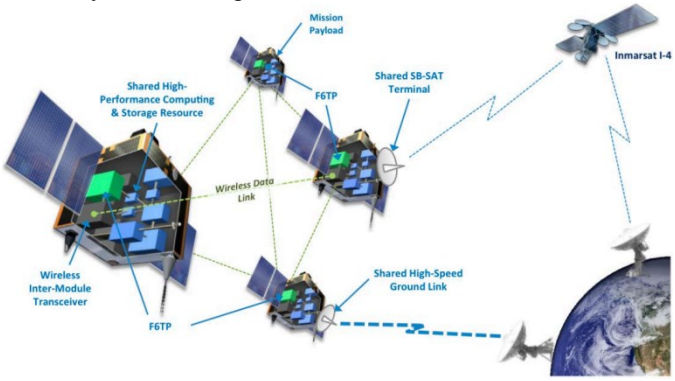

Fig. 1. Notional depiction of the F6 on-orbit demonstration [1]

Taking into account that SpaceWire lacks the capability to communicate in fractionated spacecraft, we propose HANDS, a Heterogeneous Aerospace Network architectures for Disaggregated Satellite based on SpaceWire which provide a solution to integrate wireless network into the wired one. In this architecture, satellites communicate with fixed identifier and routing among them could be based on identifier. The introduction of identifier assures the zero-loss packet during handover process. Egress Routers(ER) on satellites maintain reachable information of onboard equipment and shields the difference of their location. The equipment's address is globally coded and can promote fast routing and switching on satellite network.

II. APPLICATION SCENARIO

The proposed application scenario is shown in fig 2. Traditional satellite is disaggregated into several wirelessly

interconnected modules (S1,S2,S3,S4). Its function is divided into several independent parts at the same time. Satellite S1 provides computing and storage resources. S2 collects environmental information through sensors. S3 is used to provide high speed inter-satellite link (ISL). S4 supports high speed ground-satellite link (GSL) between fractionated satellites and gateway on the earth. Communication inside each spacecraft adopts SpaceWire standard but not between satellites. All fractionated satellites share information and resources through wireless communication which plays an important role in providing high speed information exchanging between fractionated satellites.

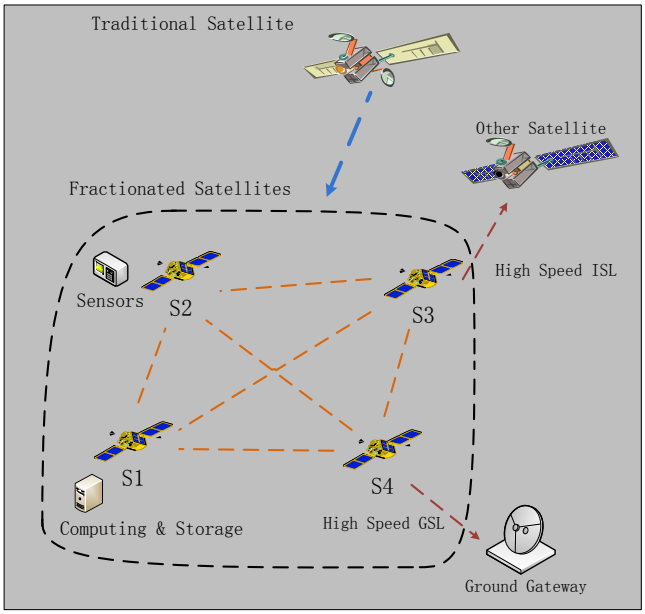

Fig. 2. Scenario of fractionated spacecraft

III. System Architecture

A. HANDS model

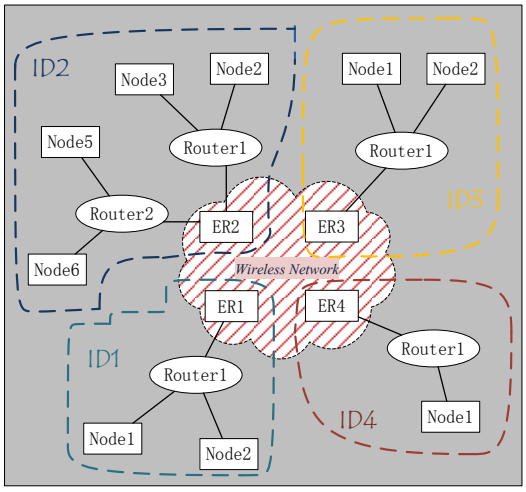

Fig. 3. Model of HANDS

There are mainly three different roles in HANDS. As is shown in Fig 3. Node is the data source on satellite and has an independent address. Router is responsible for packets routing and switching on satellite. Egres Router (ER) works as an gateway between equipment on satellite it belongs to and that on other fractionated satellites. The fractionated spacecraft forms an WLAN(Wireless Local Area Network) in space and communicate through ER. Communication between satellites adopts spacewire standard which is not available in this scenario. ER is responsible for processing and switching packets in wireless network so that heterogeneous network could cooperate to form an unified architecture.

B. Characters of system

1) *Satellites communicate with fixed identifier:* In HANDS, each satellite or teminal has a fixed identifier which is used for to communicate with other satellites or terminals. Address of satellites may change while identifier remains the same and is globally unique. Identifier is also used for rouing between satellites when address is invalid during the handover process between fractionated satellites and other terminals (e.g. other satellite, ground gateway, user terminal). The identifier makes sure that the packet be sent to destination instead of being dropped. When packet is transmitted to ER and will be sent to other terminals which is not within fractionated satellites, identifier will be encapulated in the packet header.

2) *ER maintains connection:* ER is mainly responsible for packet encapsulation and decapsulation when packet passes through. As the gateway of satellite, ER not only modifies message but also shields the difference of equipment's location by NAT(Network Address Translation) which is used to translate node's private address into the public(globally reachable) one and vice versa. It maintains an mapping table which include addresses of node on fractionated satellites and related addresses which is used to communicate with other satllites or teminal. In this way, communication between ERs is transparent to nodes and routers and nodes on different fractionated satellites would communicate as if they were in the same satellite.

3) *Coding nodes' address globally in fractionated satellites:* The nodes' addresses are coded globally so that the address can be used directly when communicating with nodes within fractionated satellites. At the same time, ER could decide the egress port of the packet by its destinated address. Therefore, communication between two nodes has no difference with that on tranditional statellite.

IV. System Scenario and Discussion

A. Networking with identifier

Identifier is used to uniquely identify certain sallite or terminal. Satellites in different location access gateway on the earth with their identifier and get allocated addresses. Then communication packets will be routed based on address while the identifier are still encapsulated in packets which can be used for routing when addresses are invalid. As shown in fig 4, fractionated satellites S1 and S2 has their own identifer ID1, ID2. When they access the ground gateway, they will get

address IP1 and IP2 repectively. With the movement of satellites, link between S2 and gteway will be interrupted while S1 will access the gateway. In this scenario, when there is a packet with destination address of IP2(S2) is transmitted to gateway, it will be dropped because the link between S2 and gateway has already been disconnected. When identifer is encapsulated in the packet, gateway will check the identifer information and know that the destionation is S2. Then packet will be forwarded to S1 according to ceratin policy and arrive at S2 correctly.

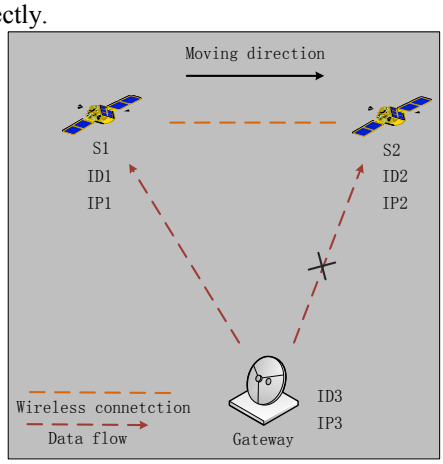

Fig. 4. Example of data flow during handover process

The introduction of identifer helps to avoid loss of packet during handover process and improves the utility of the network. Networking with identifer makes mobility management of satellites straightforward and specific and is more suitable to the scenario when network node keeps moving. Simply using IP address in network cannot meet the demand of constantly dynamic toplogy of satellites. Nodes on satellites need some fixed identifier to indicate themselves during the movement so that packets can be routed to destination when address is invalid.

B. Egress Rrouter

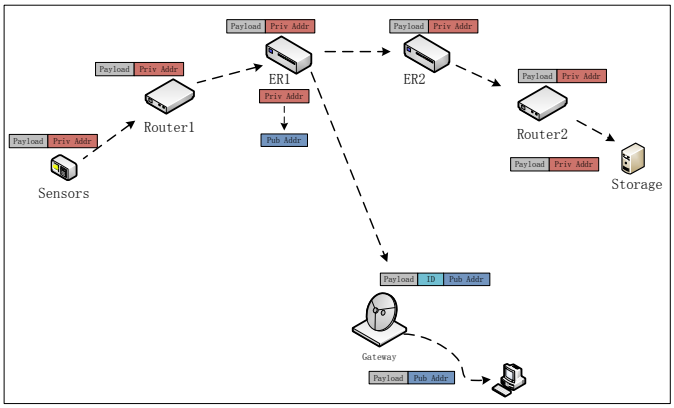

Fig. 5. Packet process in HANDS

As shown in fig 5, when communication is going on between fractionated satellites, ER can decide whether the packet is sent to the fractionated satellite based on the destination address and will send the packet through ER without further operation if it is. Gateway ER of destination receives the packet and forward it to the destinated node. When the destination is outside the fractionated satellites(gateway as an example in fig 5) and packet is passed through ER, ER will encapsulate identifier information in the packet and translate the private address into a global reachable address. And when there is packet received from wireless network, ER will decapsulate the identifer information and mapping the destination address into the private one. Then the packet will be sent to the destinated node.

Packet flow will be divided into two classes based on the private address range. Processing of packets is avoided in communication within fractionated satellites which helps to realize efficient routing and exchanging in satellites. This process is completed in ERs and is transparent to routers and nodes on satellites which maintains the compatibility to current standard. The modularity of the satllites' function does help in the development of satellites and ER plays a crucial role.

C. Globally addressing

To code the node address globally in the fractionated satellites is advantageous to fast routing and switching within satellites. The gateway on satellites could decide the destinated node's location from the destinated address and can forward the packet to ceratin egress port directly. Operation that need to be performed to packets during routing process is reduced as much as possible. Independent addressing surely benefit a lot in scalability of fractionated satellites but it involves much more operations in routing between them.

V. Conclusion

In this paper, we propose an heterogeneous architecture HANDS in fractionated satellites. The main character of the architecture is that it introduces Egress Router as the gateway of the fractionated satellites. ER helps to realize networking with identifier in satellites and transparent routing within the fractionated satellites. The introduction of identifier is a innovative way to satellite network and needs further research. This architecture made least changes to current communication standard on satellite and provide an solution to communication in F6 scenario.

Acknowledgment

The work described in this paper is supported by the project of National Science Foundation of China under grant No.61202488, No.61103182, No.61379147 the program for Changjiang Scholars and Innovative Research Team in University (No.IRT1012).

References

[1] Ong E, Brown O, Losinski M J. System F6: Progress to Date[J]. 2012.

[2] Pan J, Jain R, Paul S, et al. MILSA: A new evolutionary architecture for scalability, mobility, and multihoming in the future internet[J]. Selected Areas in Communications, IEEE Journal on, 2010, 28(8): 1344-1362.

[3] Shahriar A Z M, Atiquzzaman M, Rahman S. Mobility management protocols for next-generation all-IP satellite

networks[J]. Wireless Communications, IEEE, 2008, 15(2): 46-54.

[4] ECSS-E-50-12-C. SpaceWire Engineering: SpaceWire-Links, node, routers and networks ESA-ESTEC. November 2008.

[5] ECSS-E-ST-50-51C. SpaceWire protocol identification. ESA-ESTEC ,2010.

[6] SpaceNet-SpaceWire-RT Initial Protocol Definition. Space Technology Centre School of Computing University of Dundee, DD1 4HN Scotland, UK. October 2008.

[7] Qiao Liyan, Chen Libin, Peng Xiyuan, “Design of spacewirePCI correspondence card based on IP core”, JOURNAL OF ELECTRONIC MEASUREMENT AND INSTRUMENT, Vol.24 No.10, pp.918-923, 2010.

Towards Software Defined SpaceWire Networks

SpaceWire Networks and Protocols, Short Paper

Jinzhen Bao, Baokang Zhao, Zhenghu Gong, Chunqing Wu, Wanrong Yu, Zhenqian Feng
School of Computer
National University of Defense Technology
Changsha, Hunan, CHINA
{Jinzhenbao, bkzhao, gong, wuchunqing, wlyu, zqfeng}@nudt.edu.cn

***Abstract*—In the recent few years, Software Defined Network (SDN) brings a revolution to network technology. Comparing with the traditional techniques, SDN has several distinguished features, including fine-grained flows management, global view of the network and centralized control, etc. Since SpaceWire is becoming a standard for high-speed links and networks for use onboard spacecraft, we argue that adopting the idea of SDN into SpaceWire networks will bring several advantages, including open network topology, fine-grained control and QoS, etc.**

In this paper, we propose a new software defined SpaceWire network architecture: SDSpW. In SDSpW, the core of SpaceWire network contains three roles: controller, router and end-nodes. The controller plays a center role in managing the routing, switch within the whole network, while SDSpW router adopts the control-forward separation philosophy. In the forwarding plane, SDSpW integrates a fine-grained flow technology by importing a multi-field flow table. With the introduction of flow table, SDSpW make it easy to fine grained flow control and ensures the end-to-end Quality of Service (QoS). In the control plane, SDSpW controller and router run openSpW, a customized protocol on top of SpaceWire - RMAP.

We also conduct several experiments in the environment of mininet to evaluate the performance of SDSpW. Experimental results show that the controller can monitor the state of whole network in real time, which makes the maintenance and management more easily. Moreover, the end-to-end QoS is guaranteed.

***Index Terms*—SpaceWire, Software defined network, SDSpW, Quality of Service.**

I. Introduction

In the recent years, SDN technology has brought a revolution to network technology. The purpose of SDN is to make the network dynamic, manageable, adaptable to suit for today's network applications. Different from traditional network architecture, SDN architecture decouples the control plane from data plane. The basic SDN architecture is shown in Fig. 1, which consists of data plane, control plane and application plane. The data plane comprises network elements, which expose their capabilities toward the control plane via southbound interface. And it mainly focuses on packet forwarding. The application plane communicates with control plane via northbound interfaces in charge of network management. The control plane is a logically centralized entity which mainly focuses on (i) translating the applications' requirements to data plane and (ii) providing the real-time status of network to application plane. [1]

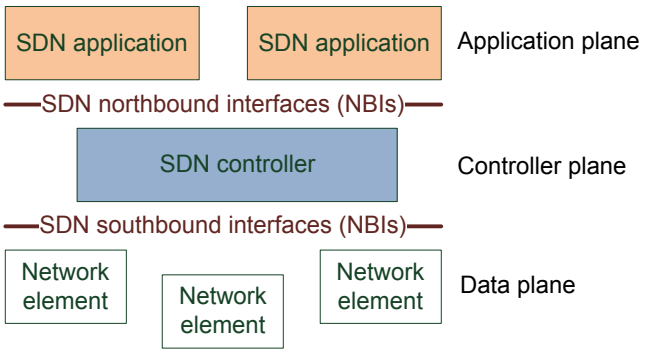

Fig. 1. Basic SDN architecture

Comparing with the traditional network techniques, SDN has several distinguished features, including fine-grained flows management, global view of the network and centralized control, etc.

And in the area of space, SpaceWire is becoming a standard for high-speed links and networks for use onboard spacecraft. But there still exists some problems in the SpaceWire network. 1) With the development of On-board system, the scale of SpaceWire network is increasing. 2) The congestion of network is still a problem caused by wormhole routing mechanism. 3) The tasks of spacecraft sometimes change, so the priority of nodes should change with the tasks to guarantee the QoS of high priority tasks. The SpaceWire network lacks feasible mechanism to support the changing.

In this context, we would adopt the idea of SDN into SpaceWire networks. And we propose a new software defined SpaceWire network architecture: SDSpW. There will bring several advantages, including open network topology, fine-grained control and end-to-end QoS guarantee, etc.

The rest of the paper proceeds as follows. Section 2 describes the architecture of SDSpW. Section 3 presents the

design and implementation of SDSpW. In section 4, we evaluate the performance of SDSpW. And section 5 concludes the paper.

II. AN OVERVIEW OF SDSPW

The overview of our software defined SpaceWire network is shown in Fig. 2.

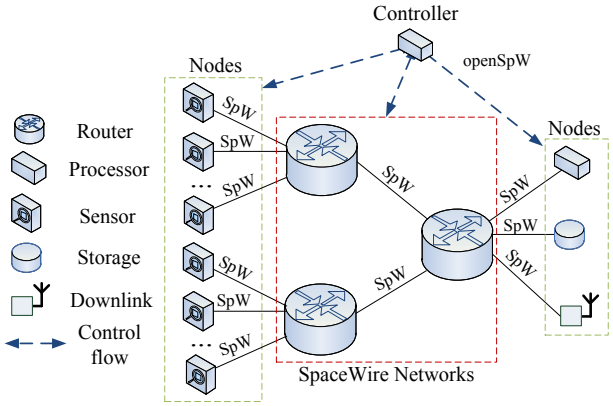

Fig. 2. Architecture of SDSpW

The SDSpW architecture contains three roles: controller, router and end-nodes. The controller plays a center role in managing the routing, switch within the whole network, while the router and nodes take the role of data plane to forward packets. In the control plane, SDSpW controller and router run openSpW, a customized protocol on top of SpaceWire - RMAP. And in the forwarding plane, SDSpW integrates a fine-grained flow technology by importing a multi-field flow table. With the introduction of flow table, SDSpW make it easy to fine grained flow control and ensures the end-to-end Quality of Service (QoS).

III. THE DESIGN AND IMPLEMENTATION OF SDSPW

In this section, we will give a detail description of the design and implementation of SDSpW.

A. Data Plane

The data plane consists of SpaceWire router and nodes. The packets in SpaceWire networks can be routed based on either path address (range 0 to 32) or logical address (range 32 to 254). Considering the standard of SpaceWire network, we would make litte modification to make the router and nodes much more feasible and fine-grained control. We will consider both the situation of path address logical addresss.

1) Path Addresses (PA)

The routing mechanism using path addresses is also called wormhole routing. The SpaceWire networks use wormhole routing to deliver packets as fast as possible with low-level flow control. Wormhole routing has the advantages of minimizing the amount of buffers and transmission latency. However, the low-level flow control leads to link congestion when a long packet occupying the router.

In SDSpW, the controller has an abstract view of the network status. It can dynamically configure the node's path addresses to choose a better route. And the router in data plane has the ability to support multi priority flows distinguished by the ports, which also can be configured by the controller. The combination of nodes and router will meet the requirements of QoS.

The processing of a SpaceWire packets by nodes and routers following the above functions is shown in Fig. 3. In this redundant SpaceWire network, the packets are transferred from node2 to node4 with the path addresses <4, 2, 1>. When the controller monitors that router1 is congested, then it can configure the path addresses of node2 to <5, 2, 1>. So the packets can be transferred in real-time with multi-path wormhole routing, which can ease the congestion phenomenon and increase the bandwidth utilization.

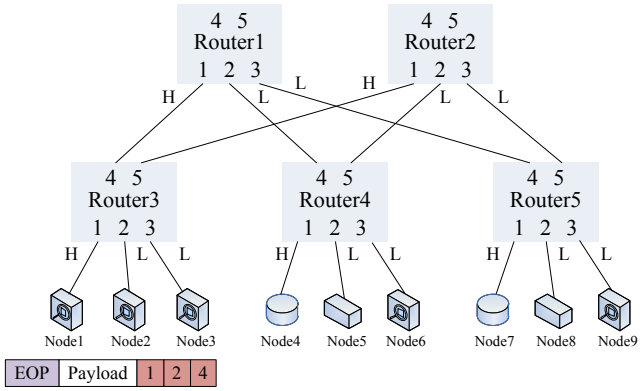

Fig. 3. Multi-path wormhole routing

On the other hand, the modified router forwards packets based on priority, as shown in Fig. 4. In Fig. 4(a), the packets from low-priority port 2 are occupying the output port 3 while the packets from high-priority port 1 are arriving. Different from traditional priority mechanism, in which the high-priority packets have to wait for an existing low-priority packet to complete, the high-priority packets are immediately transferred out, as shown in Fig. 4(b). This mechanism guarantees the high priority traffic transferred in real-time. Moreover, the priority of ports can be dynamically configured by the controller based on tasks.

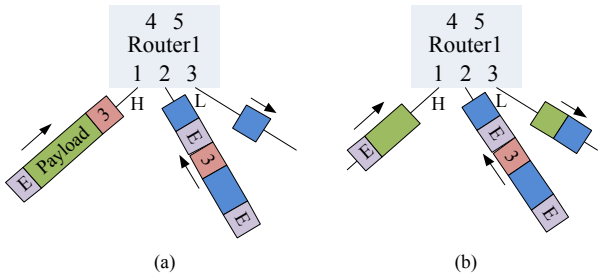

Fig. 4. Priority scheduling

2) Logical Addresses (LA)

The routing mechanism using logical addresses is similar to the algorithm used in ground routers. The complexity of path addressing is mainly handled by the source nodes and the routers are relatively simple. To support logical routing, each

router contains a routing table to forward packets which simplifies the function of nodes. For a larger network, the bandwidth utilization of wormhole routing is low since the path addresses increases. On the contrary, the logical addresses mechanism is suitable for a larger network.

In the conventional SpaceWire router, the routing table contains two fields (Logical destination, Physical output port). The router forwards packets only based on logical destination, which cannot identify different flows. In SDSpW, the router adopts a multi-field flow table as shown in Fig. 5, which contains source address, destination address, priority, action and counter.

Fig. 5. Multi-field flow table

The router matches source and destination addresses to identify different flows, and obtains the priority and output port of the flow. The counter is used to record the status of the network in different grain, such as flow level and port level. To support this mechanism, the nodes should also be made little modification. The header logical addresses add source address following the destination address.

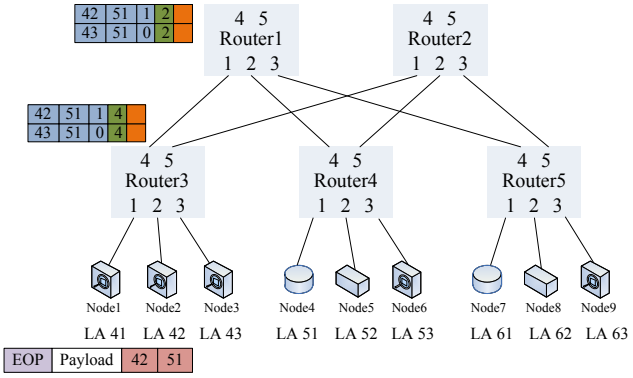

Fig. 6. Multi-field forwarding

The multi-field forwarding mechanism is shown in Fig. 6. Packets are transferred form node 2 (logical address is 42) to node 4 (logical address is 51). When receiving the packets, the routers lookup flow table, and forward the packets based on the output port and the priority. From the above figure, we can see that the priority of flow <42, 51> is higher than flow <43, 51>. The forwarding mechanism based on priority is familiar with the algorithm shown in Fig. 4. The flow table and node can also be dynamically configured which makes the SpaceWire network more feasible and controllable.

B. Control Plane

In the control plane, SDSpW controller and router run openSpW, a customized protocol on top of SpaceWire - RMAP. The customized openSpW has an abstract view of the whole SpaceWire network, and monitors the network's status (such as congestion, nodes' failure). It can dynamically configure the routers and nodes both in the situation of path addresses and logical addresses.

IV. EVALUATION

In this paper, we evaluate the performance of SDSpW in the environment of mininet. Our evaluation seeks to: 1) the controller can monitor the state of whole network in real time, 2) the QoS of end-to-end is guaranteed. The topology of the simulated network is shown in Fig. 7.

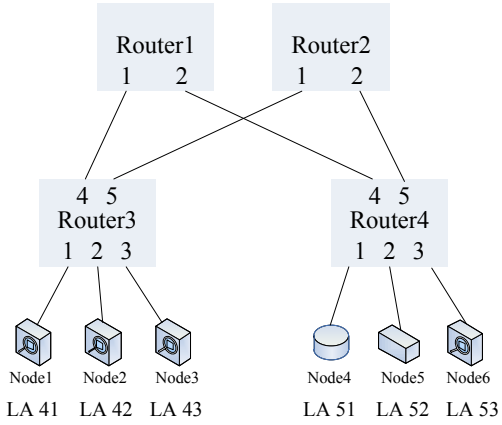

Fig. 7. The topology of evaluation SpaceWire network

As shown in Fig. 8, we establish the topology of simulated network.

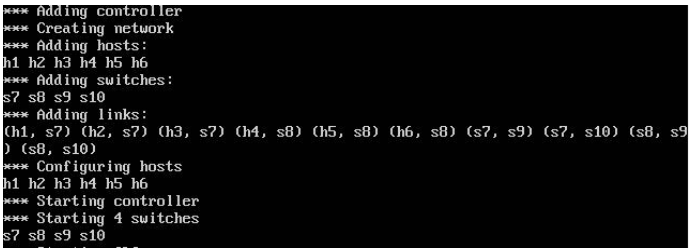

Fig. 8. The topology of established in Mininet

From the experiment, we find out that the controller can real-time monitor the status of whole network by reading the counters of the data plane. Moreover, as the bandwidth of link is limited, the performance of SpaceWire network degrades with the throughput increase of end-nodes. But in the priority mechanism, the performance of high-priority flow is guaranteed.

V. CONCLUSION

In this paper, we bring the thoughts of SDN technology into SpaceWire network which has several advantages, including open network topology, fine-grained control and QoS, etc. Then we propose a new software defined SpaceWire network architecture - SDSpW, which contains three roles: controller, router and end-nodes. The dynamically routing mechanism proposed in this paper achieves an adaptable, controllable network. As the evaluation results show, the controller can monitor the state of whole network in real time, and the end-to-end QoS is guaranteed.

ACKNOWLEDGMENT

The work described in this paper is supported by the project of National Science Foundation of China under grant No.61202488, No.61103182, No.61379147 the program for Changjiang Scholars and Innovative Research Team in University (No.IRT1012).

REFERENCES

[1] ONF, SDN architecture. 2014.

[2] ECSS-E-50-12-C. SpaceWire Engineering: SpaceWire-Links, node, routers and networks ESA-ESTEC. November 2008.

[3] ECSS-E-ST-50-51C. SpaceWire protocol identification. ESA-ESTEC, 2010.

[4] Mininet. http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet.

[5] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[6] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[7] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[8] K. Elissa, "Title of paper if known," unpublished.

[9] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[10] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[11] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

The SpaceWire Physical Layer Tester (SPLT)

SpaceWire Test and Verification, Short Paper

Pete Scott, Alan Spark
STAR-Dundee Ltd.
Dundee, Scotland, UK
pete.scott@star-dundee.com, alan.spark@star-dundee.com

Paul Crawford, Steve Parkes
University of Dundee
School of Computing
Dundee, Scotland, UK
psc@sat.dundee.ac.uk, sparkes@computing.dundee.ac.uk

***Abstract*—The STAR-Dundee SpaceWire Physical Layer Tester (SPLT) features hardware which enables it to perform tests across the SpaceWire standard from the physical and signal layer right up to the network and protocol layer. By incorporating components from other established STAR-Dundee products, including the Link Analyser Mk2 and Conformance Tester, the SPLT is the perfect tool that can be used throughout all stages of SpaceWire development from planning requirements through to production testing of flight components.**

The SPLT transmits SpaceWire LVDS signals at programmable swing and common mode voltages, slew, skew and bit speeds to test the capability of the Unit Under Tests (UUT) to maintain a SpW link without disconnecting. To do this, the SPLT can be configured as a SpaceWire interface on a SpW router. Alternatively, the SPLT can be configured to be placed in the middle of a SpW link between two SpW ports under test and manipulate the SpW signals in both directions.

Software running on a host Personal Computer (PC) is used in conjunction with STAR-Dundee's STAR-System device drivers and software to control the SPLT.

***Index Terms*—SpaceWire, Physical Layer, Signal Layer, Star-System, Spacecraft Test and Development Equipment**

I. Introduction

Throughout the specification, design, development and testing of a SpaceWire system, it is important that the system is tested and verified to the various levels of the standard. A number of tools already exist for testing a system's behaviour and performance at these levels, as is illustrated in Figure I-1.

Any problems in the physical and signal layer of the SpW system can be hard to detect and diagnose. This may be due to underlying problems not manifesting themselves in consistent, reliably reproducible symptoms. The SPLT's specialised hardware provides the capability to test and verify SpaceWire systems at these levels [1] [2].

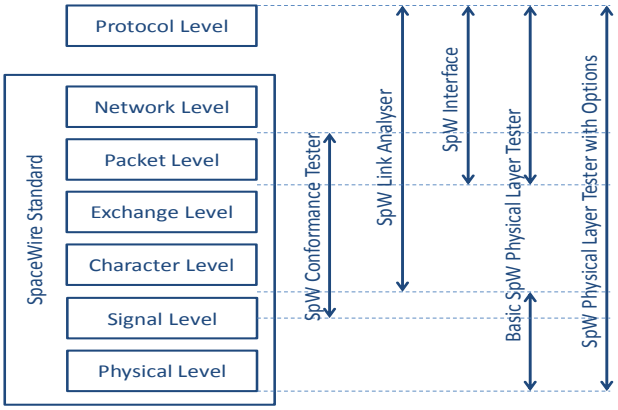

Figure I-1: Testing across the SpaceWire Standard

II. Overview of the SPLT

The front panel of the SPLT is shown in Figure II-1.

Figure II-1: Front panel of the SPLT

The SPLT features two Physical Layer Test SpaceWire ports, capable of performing tests at the physical and signal layers, as well as two normal SpaceWire ports. The configuration of these ports is shown in Figure II-2.

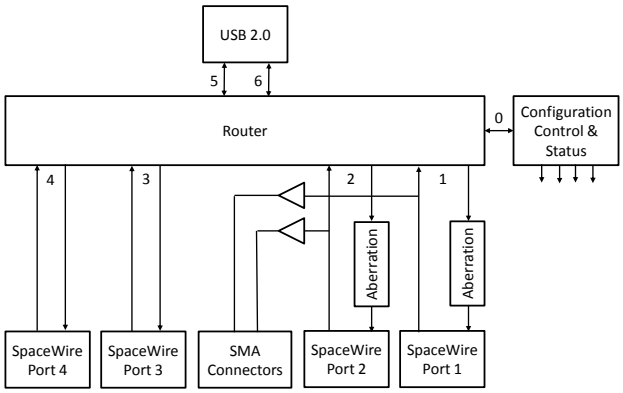

Figure II-2: Overview of the SPLT

The Physical Layer Tester is controlled by a USB 2.0 interface from a Host PC. This features two channels to the SpaceWire Router, allowing one channel to be used for control and configuration of the device and the other channel dedicated to SpW traffic data.

The outputs of the Physical Layer Test SpaceWire Ports (1 and 2) feature aberration circuitry, which is able to control the in-pair and data-strobe skew and jitter, as well as the slew, amplitude and common mode voltage of the LVDS signals.

The inputs from the Physical Layer Test SpaceWire ports are connected to high speed analogue buffers that allow easy interface of an oscilloscope to record the eye pattern of the SpaceWire LVDS signal received from the UUT.

III. SPLT SYSTEM OVERVIEW

The Physical Layer tester can be set up to interface to a UUT in three basic modes. In all three modes, the SpaceWire signals received from the UUT(s) can be buffered onto an oscilloscope. If the Link Analyser capability is selected, then a Logic Analyser may be interfaced to the mictor connector on the rear of the device to read the decoded SpaceWire traffic in a similar fashion to the STAR-Dundee Link Analyser Mk2 [3].

A. In-Line Margin Analysis

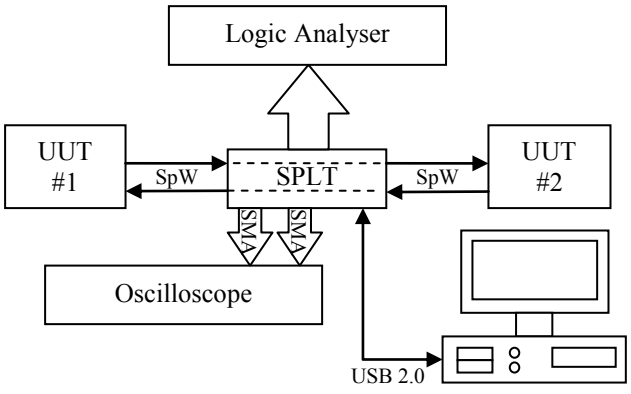

Figure III-1: In-Line Margin Analysis

The SPLT is placed in line with a SpaceWire link between two UUTs, or two SpW ports of the same UUT, as shown in Figure III-1. The SpaceWire data is passed transparently through the SPLT, allowing the two UUTs to communicate normally. Aberrations can be applied to the output SpW signals of the SPLT in one, or both, directions to explore the margins of either, or both, UUT devices.

B. Loop-Back Margin Analysis

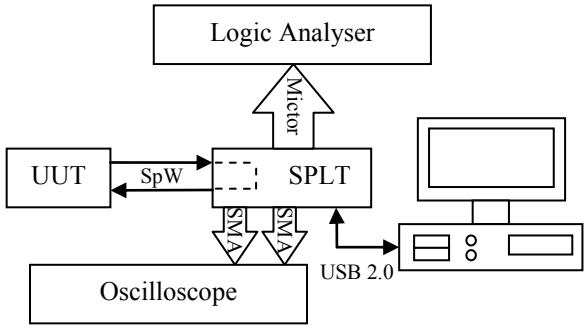

Figure III-2: Loop-Back Margin Analysis

A test can be performed where a UUT's transmitted SpW data is looped back to the same port through port 1 of the SPLT. Aberrations can be applied to the transmitted data to explore the receive margins of the UUT.

C. Interface, Routing Margin Analysis

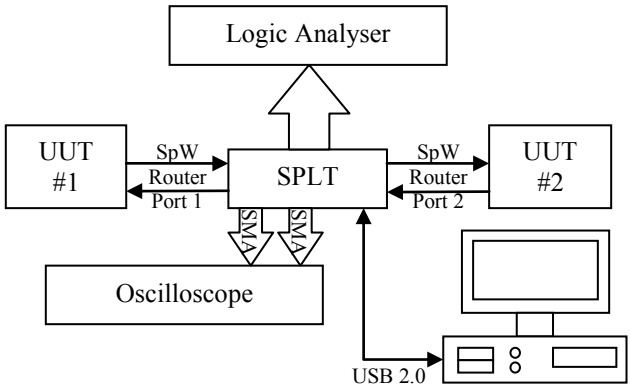

Figure III-3: Interface & Routing Margin Analysis

The SPLT can be configured in a similar way to the STAR-Dundee Brick Mk2 with the USB 2.0 port interfaced either directly to the four SpaceWire ports, or through a SpaceWire router, as illustrated in Figure II-2. This allows SpW data to be transmitted and received from the Host PC.

D. Conformance testing Analysis

If the Conformance testing option is selected on the SPLT, then the full suite of SpaceWire conformance tests can be performed from SpW link 1 [4]. The equipment is set up in the same way as shown in Figure III-2. This arrangement will be able perform a more comprehensive range of tests on the UUT than the existing STAR-Dundee Conformance Tester by taking advantage of the LVDS aberration capabilities of the SPLT.

IV. Testing with the SPLT

The SPLT Software provides Margin and Production testing modes, which were discussed in [2]. The control software now features graphical representations of the aberrations that are being applied. The user interface is shown in Figure IV-1. The graphical representations are described in sections IV.A and IV.B.

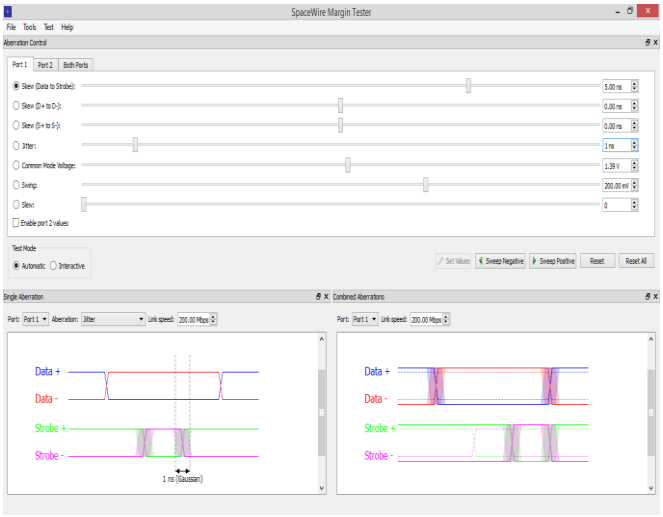

Figure IV-1: SPLT Control Software

A. Graphical representation of single aberrations

In order to assist the user in understanding the aberrations that they are applying to the signals, a graphical representation of the aberrations is provided in software. Figure IV-2 shows a SpaceWire link running with Nulls.

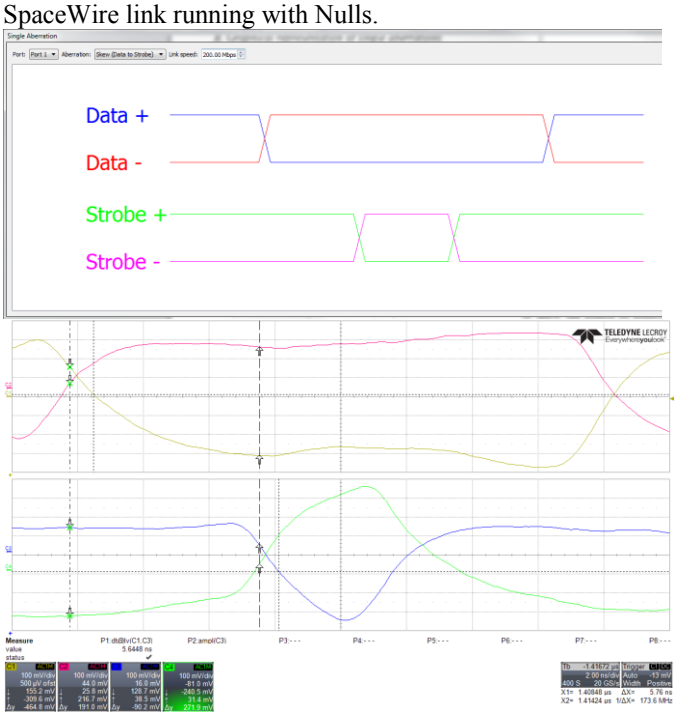

Figure IV-2: SPLT configured without aberrations

The top two waveforms in Figure IV-2 show a software representation of some SpaceWire transitions based on no aberrations being present. The bottom screenshot shows a measurement of the SpaceWire signal transmitted out of the Physical Layer Test Port, measured at the termination resistor on the other end of the link. Two vertical cursors have each been placed on adjacent transitions of the Data and Strobe.

In all oscilloscope screenshots in this paper, the Data is shown at the top of the oscilloscope screenshot, and the Strobe is shown at the bottom. The timebase is 2 ns per division, and the voltages are all shown at 200 mV per division. The scope is 1MΩ AC coupled. Measured Voltages must be divided by 2 to correct for the ×2 gain of the SPLT buffers that were used to obtain these waveforms.

A Skew of -2 ns is then set up in the software. The graphical representation of this aberration is shown in Figure IV-3. A dotted outline of the waveform shows how far from its ideal position it's being moved. In interactive mode, as the skew slider is moved sideways, the graphical representation is updated in real-time with the value of aberration and cursors to demonstrate the magnitude of aberration being applied. The user then commits this aberration to the SPLT by clicking "Set Values". In automatic mode, the graphical representation is updated in real-time as the increasing value of the aberration is sent to the SPLT.

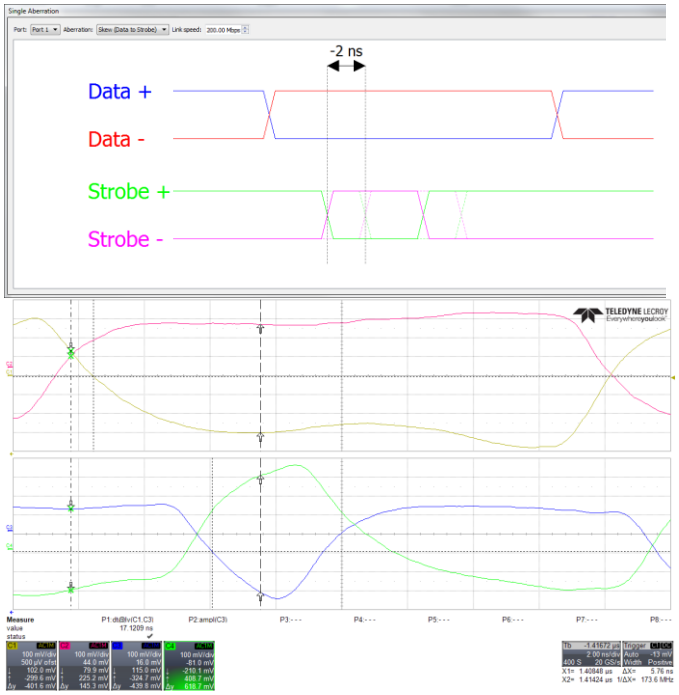

Figure IV-3: SPLT configured with 2 ns Skew

B. Graphical representation of combined aberrations

Labelling multiple aberrations with cursors and markers clutters the graphical representation. A second window is used to show the effects of combined aberrations. This window, along with the captured waveform, is shown in Figure IV-4

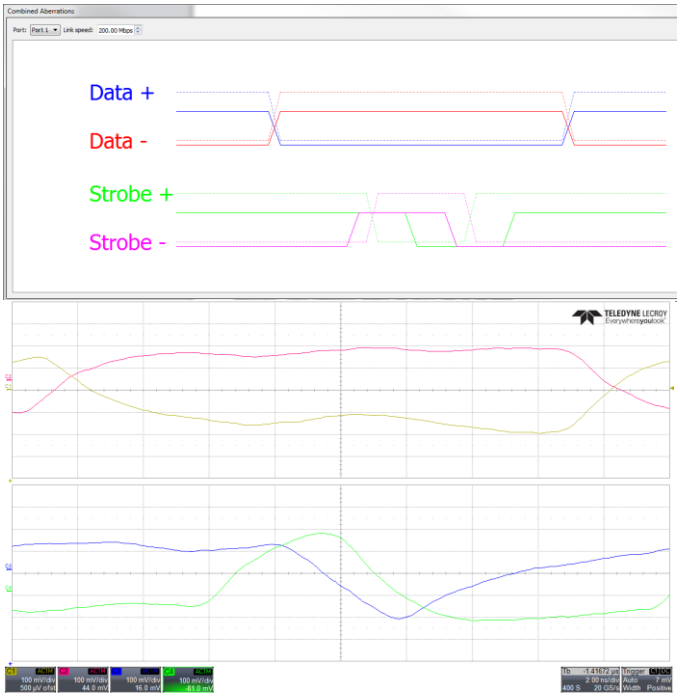

Figure IV-4: SPLT driving combined aberrations

The SPLT was configured to drive the following aberrations in Figure IV-4:

- -1 ns Data to Strobe Skew
- -3 ns of Strobe Plus to Strobe Minus in pair skew
- 1.16 Volts Common Mode
- 200 mV of Swing

The Graphical representation at the top of Figure IV-4 shows the many different ways in which the LVDS signal is now being deviated from its ideal parameters. The oscilloscope confirms the poorly degraded signal that is being driven out from the SPLT. Occasional disconnects are observed under these conditions.

V. User Calibration of the SPLT

The SPLT's physical layer test ports feature several analogue components that are calibrated from the factory for each individual unit. Users may wish to check this calibration, which may need updating as the environment in which the device is operated may vary, and as the components age. Users may also wish to calibrate the SPLT to a particular cable that they will be using with the SPLT.

The Software supplied with the SPLT includes a calibration application. In order to use this, the SPLT must be configured with a SpaceWire cable between ports 1 and 2 of the SPLT, and an oscilloscope connected to the analogue buffers on the receiver of the SPLT. This is shown in Figure V-1

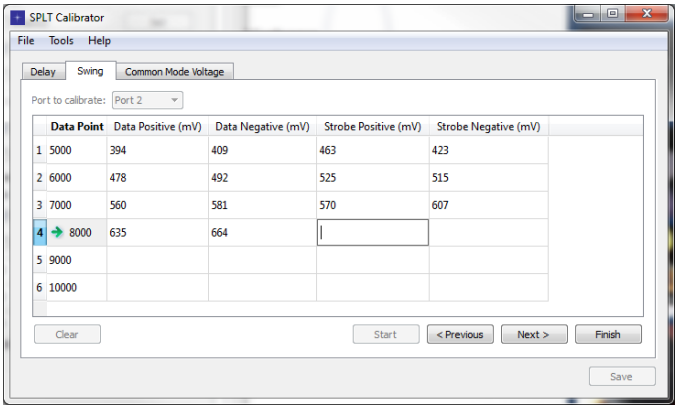

Figure V-1: SPLT Calibration Software

In Figure V-1, the LVDS Swing driven by Port 2 is being calibrated by the Calibration software. The SpaceWire cable loops this data into Port 1 so that these signals can be monitored on an oscilloscope. The calibration software steps through a number of linear data points to increase the swing. Measurements taken from the oscilloscope are entered into the appropriate boxes indicated by the green arrow on the left hand side of the current data point.

Once all of the data points are entered, the Software shows the calibration constants taken from these measurements against the factory calibration constants that the device was initially despatched with. The user calibration data can be uploaded to the SPLT for future use.

VI. Conclusion

The SPLT performs production and margin tests at the physical and signal layer in addition to higher level tests on a SpaceWire system. Protection against single point of failure in the device makes it suitable for interfacing to sensitive flight hardware. Such capability makes the SPLT one of the most comprehensive pieces of SpaceWire test equipment on the market.

VII. References

[1] P. Scott, S. Parkes, P. Crawford and J. Ilstad, "Testing SpaceWire systems across the full range of protocol levels with the SpaceWire Physical Layer Tester." International SpaceWire Conference, San Antonio, USA, 8 - 10 November. 2011

[2] A. Spark, P. Scott, S. Parkes, P. Crawford "Margin testing of SpaceWire devices" International SpaceWire Conference, Gothenburg, Sweden, 10 - 14 June. 2013

[3] Pete Scott, Steve Parkes, "SpaceWire Link Analyser Mk2: A New Analysis Device for SpaceWire Systems", International SpaceWire Conference 2010, St Petersburg, 22nd – 24th June 2010.

[4] Steve Parkes, Martin Dunstan, "Debugging SpaceWire Devices using the Conformance Tester", International SpaceWire Conference 2007, Dundee, 17th – 19th June 2010.

AXI-based SpaceFibre IP Core Implementation

SpaceFibre, Poster Paper

D. Jungewelter, D. Cozzi, D. Kleibrink, S. Korf,
J. Hagemeyer, M. Porrmann
Cognitronics and Sensor Systems Group
CITEC - Bielefeld University
Inspiration 1, 33619 Bielefeld, Germany
djungewelter, dcozzi, dkleibrink, skorf,
jhagemeyer, mporrmann@cit-ec.uni-bielefeld.de

J. Ilstad
TEC-EDP, ESTEC
European Space Agency
Keplerlaan 1, 2220AG Noordwijk ZH, The Netherlands
jorgen.ilstad@esa.int

***Abstract* — The steadily increasing demand of high-throughput interfaces, e.g., in satellite payload processing systems, drives the development of faster data transmission systems. The emerging SpaceFibre standard offers a multi-gigabit serial connection which is specified on the physical and data link layer while reusing the SpaceWire protocol specification on the higher protocol layers, thus enabling compatibility on the software layer. The AXI SpaceFibre IP core presented in this paper combines the SpaceFibre CODEC IP core developed by STAR-Dundee, a TLK2711 WizardLink Transceiver (meeting the SpaceFibre specification with up to 2.7 Gbit/s), and a DMA interface that connects the IP core to any AXI-based reconfigurable system-on-chip using FPGAs. The IP core configuration and initialization registers for the SpaceFibre RX and TX channels are accessible via AXI4-lite slave interfaces while the payload data is handled by a dedicated scatter/gather AXI-DMA core, which is connected to AXI-Stream FIFOs to provide the maximal possible payload transaction performance. The SpaceFibre IP core can be configured to implement 1 to 8 virtual channels. To evaluate the performance of the SpaceFibre IP core, we integrated it into the "Dynamically Reconfigurable Processing Module" (DRPM), a multi FPGA platform for satellite data payload processing. The AXI-based SpaceFibre IP core was synthesized on a Xilinx Spartan-6 LX150, utilizing in total 2668 slices and 36 BRAMs. For data segments larger than 1 kByte, a bandwidth of approximately 1.9 Gbit/s was achieved, corresponding to 95 % of the possible bandwidth. The IP core will be part of the ESA IP core repository.**

***Index Terms*—SpaceFibre, SpFi, AXI, FPGA, IP core, DMA, SoC, DRPM**

I. Introduction

The continuous improvement and innovation in satellite payload processing systems, often driven by advancements in the data acquisition systems like synthetic aperture radar (SAR) or hyperspectral imaging (HSI) systems, demands high bandwidth communication interfaces suitable for applications in the harsh space environment. An HSI frame with a typical resolution of 4000x4000 or 4000x8000 has a data size proportional to its resolution multiplied by its spectral bandwidth (typically 80 channels), resulting in high bandwidth requirements between the instrument and the processing system. An SAR frame is constituted by sending out a directed, pulsed radio wavefront and collecting the time delay, amplitude and angle of the backscattered signals. The amount of data per sample is comparatively small, but a complete SAR frame is many times larger than one HSI frame, requiring a high performance downstream payload processing system.

To meet this growing performance requirements, a new SpaceWire [1] based specification called SpaceFibre was developed by the University of Dundee together with the European Space Agency (ESA) over the last years (currently specification is Draft E1, [2]), offering serial high-speed data transmission (2.5 to 10 Gbit/s) by changing the physical and data-link layer of SpaceWire. Thus, SpaceFibre is compatible to SpaceWire on the upper layers. On the data-link layer, SpaceFibre introduces virtual channels (VCs). Each VC can be seen as a SpaceWire link; therefore, a bundle of up to 256 SpaceWire links can be replaced by a single SpaceFibre link resulting in a considerable mass reduction and a compacted system setup. The physical layer of SpaceFibre is implemented by utilizing high-speed Serializer/Deserializer (SERDES). Many modern FPGAs integrate these SERDES as Hard-IP-Blocks in their fabric. Additionally, discrete transceivers implementing this functionality are available, e.g., Texas Instruments provides a SpaceFibre compliant WizardLink Transceiver ASIC (TI-TLK2711A [3]) that is also available in a radiation tolerant package suitable for flight use.

The AXI SpaceFibre IP core, presented in this paper, integrates the STAR-Dundee SpaceFibre CODEC IP-Core [4] while using the TI-TLK2711A to implement the physical interface. The system interface is realized using an AXI4 [5] based interconnect. The AXI SpaceFibre IP core has been developed in the scope of the “Dynamically Reconfigurable Processing Module” (DRPM), a scalable platform for satellite payload processing deploying dynamic partial reconfiguration of FPGAs [6]. The DRPM is a heterogeneous embedded multiprocessor system composed of multiple external communication modules and processing modules as depicted in Fig. 1. The DRPM system was built based on the modular FPGA-based prototyping platform RAPTOR [7]. The flexibility of the RAPTOR system allows

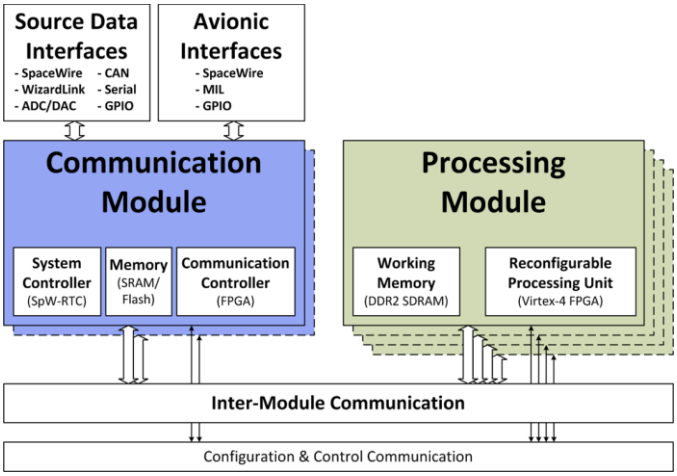

Fig. 1 Overview of the Dynamically Reconfigurable Processing Module

scaling the DRPM system by adding additional modules (called daughterboards (DBs)) to the system.

The processing modules are equipped with a Xilinx Virtex-4 FX100 FPGA and a DDR2-SODIMM-Socket for DDR2-Modules with up to 4 GByte. These daughterboards represent the main payload processing modules of the DRPM system. The FPGA available on a processing module can be dynamically reconfigured to adapt the processing module itself to changing environmental conditions. Furthermore, dynamic run-time reconfiguration is used to detect and correct errors caused by radiation (i.e., SEUs) by implementing a readback scrubbing scheme. Up to five processing modules can be connected in a DPRM to scale the processing power of the system.

Fig. 2 depicts the main components of the communication module, also referred to as DB-SPACE, which consists of a SpaceWire-RTC AT7913E [13], based on a LEON2-FT CPU (radiation hard, 50 MHz working frequency) as a fail-safe system controller, a Xilinx Coolrunner-II CPLD (XC2C384) combined with a 1 Gbit NOR-FLASH device (Numonyx PC28F00AP33EFA) acting as a configuration controller, and two Xilinx Spartan-6 (XC6SLX150/100) as EXT/INT-COMM FPGAs, which implement the external and internal communication controllers. In the context of this paper we focus on the external communication controller, which extends the interfaces available on the SpaceWire-RTC (adding a SpaceFibre link, four additional SpaceWire links, a MIL-STD-1553B connection and 32 additional GPIOs).

To the best of our knowledge no AXI-based SpaceFibre IP Core exists. In [8] and [9], pure implementations of the SpaceFibre CODEC IP-Core on a Xilinx Virtex-5 FPGA have been used to get performance evaluations and to do interoperability tests. In [10], the SpaceFibre CODEC IP-Core has been extended with a DMA controller on a Xilinx Virtex-4 FPGA which is directly connected to a commercial DSP. These implementations are not suitable for reuse in other target applications/embedded systems.

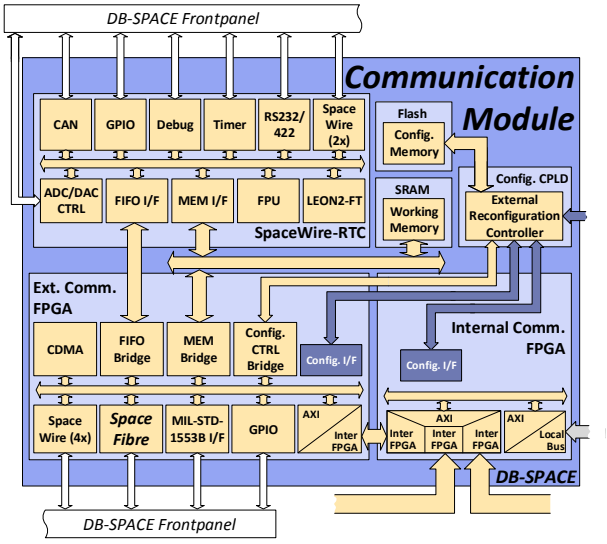

Fig. 2 Components on the communication module DB-SPACE

The following section (Section II) describes the architecture of the AXI Spacefibre IP core as well as its integration in the DRPM system. Section III is dedicated to the evaluation of the IP core regarding performance and power. Section IV concludes the paper.

II. Architecture of the AXI SpaceFibre IP Core

The AXI SpaceFibre IP core has been developed using Xilinx EDK, which offers a top-down approach for the implementation of an embedded system, comprising hardware as well as software development. The complete EDK system overview of the EXT-COMM FPGA is shown in Fig. 3. The different IP cores in an EDK-based system design are called PCores, they can be easily connected to an internal system bus like the Processor Local Bus (PLB) or the Advanced Microcontroller Bus Architecture (AMBA)-Advanced eXtensible Interface (AXI) bus without the need to use a hardware description language (HDL). Due to the advantages of independent read/write channels, configurable register slices and a powerful implementation of the interconnect matrix, the EXT-COMM system uses the AXI4 interconnect solution. The bus width is set to 32 Bit with a 100 MHz system clock resulting in an internal bandwidth of 3.2 Gbit/s.

The AXI4 interconnect is represented by the bus in the middle of Fig. 3. Both, the external rad-hard SpaceWire-RTC [13] and the embedded MicroBlaze processor can work as the central system controller. To connect the SpaceWire-RTC to the system bus, two independent IP cores are bridging the memory and FIFO interface of the SpaceWire RTC to the AXI4 interconnect, whereby the FIFO Bridge mainly forwards interrupts and mail-box messages. A central DMA (CDMA) controller offloads the system controllers from copying data between the external interface IP cores and the system memory. Two instances of 64 kByte BlockRAM are available as buffer memory for incoming and outgoing data streams.

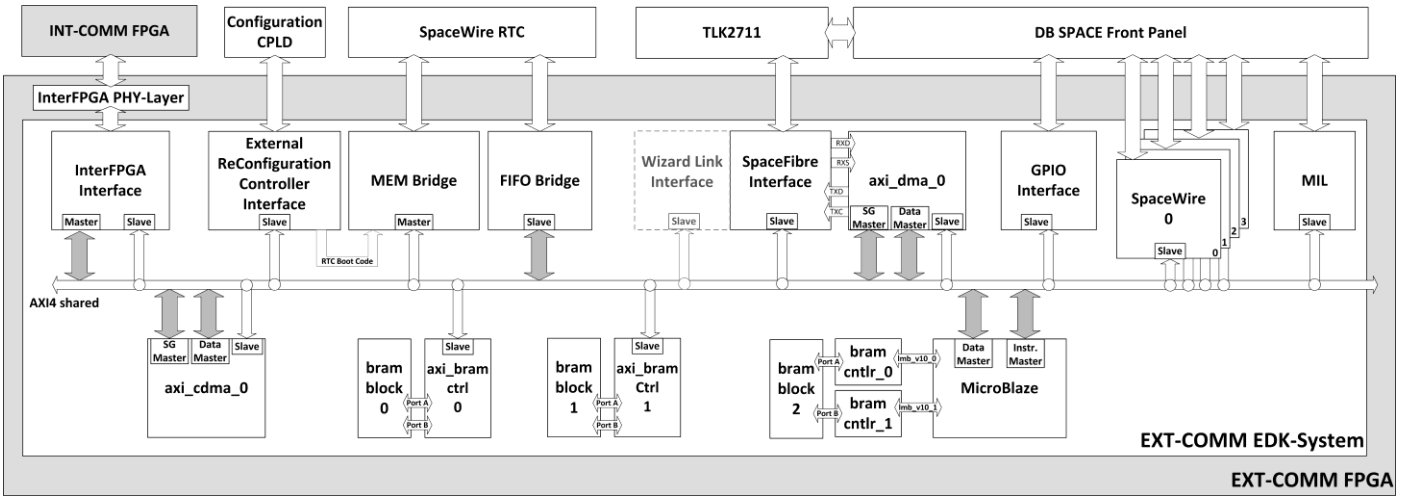

Fig. 3 Xilinx EDK-based architecture of the external communication FPGA

As an alternative to the AXI SpaceFibre IP core implementation described in this paper, a WizardLink IP core has been implemented. The WizardLink interface is a simple FIFO-based streaming communication core with minimal protocol overhead utilizing the TI-TLK2711A. It implements neither virtual channels nor bandwidth preservation or fault detection mechanisms. It is used as a reference IP-Core for later bandwidth evaluation and comparison.

In Fig. 4 a block diagram of the AXI SpaceFibre IP Core is shown. The AXI-lite slave interface provides access to a register bank for configuration and initialization as well as to a broadcast process that transmits or receives broadcast (BC) messages. BC receive messages are buffered in a FIFO, to avoid loss of messages. A control/status register is mapped to the TI-TLK2711A control and status signals to access all its functionalities. The management and status interface of the CODEC IP core signals is connected to a set of registers for initialization and configuration purposes. Additional registers are provided for interrupt handling.

Payload data is processed by a dedicated DMA controller [11] with an integrated scatter/gather engine. This allows achieving maximal performance using descriptor chains filling the outgoing (MM2S) and reading the incoming (S2MM) AXI-Streams. Two (data and control/status) streams are implemented for transmit and receive directions. The control/status stream is used to pass header information to/from the SpaceFibre IP core, i.e., start and length of routing and payload data as well as the VC to be used. Two finite state machines take care of writing (S2MM-FSM) and reading (MM2S-FSM) data to or from the FIFO buffers that connect the SpaceFibre clock domain clocked at 62.5 MHz with the AXI4 system clock domain at 100 MHz.

The **MM2S-FSM** handles the sequence of SpaceWire routing path bytes, control/status and data streams into the CODEC IP core. SpaceWire path routing bytes are defined in the very first DMA descriptor chain element (D0) of each data stream. Since the SpaceWire protocol allows an arbitrary number of routing bytes in front of each packet, but the receiver requires a 32 Bit aligned stream, a **ReAlign** unit shifts all following control/status and data words. This is necessary because D0 cannot simply be filled up with zeroes since routers will delete their path information out of the stream byte by byte. The aligned stream is forwarded to a virtual channel (VC) selectable through the A0 app-field in the control stream, whereby A1 contains the number of routing bytes and A4 the number of payload data bytes. App-fields A2 and A3 are reserved for future use. All app-fields registers are accessible through the AXI-Lite interface. Fig. 5 shows the VC stream as a composition of MM2S-Ctrl and Data Streams as discussed. D1 to Dn are the descriptors for payload data.

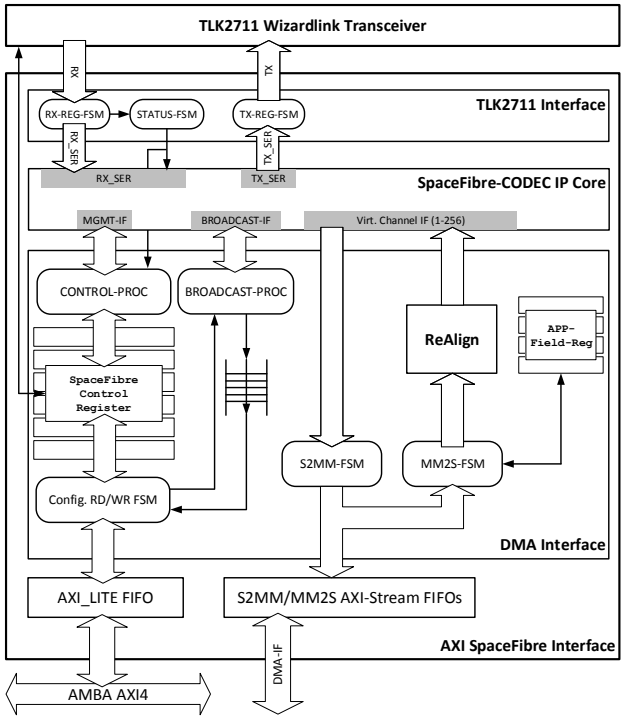

Fig. 4 Overview of the SpaceFibre EDK PCore

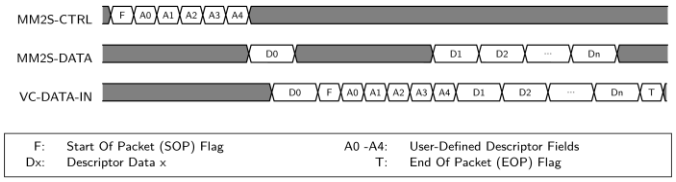

Fig. 5 MM2S stream composition

The **S2MM-FSM** is directly connected to the VC receive port of the SpaceFibre CODED IP core to fetch incoming data and to split up the stream into DMA controller data and status streams.

The **TLK2711 Interface** block connects the RX and TX channels of the SpaceFibre CODEC IP Core to the WizardLink Transceiver-IC. The data path is pipelined and utilizes FPGA-specific registers in the input and output blocks (IOBs) close to the pads to guarantee a small, deterministic input and output delay/skew. Additionally, this implementation relaxes the placement of the IP core relative to the IOs. Furthermore, the 32 Bit wide interface of the CODEC IP core is mapped to the 16 Bit wide interface of the TI-TLK2711A transceiver. Due the different data path widths, the PHY interface needs to be clocked at 125 MHz. Two different source synchronous clock domains at 125 MHz for RX and TX channels are implemented. While the TX clock is sourced by the FPGA, the RX clock is recovered from the incoming data stream by the internal PLL of the TI-TLK2711A transceiver. A Status-FSM monitors the Loss of Signal (LoS) condition of the TI-TLK2711A.

The **software API** of the AXI SpaceFibre IP included in the PCore allows configuring it in an easy way. Additionally, it enables the user to properly configure the DMA for scatter/gather transactions. Before a SpaceFibre read/write operation, the user sets up multiple linked transfer descriptors (a descriptor chain is created) which are then processed by the DMA engine. The main advantage of the scatter/gather transfers is the possibility to update the chain of descriptors during an ongoing DMA transfer. Therefore, the time taken by the processor for managing the descriptors can be hidden, reducing communication overhead. Furthermore, the utilization of the scatter/gather mechanism can be efficiently used to exploit the virtual channel (VC) feature of the SpaceFibre core, as it enables direct multiplexing of multiple SpaceWire connections.

Resource Usage: The required FPGA resources of the AXI SpaceFibre IP core are depicted in Table 1. The core has been implemented with two, four and eight VCs, including the dedicated AXI DMA IP Core that has been configured with a maximum stream length of 8 MByte. The AXI SpaceFibre IP Core configuration implementing eight VCs utilizes 12 % of the 23,038 slices available in the used Xilinx Spartan-6 LX150. 68 % of these slices are used by the embedded CODEC IP Core. For the implementations using 2 and 4 VCs, the number of slices used by the enwrapped CODEC IP Core is also larger than 60 %. Together with the dedicated AXI DMA IP core, the AXI SpaceFibre IP core configured with 8 VCs uses a total of 4408 slices, consuming almost one fifth of the available slices of the used Spartan-6 LX 150. In contrast to the resource utilization of the SpaceFibre IP core, the WizardLink IP core implementation requires only 480 Slices, summing up to 2097 slices including the AXI DMA IP core, which is less than one tenth of the total available slices. The overall utilization of the EXT-COMM FPGA implementing all components shown in Fig. 3 is more than 92 %.

Table 1 FPGA resources of the AXI SpaceFibre IP core

	SLICEs	REGs	LUTs	BRAM_16	BRAM_8	DSPs
Spartan-6 LX150	23,038	184,304	92,152	268	536	180
AXI DMA	1,617 7.0 %	4,687 2.5 %	3,813 4.1 %	8 3.0 %	4 0.7 %	0
AXI SpaceFibre IP core, incl. [SpaceFibre CODEC]						
2 VCs	1,922 [1,643]	4,889 [2,989]	5,202 [3,311]	0	24 [9]	2 [2]
4 VCs	2,207 [1,993]	5,492 [3,414]	6,219 [4,046]	0	28 [13]	2 [2]
8 VCs	2,791 [2,668] 12.1 %	6,693 [4,260] 3.6 %	8,183 [5,560] 8.9 %	0	36 [21] 6.7 %	2 [2] 1.1 %
DMA + SpaceFibre (8VCs)	4,408	11,380	11,996	8	40	2
WizardLink IP core						
	480 2.1 %	1,060 0.6 %	1,204 1.3 %	2 0.7 %	8 1.5 %	0
DMA + WizardLink	2,097	5,747	5,017	10	12	0

III. Evaluation of the AXI SpaceFibre IP Core

For performance evaluation of the AXI SpaceFibre IP Core, the core, embedded into the EXT-COMM FPGA of the DRPM system [6], has been analyzed, as discussed in the following.

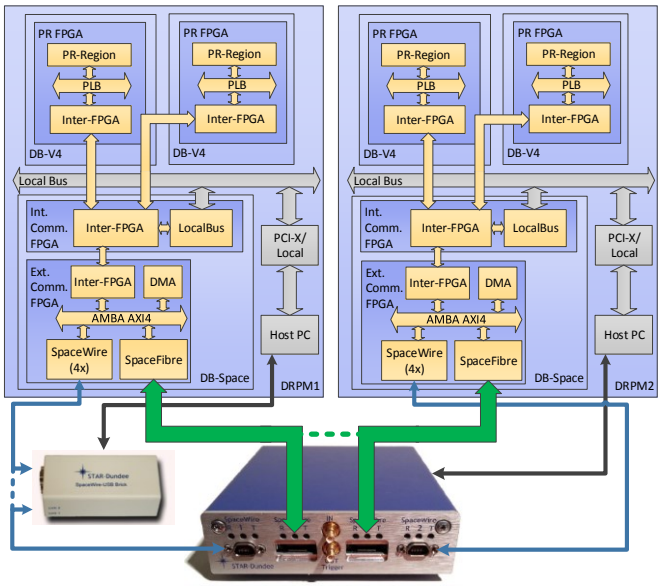

Fig. 6 Test environment used during evaluation

Fig. 6 shows the test setups used during benchmarking. Two DRPM systems (DRPM 1 and DRPM2) were used. Apart from a direct connection of the two DRPMs using SpaceFibre, a StarFire Analyser [12] was used to check the correct behavior of the link as well as the integrity of the SpaceFibre packets. Additionally, each DRPM has a SpaceWire link connected to the StarFire unit in combination with a SpaceWire USB brick to enable easy SpaceWire/SpaceFibre interoperability testing, i.e., testing of the SpaceFibre VC functionality.

The physical transmission of a SpaceFibre signal is based on high-speed serial data transmission. In the DRPM system, the transmission line comprises PCB traces as well as connectors and twinax cables. Although each part of the transmission line is impedance matched to a characteristic impedance of 100 Ohms, the transition from one transmission line to another causes signal reflections, reducing the opening of the data eye.

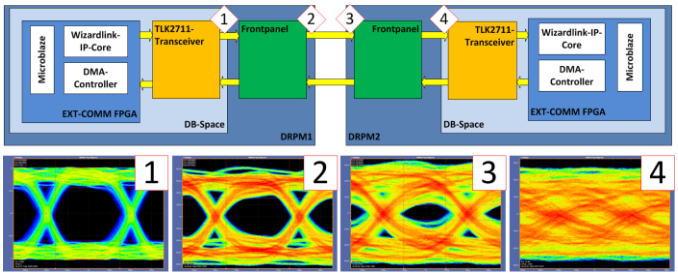

Fig. 7 Measurement positions the data eyes in the system

Additional insertion loss, caused by the dielectric and copper losses, is responsible for further reduction of the vertical and horizontal data eye opening. To characterize the signal degradation across the transmission line, the data has been measured at four different positions, as indicated in Fig. 7. Position 1 is located directly at the sender, while position 2 and 3 are at the beginning and the end of the twinax cable. Position 4 is located directly at the receiver. The different horizontal and vertical eye openings are listed in Table 2

Table 2 Eye measurement data

	0.5 m cable		1 m cable		3 m cable	
	Hor eye	Ver eye	Hor eye	Ver eye	Hor eye	Ver eye
TLK2711 [Pos 1]	309 ps	1180 mV	300 ps	1120 mV	299 ps	1100 mV
FrontPanel [Pos 2]	302 ps	619 mV	298 ps	644 mV	285 ps	617 mV
FrontPanel [Pos 3]	200 ps	197 mV	236 ps	167 mV	196 ps	116 mV
TLK2711 [Pos 4]	N/A	N/A	N/A	N/A	N/A	N/A

The length of the cable, represented by the increased insertion loss, (difference between Position 2 and Position 3) is clearly visible across the three different cables. Apart from the cable itself, the transmission lines on the PCBs and the connection from DB-SPACE to the frontpanel on both DRPM systems have a considerable impact on the total loss of the transmission line. In fact, the total additional length sums up to 1.5 m (excluding the cable length in Table 2), explaining the additional losses. Although the data eye is completely closed at the receiver for each cable, there are no data errors observed in the system using a PRBS-7 (BER < 1E-14). This is due to the use of equalizer techniques (i.e., decision feedback equalizer) in the receiver.

The bandwidth utilization of the system depending on the packet size is shown in Fig. 8. For packets bigger than 1 kByte, a utilization of more than 90 % is archived. The utilization saturates at about 95 % of the maximum bandwidth for large packets. For packets with a size of less than 256 Bytes, only half of the possible bandwidth is utilized due to the protocol overhead of the SpaceFibre protocol. Compared to the WizardLink implementation, the SpaceFibre implementation shows a significant higher protocol overhead. The WizardLink implementation however, as explained in Section II, only implements a protocol for point-to-point connections, and does not include features like virtual channels or error detection. The WizardLink implementation converges to 98 % bandwidth utilization for large packets.

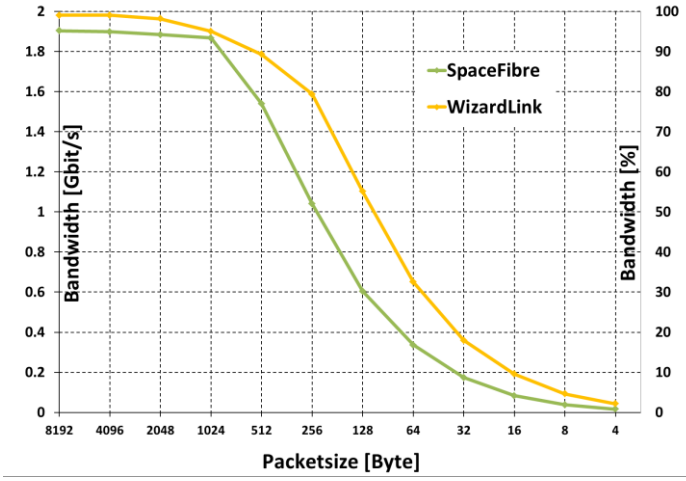

Fig. 8 Bandwidth utilization depending on packet size of SpaceFibre and WizardLink packets

The power requirements of the AXI-based SpaceFibre IP as well as the WizardLink IP core are summarized in Table 3. The values represent the power consumption of the IP cores under working and idle conditions operating at a signaling rate of 2.5 GSPS (2 Gbit/s net data rate). While the power consumption of the FPGA logic (Core Power) is reduced during idle conditions, the IO-Power of the FPGA and the power of the transceiver (TI-TLK2711A) stay nearly the same. This is due to the transmission of alignment patterns when no data is transferred. The small difference between the SpaceFibre IP core and the WizardLink IP core reflects the relatively large amount of power used by static infrastructure inside the FPGA. The clock circuitries of the FPGA alone (PLL and DCM) consume about one fourth of core power listed in Table 3.

Table 3 Power requirements of the SpaceFibre IP core implementation

	WORKING		IDLE	
	SpFi	Wizard	SpFi	Wizard
Core Power (FPGA)	0.28 W	0.20 W	0.18 W	0.14 W
IO Power (FPGA)		0.20 W		0.15 W
Transceiver Power		0.38 W		0.37 W
Total	0.86 W	0.78 W	0.7 W	0.66 W

The overall power consumption of the complete EXT-COMM FPGA implementing all components shown in Fig. 3 is about 3.3 W. The complete DB-SPACE daughterboard, including all components shown in Fig. 2 consumes 9.6 W under full load conditions.

IV. SUMMARY

The AXI SpaceFibre IP core combines a WizardLink Transceiver (TI-TLK2711A) suitable for flight use, the STAR-Dundee SpaceFibre CODEC IP core, and an AXI4‑based DMA controller. The IP Core is available as a Xilinx EDK-based PCore, allowing easy system integration and software development using the supplied API, maximizing IP reuse. The AXI4-based implementation allows the usage of the IP core in any AXI-based reconfigurable system-on-chip using FPGAs. As detailed in Section II, the DMA-based implementation features a scatter/gather unit for maximum performance and efficiency. Synthesis results based on the Xilinx Spartan-6 LX150 FPGA shows a total usage of 4408 slices (almost one fifth of the available slices) and 48 BRAMs (7 % of available BRAMs) implementing a full featured version including 8 virtual channels (VCs). A comparison with a minimal implementation using the WizardLink IP core results in only 2097 slices (about one tenth of the available slices) and 22 BRAMs.

As presented in Chapter III, the IP core has been tested and characterized using third party analyzer tools, ensuring interoperability with other SpaceFibre equipment. The IP core has shown a sustained bandwidth of 1.88 Gbit/s for data segments larger than 1 kByte, which corresponds to 95 % of the theoretical bandwidth (2.0 Gbit/s). The power requirement for the complete core including the external transceiver is considerably less than 1 Watt.

REFERENCES

[1] European Cooperation for Space Data Standardization, „ECSS-E-ST-50-12C,“ July 2008. [Online]. Available: http://www.ecss.nl/.

[2] Steve Parkes, Albert Ferrer, Alberto Gonzalez and a. C. McClements, “SpaceFibre Standard Draft E1” University of Dundee, Sep 2012.

[3] Texas Instruments, “User Manual TI TLK2711 1.6 TO 2.7 GBPS TRANSCEIVER”, Texas Instruments, 2008.

[4] A. G. Villafranca and A. Ferrer Florit, SpaceFibre VHDL IP Core User Manual, S. Ltd, Ed., STAR-Dundee Ltd, Jan 2013.

[5] ARM, “ARM AMBA AXI Protocoll Version 2.0 Specification”, ARM, Ed., 2010.

[6] J. Hagemeyer, A. Hilgenstein, D. Jungewelter, D. Cozzi, C. Felicetti, U. Rueckert, S. Korf, M. Koester, F. Margaglia, M. Porrmann, F. Dittmann, M. Ditze, J. Harris, L. Sterpone und J. Ilstad, “A scalable platform for run-time reconfigurable satellite payload processing” in *Adaptive Hardware and Systems (AHS)*, 25-28 June 2012.

[7] Porrmann M, Hagemeyer J, Pohl C, Romoth J, Strugholtz M. “RAPTOR – A Scalable Platform for Rapid Prototyping and FPGA-based Cluster Computing” In: *Parallel Computing: From Multicores and GPU's to Petascale, Advances in Parallel Computing*. Vol 19. IOS press; 2010: 592–599.

[8] Y. Otake, K. Hosokawa, Y. Sota, T. Tanaka, H. Hihara , “Performance evaluations and proposal to improve next-generation SpaceFibre protocol”, SpaceWire Conference 2013, Goteborg.

[9] T. Masuzaki, M. Nakamura, T. Kato, Y. Ido, T. Sasaki “Implementation and Interoperability Tests of SpaceFibre”, SpaceWire Conference 2013, Goteborg.

[10] B. Yu, S. Parkes, J. Franklin, C. McClements, P. Scott, D. Dillon, “High Processing Power Digital Signal Processor with SpaceWire and SpaceFibre Interfaces”, SpaceWire Conference 2013, Goteborg.

[11] Xilinx Inc., LogiCORE IP AXI DMA (v6.00a), PG021.

[12] A. Ferrer Florit, A.G. Villafranca, C. McClements, S. Parkes, “STAR Fire: SpaceFibre diagnostic interface and analyser”, SpaceWire Conference 2013, Goteborg.

[13] ESTEC, “SpW-RTC (AT7913E)”, June 2008, [Online]. Available: http://spacewire.esa.int/content/Devices/RTC.php.

Study and Implementation of SpaceWire Network Redundancy Technology Based on FPGA

SpaceWire Test and Verification, Short Paper

Chen Juan
School of Mechanical Engineering and Automation, Beihang University
Beijing, P.R.China
Chen.juan@buaa.edu.cn

Yang Shuai
School of Mechanical Engineering and Automation, Beihang University
Beijing, P.R.China
yshuai1990@gmail.com

Mei Hong
Beijing Aerospace Automatic Control Institute
Beijing, P.R.China
13810921938@139.com

***Abstract*—In order to improve the reliability of SpaceWire Bus, this paper makes a study of SpaceWire redundancy. In a spacecraft where SpaceWire is used, Redundancy is an important fault-tolerant technology to improve the reliability of the system. However, the regulation of redundancy does not be involved in the current standard of SpaceWire, so, it is necessary to study redundancy technology of SpaceWire. In this paper, without changing SpaceWire bus protocol, SpaceWire bus node with redundant functions is designed and redundant switching function is achieved on the node boards, routers and backbone links. IP logic of SpaceWire node is implemented in the FPGA. The scheme presents an Auto-Protection-Switch (APS) module which makes two independent SpaceWire nodes linked as mutual backup to achieve standby redundant switched function of SpaceWire bus. Redundancy switching process is as follows: APS continuously detects the working state of two mutual backup SpaceWire nodes in one board. When the Loss of Signal for Node A (LOS-A) is detected, APS uses Remote Defect Indicator for Node B (RDI-B) to send switching request code to the remote end through the altemate link. After receiving the switchover request data code, spare receiver module in the remote end generates switching signal to APS module at the same board and APS module immediately switches to the standby SpaceWire bus. At the same time, the confirming data is send to the local standby node. Then, local APS switches to the standby SpaceWire bus. Test results show that the switch time is 33us under the conditions of 200MHz transmission rate.**

***Index Terms*— Redundancy, Reliability, FPGA, node, Auto-Protection-Switch.**

I. INTRODUCTION

With the development of space technology, the width and depth of space exploration is increasing, which requires increasingly more higher performance of spacecraft. Some low-speed buses like RS-422/485, CAN and MIL-STD-1553 cannot meet the growing demand for bandwidth of data bus as the growth of satellite remote sensing data and payload data. In order to achieve a kind of high-speed and universal payload data processing system, a kind of high-speed, high reliability, low power consumption, long life and universal bus architecture, namely, SpaceWire bus is needed.

SpaceWire is a high-speed, point to point and full-duplex serial bus network, which is based on two commercial standards of IEEE 1355-1995 and LVDS. The standard of SpaceWire is based on the advantages of 1394 technology, ATM technology and Ethernet technology and takes into consideration the characteristics of the space applications at the same time. In addition to having a good EMC characteristic, SpaceWire shows better performance in aspects of exception handing, time deterministic, fault protection and detection.

However, technical specification of redundancy is not involved in the standard of SpaceWire bus. In order to improve the reliability of SpaceWire, this paper makes a study of redundancy technology for key equipment of SpaceWire. In this paper, the technical solution will be given to introduce how it can be designed and implemented. Then, with the test and analysis, the conclusion of redundancy will be given.

In fact, some experts have made some research on redundancy of SpaceWire. In [1], a variety of fault-tolerant methods are designed through the link, node and router in the baseband data processing network of the fourth geostationary meteorological satellite named FENGYUN. [2] shows that in spite of the deficiencies such as some issues about link bandwidth waste, bulky and power consumption, the redundancy application can tolerates single point of failure. A SpaceWire-based fault-tolerant solution with dual redundant link is proposed in [3]. A redundant program that manages to activate the correspondent backup link according to the error cause is proposed in [4] and [5].

II. TECHNICAL SOLUTIONS

In the present SpaceWire bus protocol, there is not specification about redundancy, so traditional SpaceWire bus board, even though with two SpaceWire nodes, can only work alone that either one node fails will leads the communication failure directly. In order to achieve redundancy feature of SpaceWire bus, another SpaceWire bus nodes with redundant

functions will be designed without changing SpaceWire bus protocol.

A. Redundant node

In order to achieve redundancy of SpaceWire bus, an Auto-Protection Switch (APS) is proposed, which links the two mutual backup SpaceWire nodes in one board to achieve standby redundancy switchover function. This scheme greatly improves the reliability of SpaceWire bus, as shown in Figure.1.

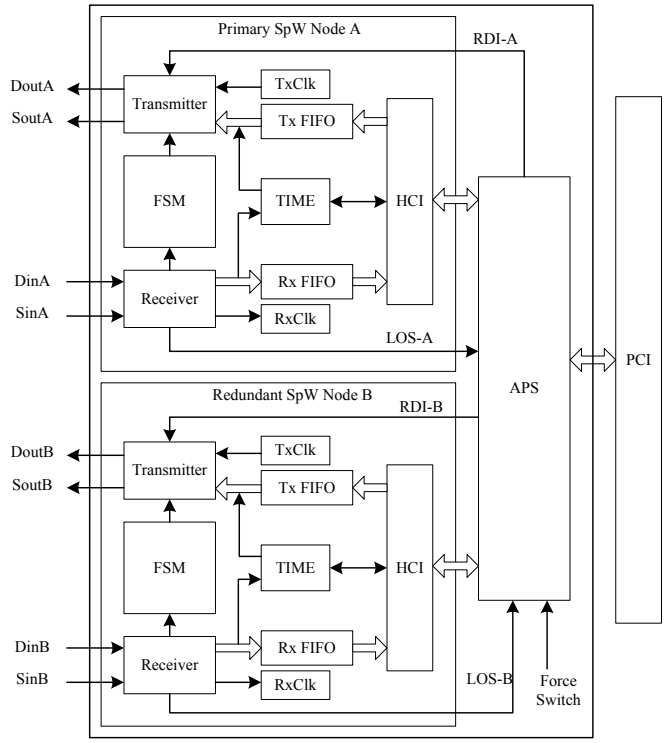

Fig. 1. SpaceWire bus board with APS

In this board, two independent SpaceWire nodes are linked with an APS module. When the board is working, APS continuously detects the working state of two mutual backup SpaceWire nodes in one board. When Loss of Signal for Node A (LOS-A) is detected, APS uses Remote Defect Indicator for Node B (RDI-B) to send switching request data code to the remote end through the alternate link. After receiving the switching request data code, the receiver module in the remote end generates switching signal to APS in the same board and APS module in the remote end immediately switches SpaceWire bus to the standby one. At the same time, the spare sending module in the remote end sends switching confirmation code to the local standby node and the standby node generates local switching signal to local APS after receiving switching confirmation code. At last, the local APS immediately switches local SpaceWire bus to the standby one. A redundancy switch is completed as shown in Figure.2.

When the link from the main sending module downstream to the main receiving module upstream or the two-way link between the master node upstream and the master node downstream has a fault, the principle of switching process is the shown in Figure 2. It is worth noting in the switching process when the two-way link between the master node upstream and the master node downstream has a fault, switching interlock circuit is provided to preventing repeated switching in bidirectional link.

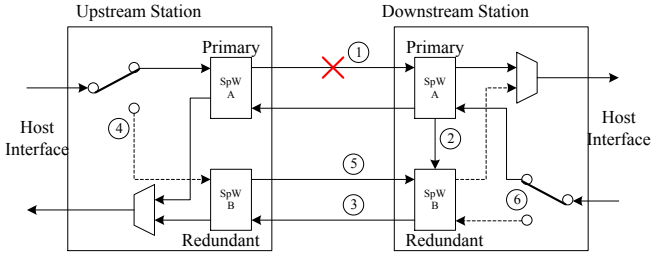

Fig. 2. Schematic diagram of APS switchover

B. Redundant network

According to the characteristics of SpaceWire bus protocol and redundancy techniques existed, this paper makes a routing switch-selected redundancy scheme of SpaceWire bus, as shown in Figure.3.

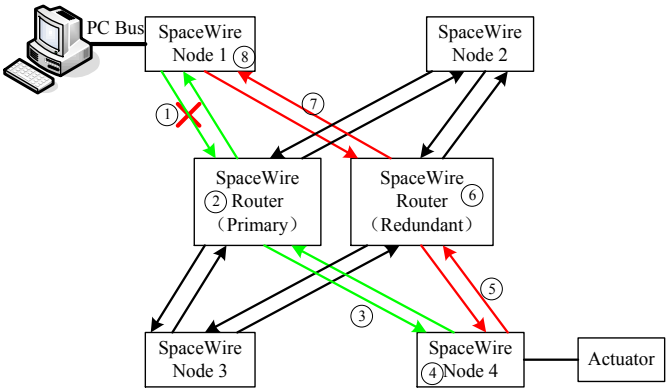

Fig. 3. Redundant network solution

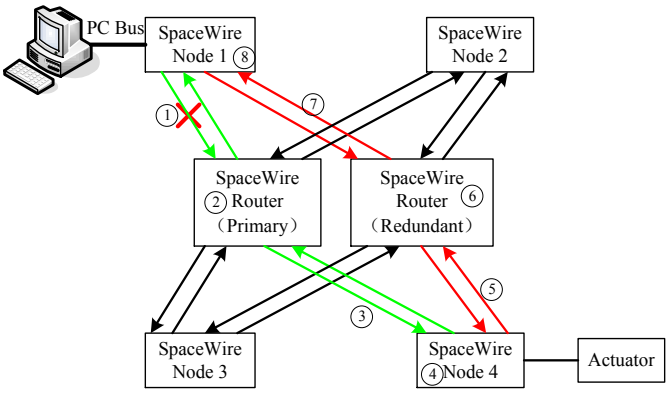

Fig. 4. Redundant network solution

As shown in Figure.4, when the network is working, redundancy switchover process is basically consistent with the switchover process shown in Figure 2. The only difference is that the switching request data code should be transmitted through the backup router to the appropriate switching node. When the switching is finished, actuator working mechanism remains the same as before.

III. DESIGN AND IMPLEMENTATION

if redundant network of SpaceWire wants to be set up, the network equipment must be designed. The network equipment of SpaceWire bus include network function node with SpaceWire bus interface and SpaceWire router.

A. SpaceWire node

This scheme adopts a hierarchical design method. The bottom-level includes all kinds of functional modules. On the bottom-level, it is signal-channel node IP which includes all the features of node and a highly versatile HCI host interface. It doesn't care about what kind of bus and processor host system uses so that the signal-channel mode IP can be referred to as SpaceWire node interface IP core. On the top-level, based on the node IP, dual-channel SpaceWire interface logic with LVDS signal-level is designed, which makes it convenient for testing and usage. The cooperation of this three levels achieves the protocol from physical layer to application layer of SpaceWire terminal node completely, as shown in Figure.5.

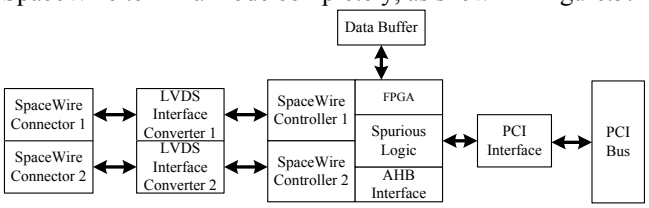

Fig. 5. Functional block diagram of node interface

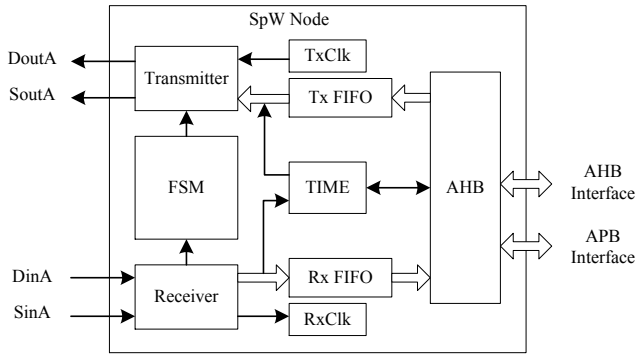

Fig. 6. Overall block diagram of node interface

In Figure.6, transceiver pins of each link are configured as LVDS mode. Each node IP contains following sub-modules: Host Control Interface (HCI), Transmitter (with Credit-Counter Module), Receiver (with Outstanding Counter Module), FSM, Receive FIFO, Transmit FIFO and Time Code Module. Two channels share a pll clock unit. The host system reads and writes each register of each node through PCI bus to achieve the control and data transmission of link state.

SpaceWire node device use card design with PCIEx1 interface, which can be plugged into the PCIEx1 slot of PC board directly. PEX 8311chip of PLX Technology Company is selected as PCIE bridge chip. IP logic of SpaceWire node is implemented in the FPGA. Cyclone Ⅱ's EP2C20F484 of Altera Company is selected as FPGA. MAX9152 chip is used to drive LVDS signals and standard serial DB9 connector is used as SpaceWire connectors.

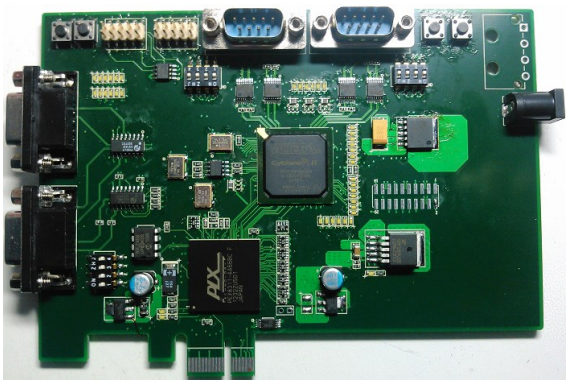

Fig. 7. SpaceWire node card

B. SpaceWire router

Since this scheme is only used to verify redundancy of SpaceWire, the router can be designed as a static manner. Each router provides 8 bidirectional portsand one mirror port. The mirror port is designed as one-way operation to detect the state and instruction of node and the information of data transmission in the network, as shown in Figure.8.

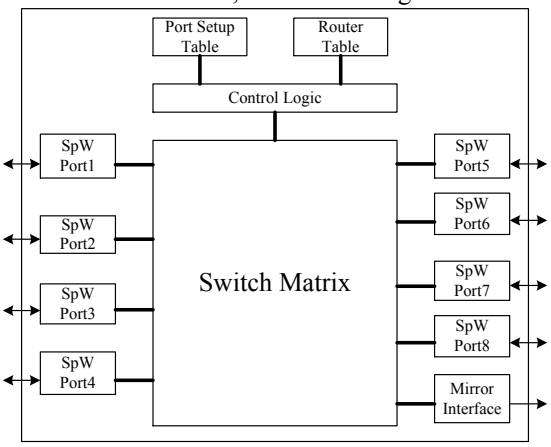

Fig. 8. block diagram of SpaceWire router

the core part of SpaceWire router is implemented in the FPGA. Virtex5's XC5VLX50T-667BGA of Xilinx Company is used as FPGA and MPC852T_50MHz is selected as CPU.

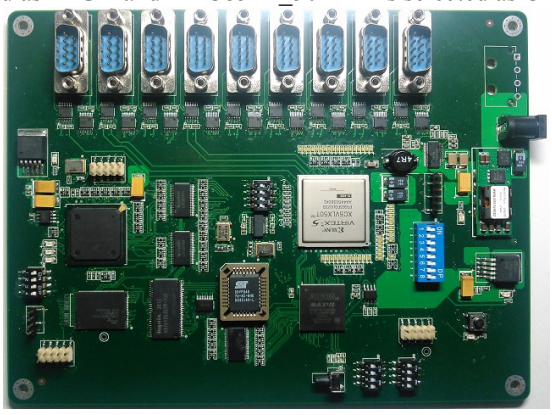

Fig. 9. SpaceWire router

IV. TEST AND ANALYSIS

The test includes two parts. One is to verify the performance of node card designed in this paper and the other is to verify the redundancy of SpaceWire.

Since the node and router have used Cyclone Ⅱ's EP2C20F484 of Altera Company and Virtex5's XC5VLX50T-667BGA of Xilinx Company, in addition to using traditional high-speed oscilloscope to test signals, this paper also uses two kinds of embedded logic analyzer from Altera Company and Xilinx Company and Modelsim, a kind of professional simulation tool.

After testing, the node board itself is turned out to work properly and it can achieve the basic functions of SpaceWire bus protocol. The test results of node board is not mentioned as it is more important to focus on the analysis of redundancy test.

A. System Recovery Time

System RecoveryTime: the time test nodes uses to return to normal working state after the overload.

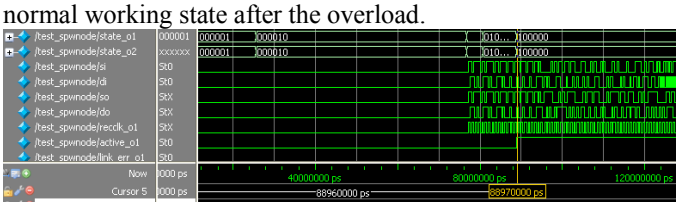

Fig. 10. SpaceWire bus system recovery simulation test chart

The actual test result:

System Recovery Time = 54.24us (under the conditions of 200MHz transmission rate).

B. Bus Switching Time

When a fault is detected by the master bus, the bus can switch automatically from the master bus to the alternate bus. Bus switching time is the time interval between detecting the fault and switching completely to make the communication recovery. Bus switching time's realistic expectation will be milliseconds and less than 1ms.

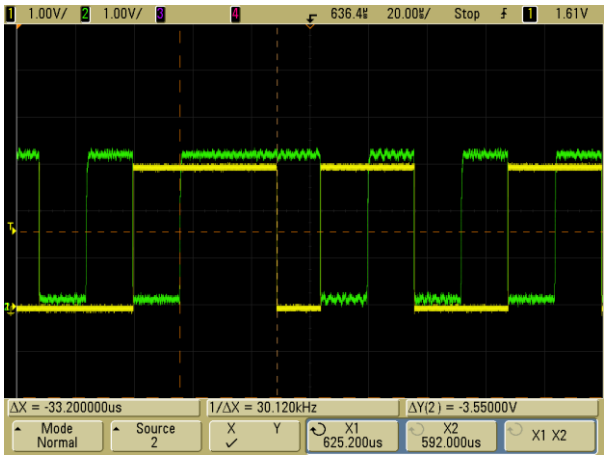

Fig. 11. SpaceWire bus switching time actual test chart

The actual result:

Bus Switching Time = 33us (under the conditions of 200MHz transmission rate).

C. Latency Time

Latency Time: the time interval from test nodes receiving the data to be transmitted to encapsulating the data as SpaceWire data packet and forwarding it out.

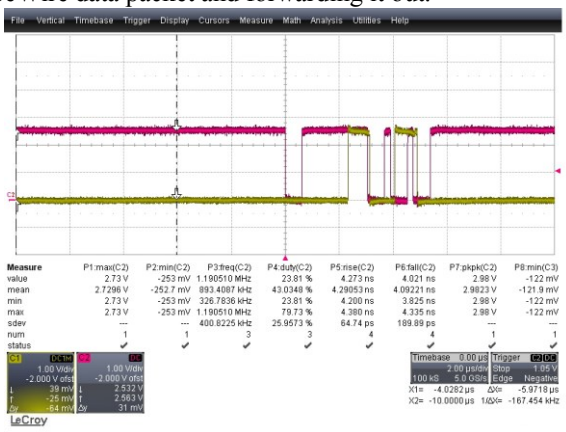

Fig. 12. SpaceWire bus latency time actual test chart

The actua test result:

Latency Time = 5.97us (under the conditions of 200MHz transmission rate).

D. Reset Time

System Reset Time: the time interval from test nodes software reset or power off restart to working normally.

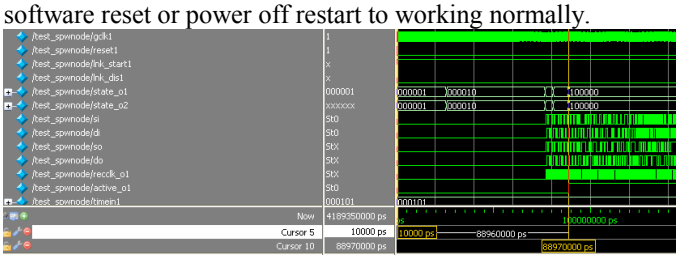

Fig. 13. SpaceWire bus reset time simulation test chart

The actual test result:

Reset Time = 88.9us (under the conditions of 200MHz transmission rate).

From the test result, the SpaceWire node designed in this paper fully meets redundancy requirements of SpaceWire. Redundancy scheme used in this paper not only be feasible, but has a good performance.

V. CONCLUSION

This paper achieves dual redundancy of node interface. The system redundancy switching time arrives millisecond and the data measured is 33us@200MHz; the actual transmission distance arrives 33m@200MHz. The redundancy scheme in this paper is feasible. Test results show that APS protection switching function is efficient to the upper application, which verifies the feasibility of APS on redundancy function of SpaceWire. As a result, the scheme further improve the reliability of SpaceWire bus.

ACKNOWLEDGMENT

First of all, I'd like to give great appreciation to my teacher Professor Chen Juan to give a lot of guides. Secondly, I want to show my gratitude to my friends Jia Jitao and Lu Yang, who give me support and help during the research and paper writing.

REFERENCES

[1] Tian Hua, Design and Study for A New Data Transmission of Generation of Geostationary Meteorological Satellite, Master thesis, ShangHai: ShangHai JiaoTong University, 2008, pp. 9–45.

[2] Steve Parkes, SpaceWire for Adaptive System, NASA/ESA Conference on Adaptive Hardware and System, 2008, pp.78–82.

[3] Zhang Lei, Sun Caihong, Solution of Data Bus for Space Solar Telescope, Astronomical Research Technology, 2rd,Vol.6, 2009, pp. 142–146.

[4] Dr. W. Gasti, A. Senior, Modular Architecture for Robust Computation, International SpaceWire Conference, Osaka University, Nara, June 2008.

[5] Muhammed Fayyaz, Tanya Vladimirova, Fault Tolerant SpaceWire Routing Topology and Protocol, International SpaceWire Conference, St PeterSburg, Russia, June 2010.

A design of on-board dual-channel data handling method based on two FPGAs

SpaceWire Missions and Application, Short paper

Zou Yaopu
Key Laboratory of Infrared System Detection and Imaging Technology
Shanghai Institute of Technical Physics, UCAS,
Shanghai, China
zouyaopu@126.com

Jiang Jiayou, Han Changpei
Key Laboratory of Infrared System Detection and Imaging Technology
Shanghai Institute of Technical Physics, CAS
Shanghai, China
johnrita@163.com, changpei_han@mail.sitp.ac.cn

***Abstract*—SpaceWire is used on FY4 meteorological satellite as on-board data-handling network. Based on SpaceWire, this paper provides a design using two FPGAs and AT7911 chips for the dual-channel data processing of a payload on FY4 to ensure that data can be transported efficiently and reliably. This design has proved to be feasible when tested on the ground.**

***Index Terms*—SpaceWire, AT7911, data-handling, dual-channel**

I. INTRODUCTION

SpaceWire has been used on many space missions for its high performance, and also be employed on FY4 weather satellite. The atmospheric vertical interferometric detector, one of the payloads of this satellite, has two data channels transporting data separately at different speeds. Both channels together with other science instruments communicate using SpaceWire. In order to facilitate the usage of SpaceWire protocol, and enhance the reliability of data transmission link, two FPGAs are used on the data-processing board for sampling, processing, and packing data, writing package to DPRAM chips and notifying corresponding AT7911, a radiation-tolerant chip developed by Atmel to support SpaceWire, of where to transmit. The link status information read from internal registers of AT7911 can be help to decide whether to resend data or not, for the link between two nodes can break off, as ensures the continuity of the transfer of data packets. The link information are exposed to both FPGAS to help the two channels exchange link information, and to help operators know the real-time status of the two SpaceWire links. Furthermore, superfluous data can be stored in free FIFOs and DPRAMs temporarily when data transmission is jammed in a short time, which increases stability and flexibility of the system.

II. ARCHITECTURE OF DATA-PROCESSING SYSTEM

The atmospheric vertical interferometric detector is an infrared Fourier spectrometer designed for meteorological detection. It contains two channels named channel1 and channel2 in the following paragraphs. Channel1 deals with the data from the detector and transports the data in SpaceWire format. At the same time, it throws the infrared data gained from the detector directly to channle2 which compresses the data for functional verification for the following satellites. Data from all of the units will be processed in the data-processing system showed in Figure1.

Fig. 1. Data-Processing System

The data-processing board, the core processing board, is mainly made up of two FPGAs, two AT7911 chips, three 4 GB mass SDRAMs and other appendages like clock generator, storage chips, power module, and so on. Figure2 shows the architecture of this data-processing system. The main FPGA works for channel1 while the other compression FPGA controlled by the main FPGA works for channel2.

This system chooses anti-fuse FPGA as main FPGA to ensure its reliability when working in space. An external SRAM is connected with the main FPGA to cache visible data. An external DPRAM is used to cache packaged data for AT7911. AT7911, also known as SMCS332SpW, provides an interface between three SpaceWire links according to the SpaceWire standard ECSS-E-50-12A specification and a data processing node consisting of a Control Processing Unit and a communication data memory [1]. All the chips mentioned above are at high radiation hardened level.

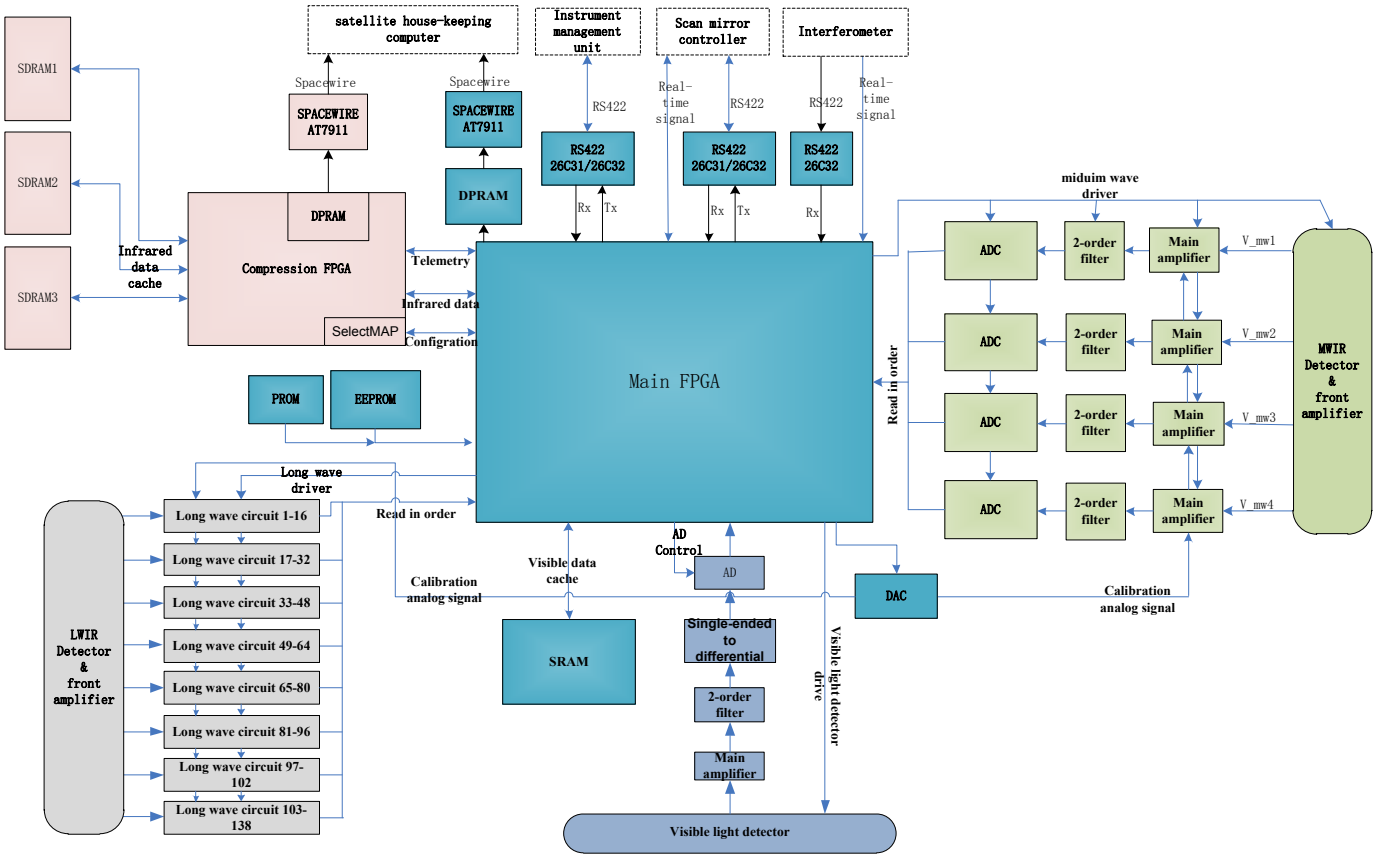

Fig. 2. Architecture of the Data-Processing System

Channel2 does the data compress job for which many complicated algorithms need to be done, so SRAM FPGA with plentiful resources, even enough to supply a large internal DPRAM, is chosen as the processing chip. In addition, plenty remote sensing data need to be cached, so three large-capacity SDRAMs with 4Gbit capacity make up the Ping-Pong buffer. High radiation performance would be considered when chosen the chips, but there's also a possibility that components without anti-fuse structure will be faulty radiated by high energy particles. In order to recover from fault and update compression algorithm, configuration of SelectMap mode is chosen as the configuration way. Configuration is controlled by the main FPGA. PROM or E2PROM is used for storing configuration information for the compression FPGA. Data of channel1 and channel2 output through the respective space-qualified 9-pin connectors.

Another part of this system is the detection data acquisition and controlling unit. Long wave has 128 independent detectors. It's not the IRFPA architecture so front amplifiers is needed to improve SNR (Signal to Noise Ratio). The long wave circuits contain main amplifiers, AD and filter, which is not shown in detail in figure. The visible light detector is CCD-array detector and the medium wave detector is IRFPA detector so they do not need front amplifiers as long wave detector. Actually they have similar architecture except visible light has a Single-ended to differential module. All the detectors controlled by the main FPGA, and their detection data read in order to the main FPGA.

RS422 is selected as the serial communication interface between different components for its usability.

III. Functional Description

FY4 weather satellite is a geostationary meteorological satellite. The atmospheric vertical interferometric detector carried on FY4 weather satellite collects interference data obtained by its core component Michelson interferometer. The usage of long-wave and medium-wave panel detectors allows the sounder gain three-dimensional remote sensing data [2], and this would greatly increase data size. Data processing is mainly handled on data processing board. As motioned above, this board contains two channels. Its main function contains data processing, telemetry and control, SpaceWire, and configuration. These would be introduced in detail in the following paragraphs.

A. Data Processing Tasks

1) Channel1

The main FPGA generates sampling sequence for medium wave detector and long wave detector, receives the infrared sampling data, and then caches the data in internal FIFOs. Visible data receives from visible-light detector and it would be stored in external SRAM. At the same time, auxiliary data contains telemetry information produced by interferometer unit, instruments management unit and scan control unit is gathered

by the main FPGA and packed together with the detection data. Figure 3 shows the data flow of channel1.

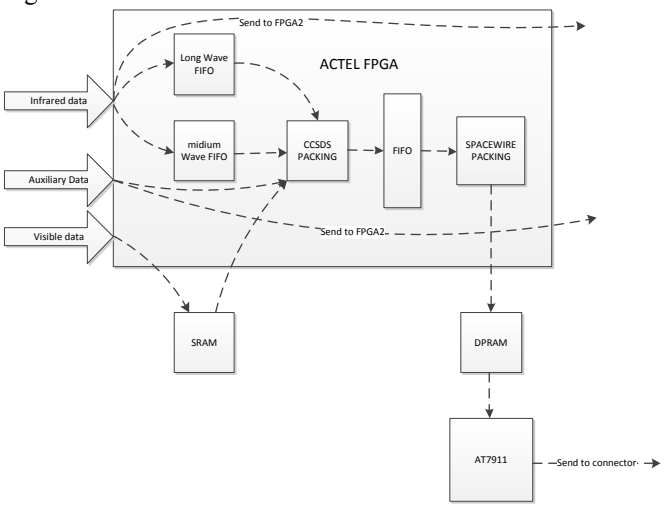

Fig. 3. Data Flow of Channel1

Detection data and auxiliary data are packed twice before sending outside. First time, infrared data read from long or short wave FIFO, visible data read from SDRAM, and together with auxiliary information are packed into CCSDS package. The packages will be stored in a FIFO before packing into short package in SpaceWire package format.

2) Channel2

As the core processor of channel2, the compression FPGA deals with all the compression algorithms. It receives detection data from FPGA1, and stores it in sdram. Because data compression is implemented for several frames, large sdrams with 4Gb capacity are employed. Figure4 shows the data flow of channel2.

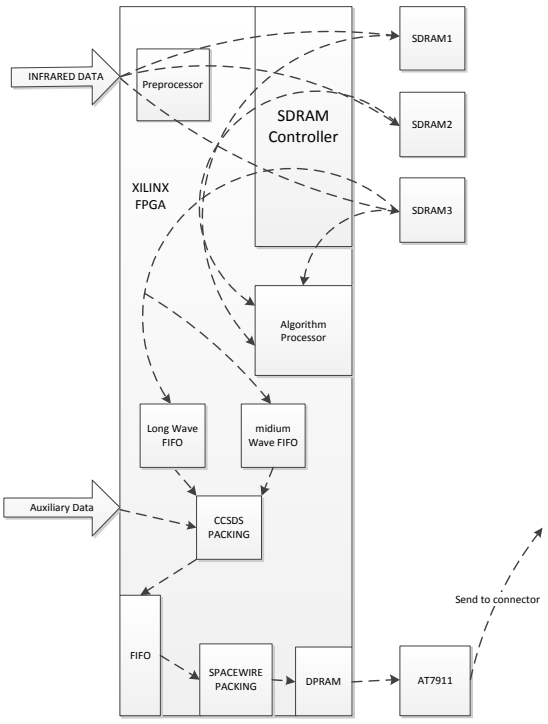

Fig. 4. Data Flow of Channel2

The strategy of SDRAM1 and SDRAM2 ping-pang cache infrared data allows data transfer and compress to perform simultaneously. SDRAM3 stores the half-compressed and compressed data. For example, when data buffered in SDRAM2, algorithm processor reads preprocessed data from SDRAM1 and half-compressed data from SDRAM3, and these two kinds of data would be calculated by the algorithm processor. Compressed data or half-compressed data would be written to SDRAM3. At the same time, compressed data is read out from SDRAM3 to long wave FIFO and medium wave FIFO respectively, and then sent to package module before transfer to SpaceWire connector through AT7911. Data package and sending process of channel2 are similar to channel1, so they are not described in detail here for brevity.

Channel2 receives control instructions and some of the telemetry informations from Channel1 through a serial port at rate of 1Mbps, and returns its internal states through another serial port every one second.

B. SpaceWire

The two channels both use SpaceWire interfaces in order to obtain high speed and reliable data communications. Figure5 excerpted from reference [2] shows the block diagram of the protocol chip, AT7911E. Both of the channels have their own AT7911 chips, so it's easier to control the SpaceWire data transmission.

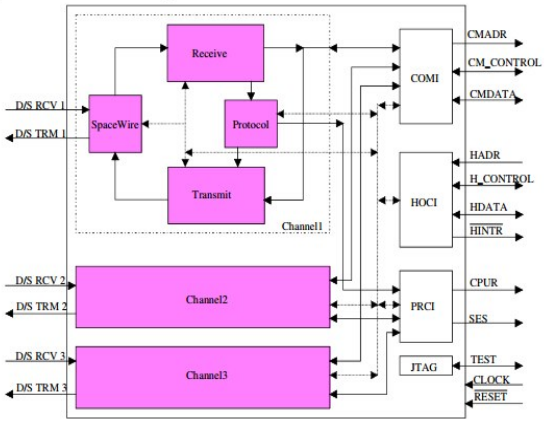

Fig. 5. AT7911E Block Diagram [2]

In this data-processing system, both channels use the same application as Figure6 shows. The chips communicate with other receivers using SpaceWire protocol. After power-up, the two ends connect automatically, then they both enter the Run-state and get ready to receive and send data. The processor, here is the FPGA chip, write data to the registers of the chips through the HOCI to command where and how many data should be sent from DPRAM. Then a following judgement will be made to determine whether resend or not.

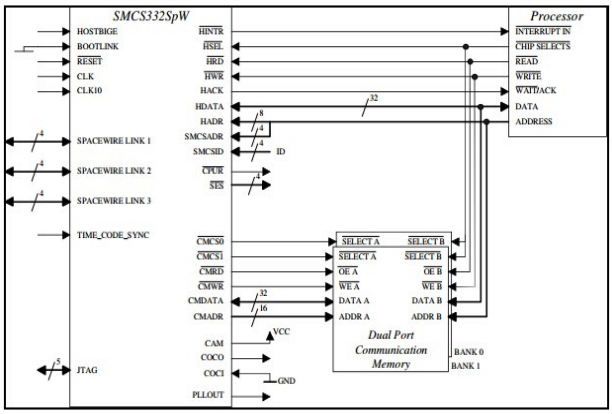

Fig. 6. Application for AT7911 [2]

For more reliable control of the chip, several methods are made to enhance robustness. First, if any errors are detected during data transfer, full reset of the chip will be made. Second, if register read acton lasts for too long, the FPGA forces the chip to initial state. Third, if the DPRAM is almost full, the FPGA stores the coming data into FIFO temporarily, this strive for extra time for the receivers.

C. Configuration

Channel1 use FPGA with anti-fuse architecture as its processor, so its configuration is done on ground. However, channel2 implement its configuration under the control of channel1 in SelectMap mode. Configuration data of channel2 is uploaded from ground to channel1, and stored in the E2PROM before writed to the compression FPGA. This kind of configuration ensuring that the compression FPGA, which is a kind of SRAM FPGA, can be totally reconfigured when it faulty radiated by high energy particles. And it will be also easy updated. Original configuration information is stored in PROM, so it's another kind of configuration for channel2.

IV. CONCLUSION

This design which is based on two FPGA chips uses a SpaceWire protocol chip AT7911 to manage the data transmission and flow control. The test results prove that with a good design on PCB layout, wiring and grounding, plus the function of re-sending of a lost packet, this design works efficiently and reliably.

References

[1] U.Liebstuckel, SMCS332SpW User Manual, ATMEL, 2007.

[2] ZhouYuan, Li Li, Zhang Jian-hua, Cui Wan-zhao, Zhao Jun-yi, Using SpaceWire In a Intellectualized Data Processor, Proceedings of International SpaceWire Conference, 2013

Thursday 25 September

Networks & Protocols 2 (Long)

A SpaceWire router architecture with non-blocking packet transfer mechanism

SpaceWire Networks and Protocols, Long Paper

Takayuki Yuasa
RIKEN The Institute for Physics and Chemical Research
2-1 Hirosawa, Wako, Saitama 351-0198, Japan
takayuki.yuasa@riken.jp

Masaharu Nomachi
Osaka University,
1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

Tadayuki Takahashi
Institute of Space and Astronautical Science, JAXA,
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan

Hiroki Hihara
NEC TOSHIBA Space Systems, Ltd.
10, Nisshin-cho 1-chome, Fuchu, Tokyo, 183-8551, Japan

***Abstract*—A new SpaceWire router architecture with packet segmentation and multiplexing capability is presented. The aim is to resolve a blocking phenomenon, or an increase of packet transfer latency, that can happen in a SpaceWire network formed by ordinary SpaceWire routers. To resolve the blocking of packet transfer in a router caused by worm-whole routing of a preceding packet, we designed, implemented, and tested, a non-blocking packet transfer mechanism that uses the SpaceWire-R protocol for segmentation of packets, acknowledgement, multiplexing of segments, and management of an end-to-end communication channel. The mechanism is implemented as an extension to an existing SpaceWire router. The SpaceWire-R part is implemented as an extension module to the existing SpaceWire router VHDL IP core, and we do not modify SpaceWire router specification at all. Details of the architecture, implementation result, and performance evaluation result are shown in the present paper.**

***Index Terms*— SpaceWire-R, packet segmentation, latency-constrained network.**

I. Introduction

The worm-hole routing mechanism specified in the SpaceWire standard can cause a so-called blocking when two (or more) packets flowing into a router share the same out-going SpaceWire port; one of the packets can go through the port first, and only after completion of transmission of the first packet, the "blocked" packet(s) can be transferred. This could lead to an unlimited increase of packet transfer latency of the blocked packet because the maximum packet length is not constrained in the standard, making the whole network undeterministic in the worst case.

To maintain the worst-case latency below an acceptable level, the maximum length of SpaceWire packet transfered in an onboard network is usually defined as a system-level specification. This is an approach taken by the SpaceWire-D [1] to complete a packet transfer within a single time slot, and thus to create a deterministic data transfer network.

In the present paper, we describe a new SpaceWire router architecture with a built-in capability of the maximum packet length limitation based on a packet segmentation mechanism and non-blocking data transfer mode where multiple packets can share the same out-going SpaceWire port. This new router architecture provides a "gateway" functionality which separates a SpaceWire network into an ordinary *blocking* SpaceWire subnetwork and a maximum-latency-constrained *non-blocking* subnetwork. The basic idea of this architecture was presented as [2] in the 18th SpaceWire Working Group meeting. Figure 1 shows a conceptual diagram of the gateway router and the network separation.

In the following sections, details of the proposed architecture and its implementation in VHDL are described followed by the performance measurement result showing how the worst-case latency is reduced by the architecture.

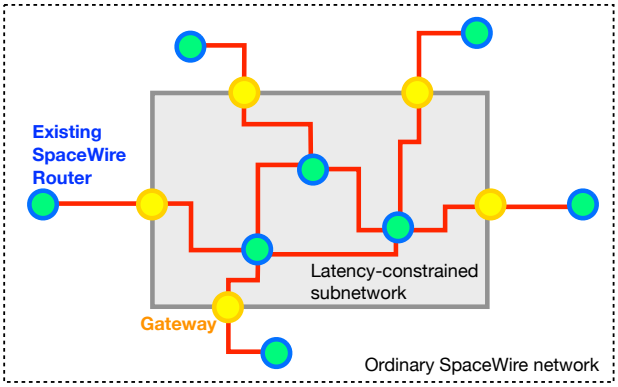

Fig. 1. Conceptual diagram showing an ordinary *blocking* SpaceWire subnetwork and a maximum-latency-constrained *non-blocking* subnetwork separated by a gateway router of which functionality the present paper descibes.

II. Segmentation and Virtual Channeling

Packet segmentation is a key to achieve the worst-case latency constraint in a SpaceWire network as noted above. Therefore, the gateway router that sends a payload packet coming from the ordinary SpaceWire network to the non-blocking network should have capability to segment a packet into multiple relatively small segments, and the receiving gateway has to unsegment the segments forming a complete SpaceWire packet. To avoid congestion within the non-blocking subnetwork, an end-to-end (i.e. gateway-to-gateway) flow control should be carried out by the gateway modules.

To realize these within the SpaceWire standard, we applied the SpaceWire-R upper-layer protocol [3] which is a successor of the NASA GOES-R Reliable Data Delivery Protocol (RDDP) and the Sandia Natinal Laboratory Joint Architecture Standard RDDP [4,5]. SpaceWire-R provides a packet segmentation function and end-to-end flow control, as well as communication channel control, i.e. Open/Close of a segment transmission channel between a pair of gateway modules.

Since SpaceWire-R is an upper-layer protocol of SpaceWire, all necessary communication between a pair of non-blocking gateways is performed using SpaceWire packets. Within the non-blocking (segmented packet transfer) subnetwork, all links operate as ordinary SpaceWire links but packet transfer is performed with segmented packets whose size does not exceed the predefined maximum segment length, thus allowing system designers to estimate the worst-case packet transfer latency in the subnetwork. Management of communication channel and packet segmentation/un-segmentation take place automatically in the "gateway" routers at the entrance and the exit of the non-blocking subnetwork, and therefore, no modification is required to existing standard SpaceWire devices connected to the non-blocking network.

Hereafter, we refer a packet tranfer using the SpaceWire-R segmentation and the end-to-end channel control mechanisms as the *non-blocking* packet transfer mode. The ordinary packet transfer using the existing SpaceWire routers is referred to as the *blocking-mode* packet transfer.

A functional block diagram of the gateway router is shown in Figure 2. In addition to ordinary SpaceWire ports, there are several additional modules to achive SpaceWire-R based packet transfer; SpaceWire-Conversion module which consists of the Framing and the De-Framing submodules. The Framing

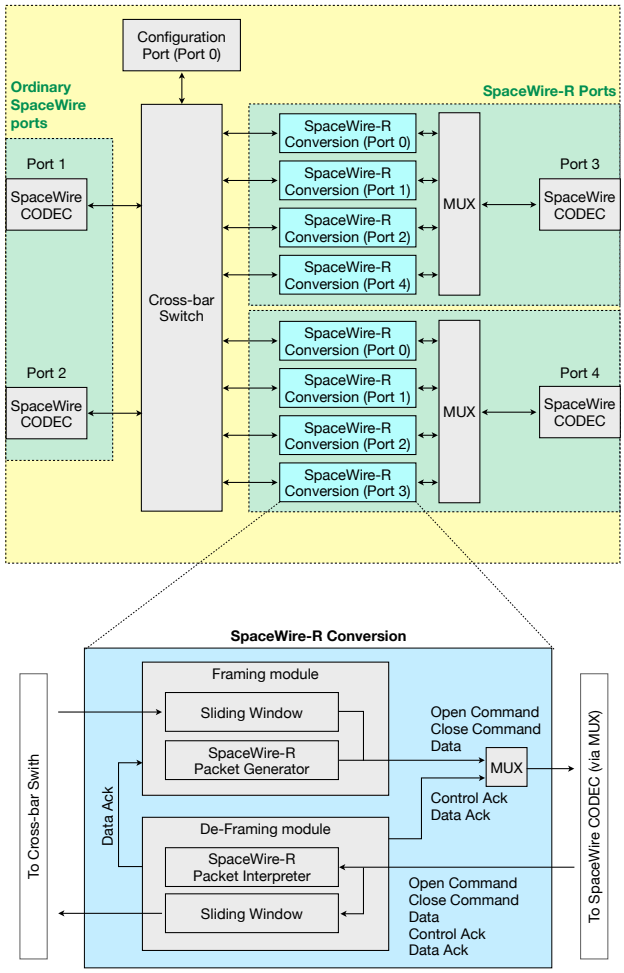

Fig. 2. Block diagram of the gateway router which transfers SpaceWire packets using the SpaceWire-R segmentation and end-to-end channel control mechanisms.

submodule carry out a segmentation of a SpaceWire packet creating multiple SpaceWire-R Data packets, and the De-Framing module unsegments them into a SpaceWire packe, and transfers to a destination SpaceWire port in the same router. All communication including channel control (Open/Close/Control Ack) and data transfer (Data/Data Ack) are performed using SpaceWire-R packets of which packet format is shown in Figure 3. The SpaceWire-R packet is a

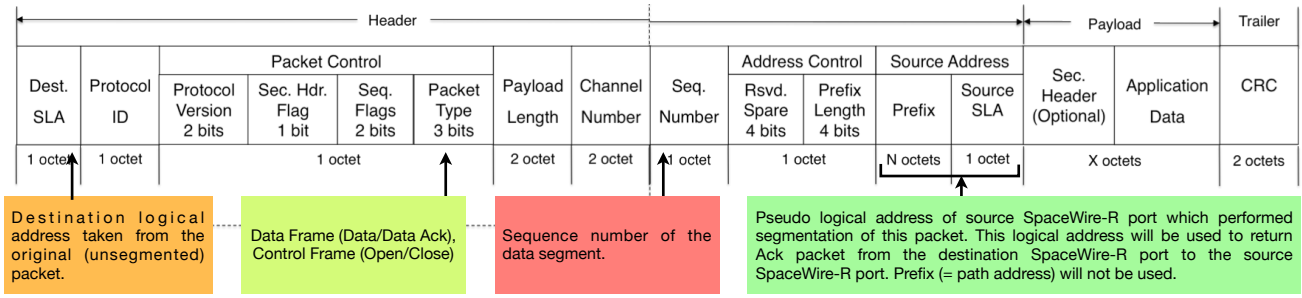

Fig. 3. Packet structure of the SpaceWire-R data transfer protocol [3]. All the packets transferred within the Non-blocking SpaceWire network have this packet structure, being either of Open Command, Close Command, Control Ack (Open/Close Ack), Data, and Data Ack.

subclass of the SpaceWire packet, and can be tranfered using existing SpaceWire routers without any modification inside the non-blocking subnetwork.

Figure 4 schematically shows the packet transfer procedure in the non-blocking mode. Below, we describe it by breaking down it into multiple steps.

1. A SpaceWire packet arrives, from the ordinary SpaceWire network, at the gateway router, and it is routed to a port connected to the non-blocking subnetwork. The gateway router constructs destination information based on the logical and path address in the header of the SpaceWire packet, and route the packet to one of the SpaceWire-R Conversion module of the outgoing SpaceWire-R ports. When a packet is written to the Sliding Window of the Framing module (Figure 2 lower panel), the module creats and send an Open command to a destination gateway router. The Open command packet, and also following Data-segment packet and a Close command packet, are routed within the non-blocking subnetwork based on the path address and logical addresses like ordinary SpaceWire packets. The Open command will be written to the De-Framing module of a SpaceWire-R Conversion module which corresponds to the destination SpaceWire port of the packet. The De-Framing module returns a Control Ack packet to acknowledge the command. To route this returning packet to the correct Framing module, it is necessary to identify the source Framing module that emtited this Open command using the source SLA (SpaceWire Logical Address) field. In the present architecture, we assign pseudo logical addresses to each SpaceWire-R Conversion module, and fill its specific value when a Framing module sends SpaceWire-R packet. A receiving gateway router and interleaving routers should have a look-up table entry corresponding to the pseudo logical address so that they can properly route Ack packets sent from a De-Framing module with logical addressing; the pseudo logical address filled in the SLA field is used as the logical address values of Ack packet, and the packet is returned to the Framing module using logical addressing.

2. Following the establishment of a communication channel between two SpaceWire-R Conversion modules, the Framing module starts sending Data segments. The segment size and the depth of the Sliding Window are parameters that should be determined in the system-level design as summarized in the SpaceWire-R specification document [3].

3. On receive of a Data segment, the De-Framing module in the destination SpaceWire-R Conversion module reconstruct the original SpaceWire packet, and outputs it to routed

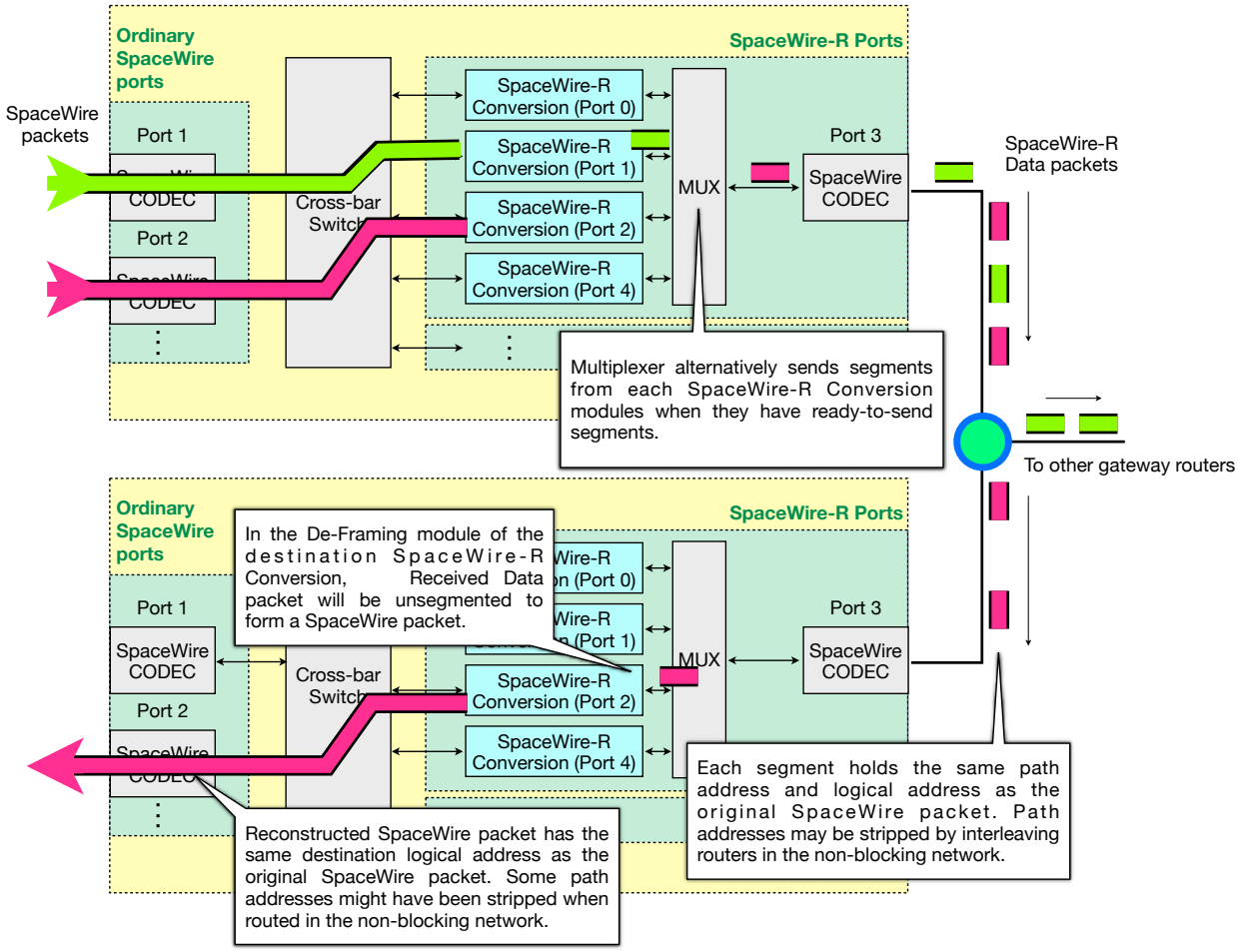

Fig. 4. Block diagram of the gateway router which transfers SpaceWire packets using the SpaceWire-R segmentation and end-to-end channel control mechanisms.

SpaceWire port selected based on a path address or a logical address available in the Data packet. If interpretation of the Data packet and reconstruction of payload data are successful, a Data Ack packet will be returned making Sliding Window pointer to slide.

4. When all outstanding Data segments were transferred and acknowledged, the Framing module sends a Close command to finalize the opened channel. When a Close Ack packet returns to the Framing module, procedure for transfer of a single SpaceWire packet completes. The Framing module can start processing another incoming SpaceWire packet.

These procedures can be concurrently proceeded by multiple SpaceWire-R Conversion modules in a SpaceWire-R port of a gateway router unless the same destination SpaceWire-R Conversion module is targeted. Thus, multiple large SpaceWire packets sent via the same SpaceWire link do not block each other except for short blocking period caused by transmission of Data segment. Duration of the short blocking can be controlled by changing the segmentation size parameterso as to fulfill the system requirement on the worst-case latency.

III. Implementation

We performed an R&D study on this non-blocking network architecture from 2013 to 2014. After the conceptual study, the above described mechanism was implemented as additional modules for the existing open-source SpaceWire router VHDL IP core. The major functionalities of the added modules include SpaceWire-R packet generation and interpretation, sliding window, transmission/receive-end-point control. Since the purpose of the present implementation is to validate the concept of the non-blocking packet transfer, the number of SpaceWire ports was limited at 3; two ports are ordinary SpaceWire ports, and the other is a port which transfers SpaceWire-R packets (segments). The SpaceWire-R port should be connected to a SpaceWire network where the worst-case latency is guaranteed by the tranfer of segmented SpaceWire-R packets. This routing switch can be regarded as a gateway from the ordinary SpaceWire network to the latency-guaranteed segmented network.

We confirmed that this newly developed IP core properly work on an FPGA (Xilinx Spartan-6 in our case), achieving the maximum SpaceWire link frequency of 100MHz (with a 50-MHz internal system clock for routing and SpaceWire-R-related processes). The logic footprint increase from the original router IP core is dominated by the memory used in the Sliding Window module and the crossbar switch structure; note that the number of cross-bar end points significantly increases because each SpaceWire-R port needs to implement SpaceWire-R Conversion modules for ports other than itself.

Technically, there is no limitation other than the maximum port number limitaion in the SpaceWire standard, and the number of SpaceWire and SpaceWire-R ports can be increased as long as the logic elements are available. The segment size is a parameter of the IP core, and selectable from available options of 128, 256, 512, and 1024 bytes in our implementation.

IV. Performance

To measure data transfer performance of the Non-blocking architecture, we simulated packet transfer using the newly implemented Non-blocking-mode SpaceWire routers in a VHDL testbench. Figure 5 shows a simulation configuration, and Table 1 lists parameters used therein. In the simulation, two packets are simultaneously generated by the Packet Source nodes, and transferred to the first router via 100-MHz SpaceWire links. The packets are routerd to Port 3 of the router which is connected, via a 100-MHz link, to the other router which the Packet Sink nodes are connected via 100-MHz link. For referring to the data transfer path between a pair of Source and Sink nodes, we use Channels 1 and 2 (see figure). We also performed the similar simulation using the existing ordinary SpaceWire router IP core same as the one used to implement the present Non-blocking architecture. It is obvious that the router-router link is shared by the two channels, a simultaneous transmission of packets from the two Source nodes will cause the blocking phenomenon in the first router.

"Latency" is defined, in the present analysis, as the time duration between the start of packet transmission in the Packet Source node (more specifically, first write to the Tx FIFO of SpaceWire CODEC) and the start of the receival in the corresponding Packet Sink node (receive of the first byte from the Rx FIFO of SpaceWire CODEC).

Figure 6 shows waveforms obtained in the blocking mode and non-blocking mode simulations with a packet size of 1024 bytes. A segment size of 256 bytes was utlitied in the non-blocking mode. Time duration where Packet Source and Packet Sink nodes are sending/receiving packets are indiacted with horizontal arrows. In the blocking mode, upper panel of Figure 6, transmission of the second packet (packet transferred from Source 2 to Sink 2) is suspended while the first packet (from Source 1 to Sink 1) is passing through the router-router link. This is a simple example of the blocking phenomenon.

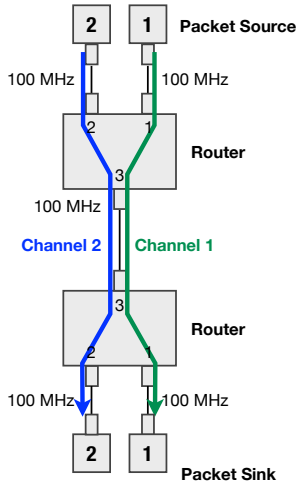

Fig. 5. Configuration of the simulation. Green and blue arrows show data transfer paths used by Packet Source 1 and 2. As the "Router" components in the diagram, a pair of ordinary SpaceWire routers or the newly implemented rouers with the Non-blocking data transfer mode was used depending on the simulation cases (see text).

In the non-blocking mode simulation, however, two packets are concurrently received by the two Sink nodes. A single packet is segmented into four 256-byte segments and alternatively transferred via the router-router link. This is why data are received intermittently (and alternatively) in the Sink nodes. Although the horizontal scales are not the same in these two pictures, one can note that smaller latency is achieved in the non-blocking mode.

We executed multiple simulations of these two cases with different packet sizes ranging from 16 bytes to 16kBytes. Observed latency values are summarized in Table II and plotted in Figure 7. Since the present non-blocking data transfer architecture requires an end-to-end Open/Close control before and after trasmitting a packet, and this is an overhead of this archtecture. Impact of this overhead time is particularly large for latency of preceing (first outgoing) packets shorter

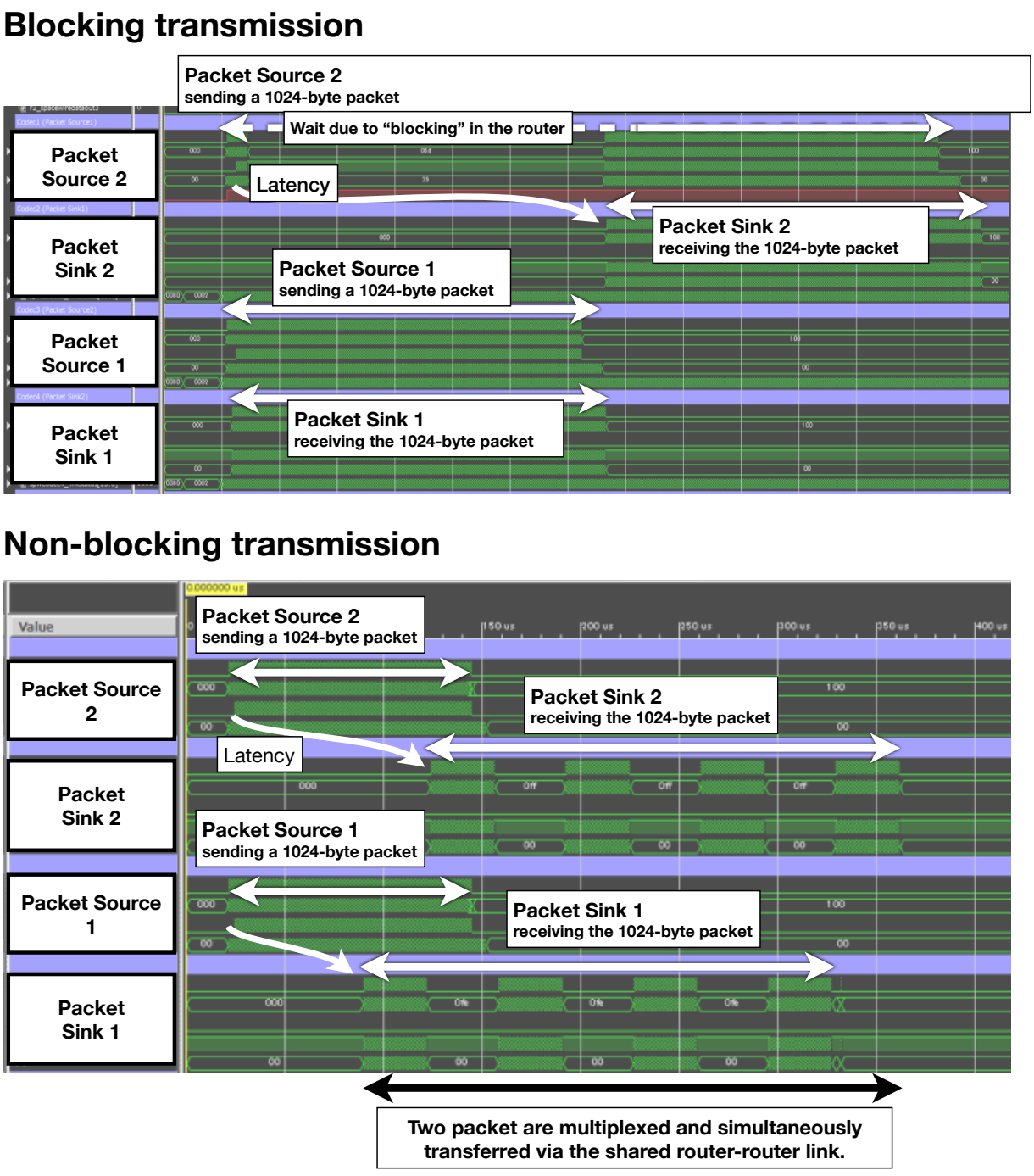

Fig. 6. Latency obserbed when two packets were simultaneously routed to the same port of a router in the Blocking mode (top) and the Non-blocking mode (bottom). In the bottom panel, the same results for the Blocking mode are also shown, in lighter colors, for easier comparison.

than a segment size; e.g. in the case of 16-byte packet, the non-blocking mode resulted 1.8µs of latency, but it increases to 10.44µs in the non-blocking mode (a factor of 5.8 increase.

On the other hand, in packet sizes that are larger than that of the segment size (256 bytes in this case), latency saturates at a constant value (68.2 µs and 102.44 µs for the two packets), and reduction from the blocking mode result is significant; e.g. in the 16-kByte packet case, a factor of 20 reduction is achieved, and higher for longer packets.

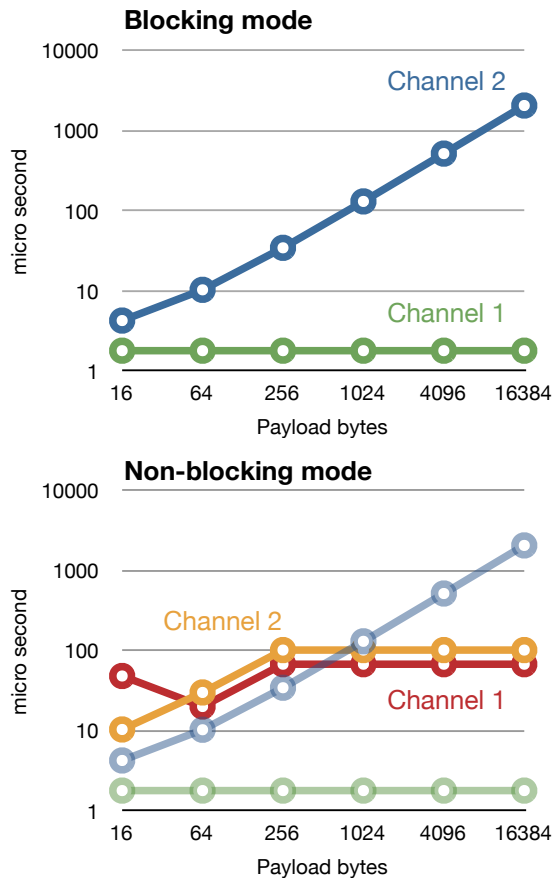

Fig. 7. Latency obserbed when two 1024-byte packets were simultaneously routed to the same output port of a router in the Blocking mode (top) and the Non-blocking mode (bottom; the same results for the Blocking mode are also shown, in lighter colors, for easier comparison). The segment size of 256 bytes was used in the Non-blocking mode simulation.

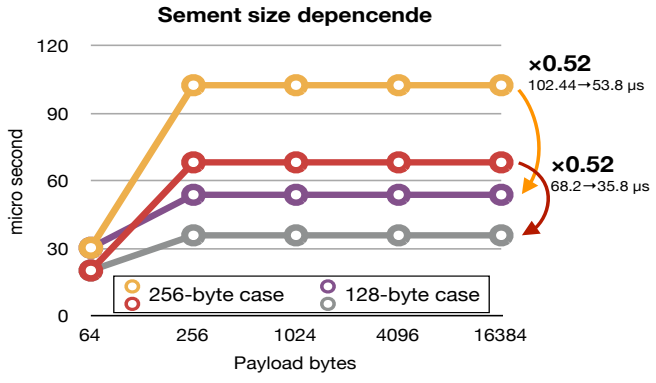

Fig. 8. Reduction of latency observed between segment sizes of 256 byts and 128 bytes. In the Non-blocking mode, latency is proportional to the segment size (c.f. it is proportioanl to the total packet length in the Blocking mode).

□PARAMETERS USED IN THE SIMULATION

Parameter	Value
Segment size	256 bytes
Sliding window depth	3
Incoming packet size	16, 64, 256, 1024, 4096, and 16384 bytes
Link frequency	all links operated at 100 MHz

Figure 8 shows a reduction of latency achieved when the segment size is halved to 128 bytes. In the blocking mode (ordinary SpaceWire router), latency observed in a blocking phenomenon linearly scales with the total size of the packet that impeding the transmission of the blocked packet. The non-blocking mode shows latency that is proportional to the segment size, and the worst case latency can be easily configured by setting an appropriate segment size to the gateway routers although this is not easily possible in the ordinary (unsegmented) SpaceWire networks requiring modifications of individual packet sending and receiving nodes.

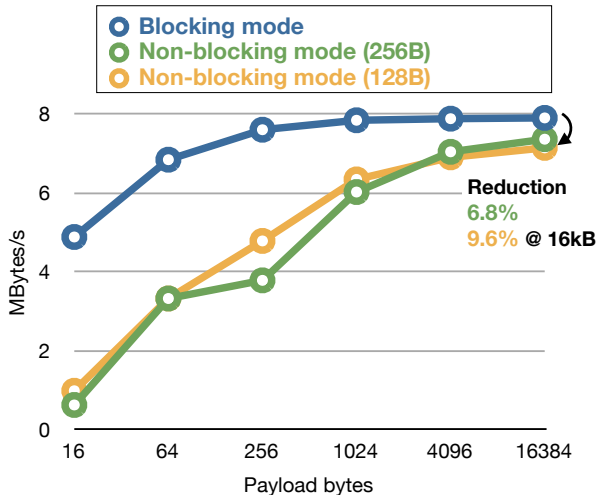

Fig. 9. Throughput calculated for each packet transferred between the Source-Sink pairs. Green and blue lines are for the blocking case, and red and orange lines are for the non-blocking case (256-byte segmentation).

To compare efficiency of bandwidth utilization of the shared link, we calculated throughput by dividing the total payload size (i.e. sum of the two packets) with a duration between the data send start time (when the first byte is written to the Source Tx FIFO) and the data receive end time (when the EOP is received by the Sink node). Results are tabulated in Table III and plotted for the blocking and the non-blocking cases in Figure 9. In short-payload cases, reduction is significant as expected, and reaches e.g. ~80% in the 16-byte case. This is explained as the overhead caused by extra time necessary for SpaceWire-R Open, Data Ack, and Close control. As the payload size increases, throughput reduction saturates at

RESULTS OF PACKET TRANSFER SIMULATION WITH THE BLOCKING AND THE NON-BLOCKING MODES. VALUES ARE IN MICRO SECONDS.

	Blocking mode		Non-blocking mode 256-byte segment		Non-blocking mode 128-byte segment	
Payload	**Channel1**	**Channel2**	**Channel1**	**Channel2**	**Channel1**	**Channel2**
16	4.3	1.8	10.4	48.6	10.4	32.4
64	10.4	1.8	30.3	20.2	30.3	20.2
256	34.7	1.8	102.4	68.2	53.8	35.8
1024	131.7	1.8	102.4	68.2	53.8	35.8
4096	521.1	1.8	102.4	68.2	53.8	35.8
16384	2077.6	1.8	102.4	68.2	53.8	35.8

TABLE III. THROUGHPUT CALCULATED FOR EACH PACKET TRANSFER. VALUES ARE IN UNITS OF MBYTES/S.

	Blocking	Non-blocking mode		Reduction from Blocking mode	
Payload		**256B**	**128B**	**256B**	**128B**
16	4.88	0.62	0.98	87.3%	79.9%
64	6.84	3.32	3.32	51.4%	51.5%
256	7.60	3.78	4.78	50.4%	37.1%
1024	7.84	6.02	6.34	23.2%	19.1%
4096	7.88	7.04	6.9	10.7%	12.4%
16384	7.90	7.36	7.14	6.8%	9.6%

~6.8% and ~9.6% of the blocking-mode case, and depending on the system requirement these values could be acceptable.

V. CONCLUSION

The non-blocking packet transfer architecture based on the SpaceWire-R upper-layer protocol is presented. VHDL simulation showed that the implementation works as designed, transfers multiple SpaceWire packets concurrently avoiding the *blocking* phenomena observed in the ordinary SpaceWire router. The worst-case latency reduction is effective in particular for relatively large SpaceWire packets for example longer than 10 kB. The worst-case latency is a controllable parameter that is proportionally dependent on the segment size and the link frequency.

This architecture is appilicable to those systems that carries sensor modules which output large telemetry data as a single (long) packet, and transferred to a mass memory via a shared network. If SpaceWire-D is applied to the network to maintain the determinism, the sensor nodes should implement time-division multiplexing. However the presented non-blocking architecture offloads the costs necessary to modify the sensor nodes, and the gateway routers assure the determinism by liming the worst-case latency. Thus, this architecture could be an option where there is a requirement to reuse existing devices and components with a network that is required to be highly reliable.

ACKNOWLEDGMENT

This work has been partially supported by the Special Postdoctral Reseacher program of RIKEN.

REFERENCES

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] Parkes, S., “SpaceWire-D draft specification”, available from SpaceWire Working Group website.

[2] Nomachi, M., “Non-blocking SpaceWire network”, 18th SpaceWire Working Group meeting, April 23, 2012.

[3] Yamada, T., “SpaceWire-R SCDHA 151-0.3”, Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency.

[4] NASA Goddard Space Flight Center GOES-R Project, “GOES-R Reliable Data Delivery Protocol”, 417-R-RTP-0050, 2008

[5] Gardner, M., et al., “Joint Architecture Standard (JAS) Reliable Data Delivery Protocol (RDDP) Specification”, SAND2011-3500, 2011.

SpaceWire-D on the Castor Spaceflight Processor

SpaceWire Networks and Protocols, Long Paper

David Gibson, Steve Parkes
Space Technology Centre
University of Dundee
Dundee, Scotland
{dzgibson, smparkes}@dundee.ac.uk

Chris McClements, Stuart Mills, David Paterson
STAR-Dundee
Dundee, Scotland

***Abstract*—SpaceWire-D is a deterministic extension to the SpaceWire protocol designed to satisfy hard real-time constraints on a SpaceWire network. This allows a single SpaceWire network to be used for both control applications and payload data-handling.**

The Atmel AT6981 Castor device is a LEON2-FT based system-on-chip with multiple integrated peripherals including an eight-port SpaceWire router and three internal SpaceWire engines each containing three DMA channels, an RMAP initiator, and an RMAP target.

This paper describes the SpaceWire-D protocol; the design of RTEMS networking software to test the protocol using the AT6981 system-on-chip; and the results of those tests.

***Index Terms*— SpaceWire, SpaceWire-D, deterministic networks, spacecraft onboard processing, AT6981**

I. Introduction

SpaceWire-D is a deterministic extension to the SpaceWire on-board data handling network [1] being designed by the University of Dundee for ESA [2] [3]. To provide a deterministic capability, SpaceWire-D uses time-division multiplexing and slices network time into a series of time-slots in which RMAP [4] transactions are executed. These transactions are grouped into a virtual bus system, where each bus consists of an initiator node, one or more target nodes, and the set of links that make up the paths between the nodes.

Figure 1 shows an example of a virtual bus with an initiator, three targets, and five links. The semi-transparent nodes and links are not part of the virtual bus.

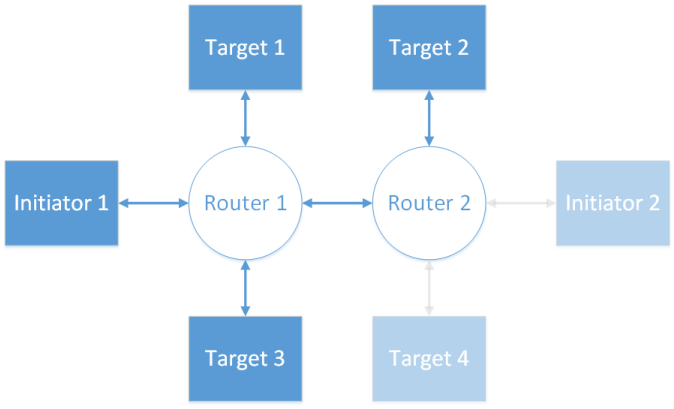

Fig. 1. Example of a virtual bus with three targets and five links

Due to the wormhole routing used by SpaceWire enabled routers, if there are multiple data-flows in a SpaceWire network there is a possibility of a packet being blocked if one of the SpaceWire links it requires is already in use. There may be more than one initiator operating in a SpaceWire-D network at the same time, so a set of initiator schedules is required to constrain traffic such that no two virtual buses are active in the same slot if there is a chance that they could have a colliding transaction i.e. if they have any shared links.

II. SpaceWire-D

The following subsections briefly describe the features of SpaceWire-D. For more in-depth coverage, see the standard draft [2] and [3].

A. Time-Division Multiplexing

In a SpaceWire-D network, the end of the current time-slot and the beginning of the next time-slot is signaled by the arrival of the next valid time-code. SpaceWire time-codes contain a 6-bit time value, so there are 64 slots in a SpaceWire-D schedule beginning at slot 0 and ending at slot 63. Additionally, a local timer can be used to synchronise with the arriving time-codes to provide redundancy in case a time-code fails to arrive.

Each time-slot can be assigned a single virtual bus. However, this is not a symmetric relationship because depending on the type of virtual bus, a bus may be assigned to multiple time-slots or adjacent sequences of time-slots called multi-slots, as described in the following sections.

When a new time-slot begins, if there is a virtual bus assigned to the time-slot, the group of transactions associated with the virtual bus is executed.

B. Static Bus

The SpaceWire-D protocol provides services to open, load, execute, and close four different types of virtual buses. The first and simplest virtual bus is the static bus.

Each static bus is assigned to a single time-slot or single multi-slot. Once opened, the user application can then load the static bus with a group of RMAP transactions. During the loading operation, the transaction group's worst-case execution time (WCET) is checked before the transaction group is accepted into the static bus. If the WCET of the transaction

group exceeds the duration of the time-slot or multi-slot it may interfere with the next slot's transactions, so the transaction group is not loaded and an erroneous response is sent to the user application.

A transaction group can be loaded as a repeating group in which case it is repeated every time the bus's time slot occurs until the bus is reloaded or closed, or as a single shot group where the transaction group is unloaded after a single execution.

C. Dynamic Bus

A dynamic bus can be assigned to multiple time-slots or multi-slots. When a transaction group is loaded, its WCET is checked, like the static bus, before it is accepted. If a transaction group is accepted and loaded into a dynamic bus, it is executed in the next time-slot or multi-slot assigned to the bus. This results in less predictability than a static bus because a transaction group could be executed in one of multiple time-slots.

D. Asynchronous Bus

As with a dynamic bus, an asynchronous bus can be assigned to multiple time-slots or multi-slots.

However, unlike the static bus and dynamic bus, which are based around loading groups of transactions, the asynchronous bus works on a single transaction basis. When a user application loads an asynchronous bus, it sends a data structure describing a single transaction along with the transaction's priority. The asynchronous bus maintains a prioritised queue of transactions, and in each available time-slot or multi-slot assigned to the bus, a subset of the highest priority transactions is removed from the queue and executed. The subset of transactions to be executed in the next available time-slot or multi-slot is updated whenever the user application loads a new transaction.

E. Packet Bus

The packet bus is a bi-directional channel between an initiator node and a target node. Receiving packets from and sending packets to a target are controlled by the initiator via RMAP read and write operations, respectively.

An initiator node can open multiple channels to targets and a target can open multiple channels to initiators. When the channel has been opened on both the initiator and target side, the packet bus is ready to handle RMAP transactions between the two nodes.

When a packet bus's time-slot or multi-slot begins, the status of all channels is checked to make sure a channel is not busy before it is used by the packet bus. This allows multiple initiators to open a channel to the same target and reserve it for exclusive use.

Optionally, the packet bus can use segmentation to split the transmission or receiving of a large packet over multiple time-slots or multi-slots.

F. Schedules

The source of unpredictability in a SpaceWire network is the possibility of packets being blocked by wormhole routing. Wormhole routing enables a packet to be switched from an input port to an output port quickly, but only if the output port is not already in use. If it is in use, the packet is blocked until the output port is released.

In order for traffic in a SpaceWire-D network to be deterministic, the possibility of blocking must be removed. This is done by ensuring that in each time-slot, the set of links used by an initiator's virtual bus is distinct from the set used by every other initiator's virtual bus operating within the same time-slot. If no link is used by two buses at the same time, then the blocking of SpaceWire packets cannot occur. This means that for each initiator, a schedule must be created that simultaneously satisfies this constraint and meets the bandwidth demands of a mission.

Research into the configuration of schedules for SpaceWire-D networks is ongoing at the University of Dundee and elsewhere [5] [6].

Figure 2 shows an example schedule for a single initiator with 64 time-slots and a combination of different virtual bus types [3].

Time-Slot	Bus
0	Static 0
1	Dynamic 1
2	Static 2
3	Async 3
4	Static 4
5	Async 5
6	Async 5
7	Dynamic 7
8	Empty
9	Dynamic 1
10	Dynamic 7
11	Packet 11
12	Packet 11
13	Packet 11
14	Packet 11
…	
61	Static 61
62	Dynamic 7
63	Static 63

Fig. 2. Schedule for a single initiator with 64 time-slots [3]

III. AT6981 CASTOR SYSTEM-ON-CHIP

The Atmel AT6981 Castor system-on-chip [7] is a LEON2-FT (SPARC V8 ISA) based flight processor with multiple integrated peripherals including extensive SpaceWire support.

Figure 3 shows a photo of the cPCI variant of the prototype AT6981 board, with three SpaceWire cables connected to the router

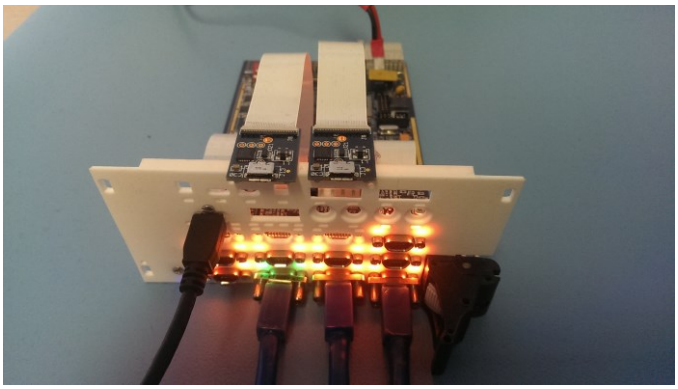

Fig. 3. cPCI variant of the prototype AT6981 board

The following subsections briefly describe the relevant SpaceWire peripherals, and features in the prototype board used for this research.

A. SpaceWire Router

The SpaceWire front-end for the AT6981 board is a SpaceWire router with eight external ports and three internal ports connected to the SpaceWire engines. The internal ports have physical addresses 9, 10, and 11 which connect to SpaceWire engines 1, 2, and 3 respectively.

B. SpaceWire Engines

Connected to the SpaceWire router, the three SpaceWire engines each contain three DMA channels, an RMAP initiator, and an RMAP target. The SpaceWire-D tests described in this paper use only the RMAP functionality in the engines.

In order to allow a SpaceWire packet to address individual DMA channels, RMAP initiator, or RMAP target, the SpaceWire engines use a de-multiplexer. The de-multiplexer matches up to four bytes of the incoming packet against a pattern and mask configured by the user in the engine's registers. It then uses this matching to filter the packet into the correct DMA channel, RMAP initiator, or RMAP target. This allows the RMAP initiator and target to have their own logical addresses.

The execution of RMAP commands are offloaded to the SpaceWire engines, reducing the demands on the LEON2-FT processor. The user application holds a list of data structures in memory describing the required RMAP commands and then writes the memory address of the list to the RMAP initiator's registers. Consequently, if the list of commands is unchanging over time as in the case of a static bus with a repeating transaction group, the processing required to begin executing the transactions is minimal.

C. Memory and Processor

The prototype AT6981 board has 128Kbyte of SRAM and 256MByte of DRAM and the LEON2-FT processor clock rate is 33MHz, while the production board will run at 200MHz.

D. Debug Support Unit

Loading and debugging a program is done via the LEON2-FT debug support unit (DSU). The DSU provides a simple protocol to read and write to memory on the board directly through hardware. This allows software running on the development machine to load a program directly into the AT6981's memory without the requirement of a bootloader. To debug a program, a STAR-Dundee software module on the development machine acts as a GDB remote protocol server and translates GDB commands into interactions with the DSU, allowing a simple method for debugging. The AT6981 prototype board provides a USB to UART bridge for connecting a computer to the DSU.

IV. RTEMS SUPPORT

The tests described in this paper use version 4.10.2 of the RTEMS real-time operating system, which is an open-source project being used in many space applications as well as in other industries [8]. The following subsections describe our use of RTEMS and its relevant features.

A. Board Support Package

A board support package (BSP) was designed to port RTEMS to the AT6981 board [9]. The basic BSP consists of the minimum requirements to run the basic RTEMS tests and examples. This includes the board initialisation code, a UART console driver, a clock driver, and support files such as a linker script file. There exists a BSP for an existing LEON2 device in the RTEMS source tree, however, the AT6981 is sufficiently different that it requires a separate BSP.

The AT6981 BSP uses the LEON2-FT's on-chip UARTs and timers with slightly modified drivers from the existing LEON2 BSP. Like the DSU UART, the LEON2-FT on-chip UARTs are accessible through USB to UART bridges on the prototype board.

As the AT6981 shares an interrupt between SpaceWire DMA and RMAP engines in the primary interrupt controller, the interrupt handling has been extended to allow an interrupt service routine (ISR) to be registered for either DMA or RMAP interrupts. When an interrupt is raised on the primary controller, the interrupt handler then filters it to the relevant ISR. This allows for separate device drivers for DMA and RMAP engines.

B. RTEMS Features

RTEMS is a real-time multi-task operating system with a unified address space. It provides features common in most operating systems such as tasks, interrupt handling, inter-process communication, synchronisation, standard data structures, and a device driver framework. RTEMS also provides in-depth compile time customisation.

A real-time operating system is designed to value predictability above other features [10]. As RTEMS is a real-time operating system, it provides task scheduling algorithms relevant to a real-time environment. In our case, we are using the default priority based pre-emptive scheduler which will

switch context to a higher priority task if one becomes available at any time.

V. Networking Software

RTEMS based networking software is responsible for providing the SpaceWire-D API to the user application, managing the virtual buses, managing the transition between time-slots, and dispatching RMAP commands.

The following subsections describe the different modules of the SpaceWire-D test software.

A. SpaceWire-D API

The SpaceWire-D API provides a public interface to the user application and enables an application to initialise the other SpaceWire-D modules, open a virtual bus, load a virtual bus, and close a virtual bus. During initialisation, the API creates tasks for the other modules as well as a task for itself and uses the RTEMS message queue manager in order to listen for requests from user applications. These requests are then handled by the virtual bus manager.

B. Virtual Bus Manager

All functionality related to the opening, loading, and closing of virtual buses is controlled by the virtual bus manager. It also contains the data structures describing the parameters of a virtual bus and its transactions.

C. Time Manager

The time manager is responsible for transitioning between time-slots, based on the arrival of valid time-codes. In this version, we are using only time-codes to signal the beginning and end of time-slots. However, the standard also describes the use of local timers to synchronise with the arrival of time-codes for redundancy in case a time-code fails to arrive.

During initialisation of the time manager, we enable interrupts for the receiving of time-codes using a simple device driver for the AT6981 SpaceWire router. We then install a callback function which is called during the router driver's ISR. The callback function uses the RTEMS event manager to send an event to the transaction dispatcher, signalling the start of the next time-slot.

D. Transaction Dispatcher

When the SpaceWire-D API initialises the other modules, a task is created for the transaction dispatcher. This task begins and then blocks, waiting for an event to be received from the time manager. The task wakes up when the event is received and, if there is a virtual bus assigned to the time-slot, executes the virtual bus. For example, if there is a static bus assigned to the time-slot, the bus's transaction group will be executed, assuming one is loaded.

A simple RMAP driver was designed to provide three features: the first is a function to start a group of RMAP transactions, the second is an ISR to handle RMAP initiator interrupts, and the third is a function to initialise one of the AT6981's RMAP targets to act as a target node for the purpose of the experiments.

VI. Experimental Setup

For our experiments we used two different network architectures. The following subsections describe and illustrate both architectures, and the additional supporting hardware and software used.

A. Single Initiator Architecture

The single initiator architecture as shown in Figure 4, uses a single AT6981 board as an RMAP initiator and RMAP target. The AT6981 is connected to a development machine for loading and debugging programs. The board's router loops back to itself with a STAR-Dundee SpaceWire Link Analyser Mk2 in the middle, to view the transactions and time-codes flowing through the links. A STAR-Dundee SpaceWire-USB Brick Mk2 is connected to the AT6981 and acts as the time-code master. The Link Analyser Mk2 and the Brick Mk2 are connected to a second laptop for ease of use. The Brick's time-code generation is controlled via STAR-Dundee's STAR-System software [11]. In this architecture, all SpaceWire links are operating at 100Mbps.

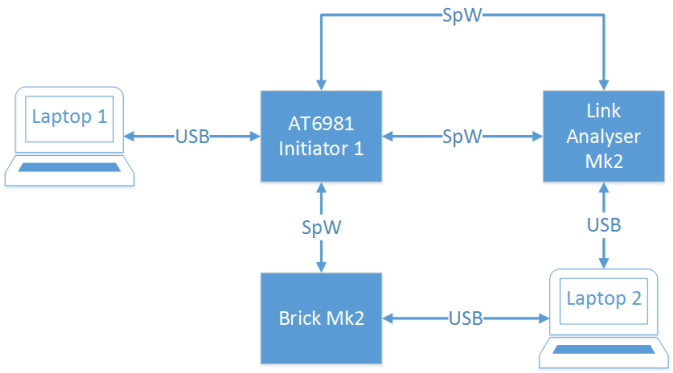

Fig. 4. Single initiator architecture

VII. Multiple Initiator Architecture

The multiple initiator architecture shown in Figure 5 is similar to the single initiator architecture shown in Figure 4 with the exception that the loopback through the Link Analyser is removed and replaced by a link between both AT6981 boards, again through a Link Analyser. In this architecture, the first AT6981 board's SpaceWire links are operating at 100Mbps and the second board's links are running at 50Mbps.

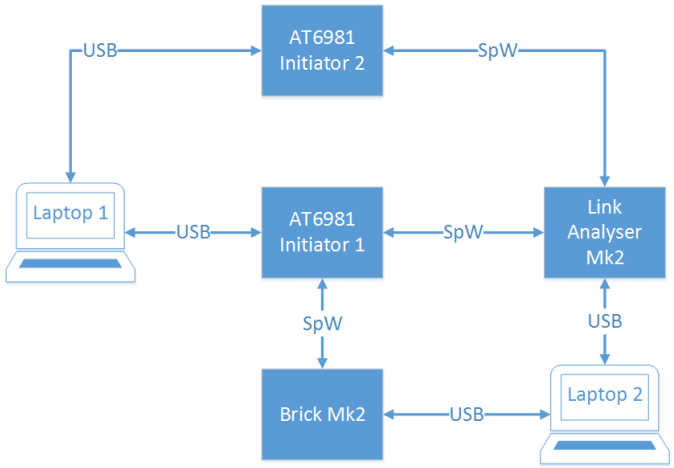

Fig. 5. Multiple initiator architecture

Figure 6 shows a photo of the hardware used in the multiple initiator architecture setup. From left to right, the hardware is the first AT6981 prototype board, a STAR-Dundee SpaceWire Link Analyser Mk2, the second AT6981 prototype board, and a STAR-Dundee SpaceWire-USB Brick Mk2

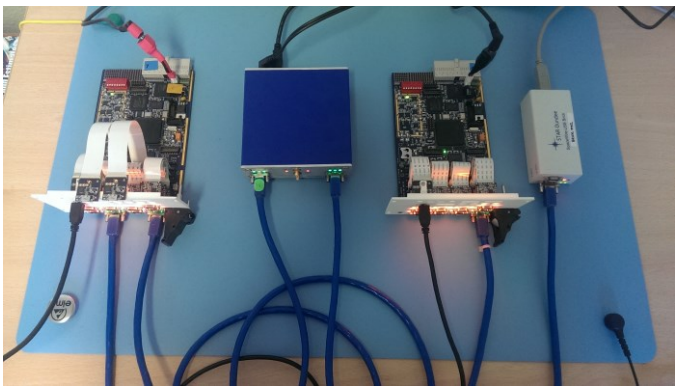

Fig. 6. Photo of the multiple initiator architecture

VIII. Experimental Results

The following subsections describe the experiments carried out to test the SpaceWire-D static bus with both single and multiple initiator architectures, and presents the results obtained.

A. RMAP Driver

The first iteration of the RMAP driver used by the transaction dispatcher utilised the UNIX-like device file driver framework provided by RTEMS. Within this framework, every device is treated as a file and a driver provides initialise, open, close, read, write, and ioctl functions to be used with standard system calls.

After performing some initial tests and measuring the performance of the driver, it was found that the overhead required by opening a device file and using an ioctl system call when dispatching a transaction group was too expensive. By allowing the transaction dispatcher to call the driver functions directly instead of through the ioctl system call interface, the processing time between a time-code being received and the first RMAP transaction leaving the router was reduced from 741µs to 389µs.

Further optimisation was introduced by simplifying the event handling when a time-code is received. Originally, the time manager would receive an event from the router ISR, then forward the event to the transaction dispatcher. By sending the event directly from the time manager's callback function, the processing time was further reduced from 389µs to 201µs.

Reducing the initiator processing time between a time-code being received and the first RMAP transaction leaving the router from 741µs to 201µs allows the SpaceWire-D network to run at the minimum slot duration of 1ms. With the production version of the AT6981 running at 200MHz, compared to the prototype's 33MHz, and with additional software optimisations, the initiator processing time should be further reduced.

B. Single Initiator Experiments

In the single initiator architecture, the AT6981 board acts as both the RMAP initiator and the RMAP target. During the test setup, the SpaceWire-D API is initialised, the RMAP target is initialised, and the test static buses are opened and loaded with a transaction group.

The first experiment involved opening a single static bus in slot 0 and loading it with a transaction group containing 32 RMAP write-with-reply commands with a data size of 1KB. The Brick is generating time-codes every 10ms.

Figure 7 shows a screenshot of the Link Analyser status counter display interface for the first experiment. In this interface, the number of various types of characters received per second are displayed. The first column is the Link Analyser port that the RMAP headers are transmitting through, and the second column is the RMAP replies. We can see that there are 32 RMAP transactions being executed by viewing the number of EOP characters and confirming that the commands were executed successfully by viewing the packet display interface within the Link Analyser software. The number of data characters being transmitted per second can be verified by calculating the size of the RMAP headers and replies. For the header, there is 1 physical address at the head of the packet, a 21 byte RMAP header, and 1024 bytes of data. This results in 1046 data characters multiplied by 32 which is 33472 data characters per second. The reply has 1 physical address at the head of the packet, and an 8 byte reply, which results in 9 data characters multiplied by 32, giving 288 data characters per second.

Data Character	33,472	288
EOP Character	32	32
EEP Character	0	0
FCT Character	40	4,188
NULL Character	12,486,170	12,525,577
Time-code Character	100	100

Fig. 7. Link Analyser output for a single slot schedule

Next, we opened a static bus on all 64 slots and loaded them with the same transaction group as the previous experiment.

Figure 8 shows the results from opening a static bus on all 64 slots. Again, the number of data characters transmitted can be verified by multiplying 1046 data characters by 3200 in this case, which is 3,347,200 data characters per second. Similarly the data characters per second for the replies is calculated by multiplying 9 data characters by 3200 which is 28800.

Data Character	3,347,200	28,800
EOP Character	3,200	3,200
EEP Character	0	0
FCT Character	4,000	418,800
NULL Character	8,340,452	12,281,052
Time-code Character	100	100

Fig. 8. Link Analyser output for a 64 slot schedule

In both cases of the single slot and the 64 slot schedule, the observed WCET of the transaction group is 4182µs. The observed worst-case processing time of 201µs can be added to this to give a total static bus execution time of 4383µs.

C. Multiple Initiator Experiments

In the multiple initiator architecture, there are two AT6981 both acting as initiators and as targets for each other. The schedule in this experiment is split between the two boards. In all of the even numbered time-slots, the first board opens a static bus. The second board opens a static bus in all of the odd numbered time-slots. Each static bus is loaded with the same transaction group as the single initiator experiments, 32 RMAP write-with-reply transactions with a data size of 1KB.

Figure 9 shows a screenshot of the Link Analyser status counter display for the multiple initiator architecture experiment. In this case, both RMAP headers and RMAP replies are travelling bidirectionally through both links. To verify the number of data characters being transmitted per second for each side of the Link Analyser, we can add the data characters for both the RMAP headers and the replies. In this case, there are 1600 transactions being executed every second by each initiator. This results in 1600 multiplied by 1046 data characters for the RMAP headers, which is 1,673,600 data characters per second. For the RMAP replies, there is 1600 multiplied by 9 data characters, which is 14400 data characters per second. Summing the two gives 1,688,000 as supported by the screenshot.

Data Character	1,688,000	1,688,000
EOP Character	3,200	3,200
EEP Character	0	0
FCT Character	211,400	211,400
NULL Character	10,310,758	4,068,663
Time-code Character	100	0

Fig. 9. Link Analyser output for the multiple initiator architecture

IX. Future Work

The experiments described in this paper were focused on parts of the static bus of SpaceWire-D networks. Further work is required to test the remaining features of the static bus such as transaction group execution time calculation and multi-slot buses. Additionally, the remaining virtual bus types: the dynamic bus, the asynchronous bus, and the packet bus require similar experimentation and testing. Future research will be carried out to fulfil these goals.

As mentioned in Section 2, the schedulability of SpaceWire-D networks is an important problem. Research is being carried out to investigate scheduling methods for the latest draft of the standard.

X. Conclusions

This paper has briefly described the latest version of SpaceWire-D [3] and presented the results from experiments using the AT6981 [7] prototype board, an RTEMS port for the AT6981 [9], and RTEMS based networking software to test the static bus functionality of SpaceWire-D.

The results show that the AT6981 prototype board can be used to operate a SpaceWire-D network using the static bus with schedules utilising single slots and all 64 slots. An experiment was successfully carried out to test a SpaceWire-D network with two AT6981 boards acting as RMAP initiators operating in alternating time-slots.

Acknowledgments

The research leading to these results has received funding from the European Space Agency under ESA contract number 4000107346/12/NL/LvH/fe. We would also like to thank David Jameux the ESA project manager for the SpaceWire-D related activity.

References

[1] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire – Links, nodes, routers and networks”, Issue 2, European Cooperation for Space Standardization, 31 July 2008, available from http://www.ecss.nl

[2] S. Parkes, A. Ferrer Florit, A. Gonzalez Villafranca, C. McClements, “SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks”, Draft C, April 2012, available from http://spacewire.esa.int/WG/SpaceWire

[3] S. Parkes, D. Gibson, A. Ferrer, “SpaceWire-D: Deterministic Data Delivery over SpaceWire”, Data Systems in Aerospace, Warsaw, Poland, June 2014

[4] ECSS Standard ECSS-E-ST-50-52C, “SpaceWire - Remote memory access protocol”, Issue 1, European Cooperation for Space Standardization, 5 February 2010, available from http://www.ecss.nl

[5] D. Raszhivin, Y. Sheynin, A. Abramov, “Deterministic Scheduling of SpaceWire Data Streams”, International SpaceWire Conference, Gothenburg, Sweden, June 2013

[6] Y. Chen, M. Takada, R. Kurachi, H. Takada, “A Scheduling Method of RMAP Packets for SpaceWire-D”, International SpaceWire Conference, Gothenburg, Sweden, June 2013

[7] S. Parkes, C. McClements, G. Mantelet, N. Ganry, “The Next Generation of Spaceflight Processors: Low Power, High Performance, with Integrated SpaceWire Router and Protocol Engines”, International Astronautical Congress, Beijing, China, September 2013

[8] RTEMS, “RTEMS Applications”, Available at: http://rtems.org/wiki/index.php/RTEMSApplications

[9] D. Paterson, D. Gibson, S. Parkes, “An RTEMS Port for the AT6981 SpaceWire-Enabled Processor: Features and Performance”, International SpaceWire Conference, Athens, Greece, September 2014

[10] J. Stankovic, R. Rajkumar, “Real-Time Operating Systems”, Real-Time Systems, vol 28.2-3, pp. 237-253, 2004

[11] S. Mills, A. Mason, C. McClements, D. Paterson, I. Martin, S. Parkes, “Developing SpaceWire Devices with STAR-Dundee Test and Development Equipment”, International SpaceWire Conference, Gothenburg, Sweden, June 2013

SpaceFibre (Short)

An Experimental Evaluation of SpaceFibre Resource Requirements

Session: SpaceFibre, Short Paper

Matthew Rowlings, Martin Suess
ESTEC, European Space Agency
Noordwijk, The Netherlands
mr589@york.ac.uk
martin.suess@esa.int

***Abstract*— SpaceFibre is the next generation of high speed interconnects for spacecraft communication networks. However the high performance and advanced features mean that SpaceFibre interfaces have an inherently high level of complexity, rendering implementation in some FPGA systems difficult due to limited hardware resources. This paper presents an investigation into the hardware resource characteristics of SpaceFibre through an experimental evaluation using the StarDundee IP core, including discussion of functionality tradeoffs that can be made when designing an interface for a limited number of hardware resources. A simple SpaceFibre interface optimised for a high-throughput instrument is subsequently described, with a specific focus on resource savings achieved when implementing for the RTAX2000 FPGA.**

***Index Terms*— SpaceFibre, Networking, FPGA, Spacecraft Electronics.**

I. Introduction

SpaceFibre [1] is a multi-gigabit spacecraft network standard supporting the future of high performance spacecraft communication network requirements. It improves on SpaceWire by offering data rates of up to 5Gb/s, several channel interfaces per link via a Virtual Channel (VC) service, support of mixed mode networks through Quality of Service (QoS) mechanisms and error free data reception through the use of a retry scheme.

Naturally these additional features result in an interface that is significantly more complex than a standard SpaceWire interface. As the SpaceFibre development matures this will eventually result in standalone SpaceFibre interface products implemented as an *Application Specific Integrated Circuit* (ASIC) (e.g. [2]), where this increased complexity is not an issue due to the high density of logic elements available in an ASIC technology. In the meantime, and always available as an additional option for system designers, the interface can also be integrated as part of a *System on Chip* (SoC) within *Field Programmable Gate Array* (FPGA) technology. Such implementations are typically designed for a radiation hardened FPGA, coupled with a space qualified Serialiser/Deserialiser. An issue with this approach is that currently available FPGAs suitable for spaceflight (for example the RTAX-S family [3]) lack the high number of logic elements required to implement the full interface and ensure that a sufficient amount of FPGA resources are still available for the implementation of other elements of the data handling SoC.

This paper describes a series of tasks undertaken with the SpaceFibre IP core developed by StarDundee to gain understanding of how features of the SpaceFibre standard affect FPGA resource requirements, followed by an experimental exploration into how the standard can be implemented to minimise resource requirements.

An investigation architecture based on Microblaze and implemented on a Virtex-6 FPGA is used to undertake analysis of the FPGA resource usage and performance of the IP core. Experimental data on how key parameters of the IP (e.g. number of virtual channels) relate to resource requirements is presented. These findings are then used to guide some experimental modifications to the IP core that explore reducing functionality of the interface with the prospect of achieving a considerable reduction in the amount of hardware resources required.

Finally an implementation of the SpaceFibre standard is presented that uses the previous findings to produce a minimal adaption optimised to the typical interface requirements of a high data rate spacecraft instrument. This results in an specialised interface that is better suited for implementation within the current generation of flight FPGAs, whilst still maintaining compatibility with standard SpaceFibre interfaces.

II. Hardware Resources Evaluation

To facilitate investigation into the hardware resource requirements of the SpaceFibre interface, a test architecture based on Microblaze and implemented on a Virtex-6 FPGA was created as detailed in Fig 1. This system provides versatile manipulation of the Management and Virtual Channel interfaces of the SpaceFibre IP, allowing a range of test parameters to be set on the Management interface and full loading of the virtual channels at a net data throughput of 1.94 Gbps from a 2.5Gpbs line rate with 8b/10b encoding. The IP core also provides several hardware parameters that allow the core to be customised; the number of virtual channels and the size of the retry buffers are of particular interest to resource requirements.

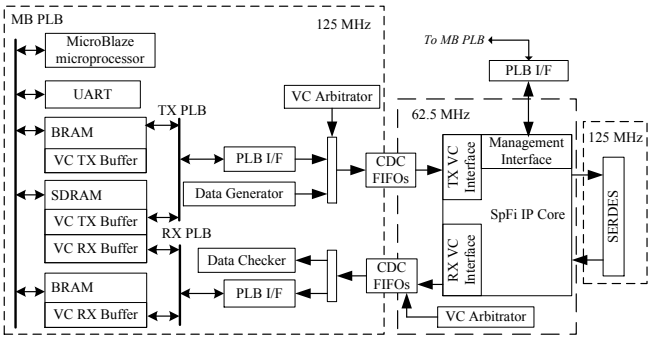

Figure 1 Investigation Test Architecture

Several designs were implemented to explore the range of each hardware parameter in turn, with the resource requirements for the SpaceFibre IP taken from the Xilinx mapping report.

A. Number of Virtual Channels

Each virtual channel provides a "FIFO-style" external application interface, requiring each virtual channel to include a transmit buffer and a receive buffer. The QoS is responsible for multiplexing the transmit buffers together according to the QoS scheme configured. Therefore we expect each virtual channel added to the interface to require two memory elements and an increasing amount of logic to integrate with the QoS. Fig 2 illustrates the resource requirements for interfaces with 2, 4, 8 and 16 virtual channel interfaces.

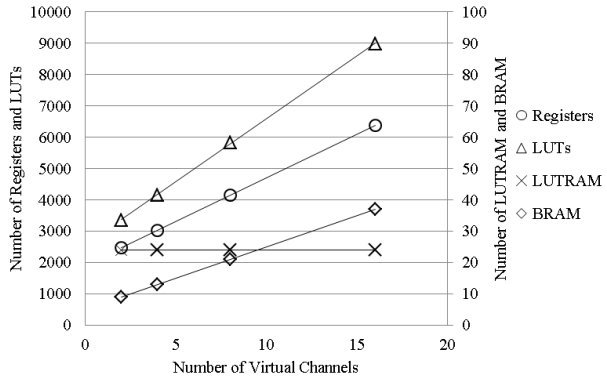

Figure 2 Scaling of FPGA resource with number of Virtual Channels

From this graph we see the expected increase in the number of buffers, implemented as Block RAMs (BRAMs). As the number of virtual channels is increased from 2 to 16, the number of BRAMs required also increases by a difference of 28 from 9 to 37. This is of note as it confirms that each virtual channel will require two distinct memory elements, with the observation that the size of the buffer may as well be set to the maximum size of the BRAM element as any size smaller will result in wasted memory resource that cannot be used in any other buffer.

The sharp increase in registers and Look Up Tables (LUTs) required is due to the QoS logic required for each virtual channel; this includes not only the monitoring of QoS parameters for each VC (such as the bandwidth used), but also the large interconnect between all of the virtual channels and the logic that uses the QoS parameters to select which virtual channel to transmit from. Thus an effective minimisation is to ensure that the number of virtual channels is optimised to the needs of the application.

B. Size of Retry Buffers

There are three retry buffers on the transmit side of SpaceFibre which are necessary to facilitate prioritised retry of broadcast frames, Flow Control Tokens (FCTs) and data frames that all could be corrupted during transmission. Each primitive has a different unit size: broadcast frames are four words, FCTs are a single word and a data frame is up to 66 words (including the Start of Frame and End of Frame control words: SDF and EDF). Therefore there is a lot of scope for individual retry buffer size optimisation. However, as with the virtual channel buffering, a fundamental issue is that if a memory element is used then the minimum size is bounded by the size of the embedded memory element on the implementation device. This is illustrated by Fig 3, whereby an increase in virtual channel retry buffer size does not increase resource requirements until the buffer is larger than one Block RAM element.

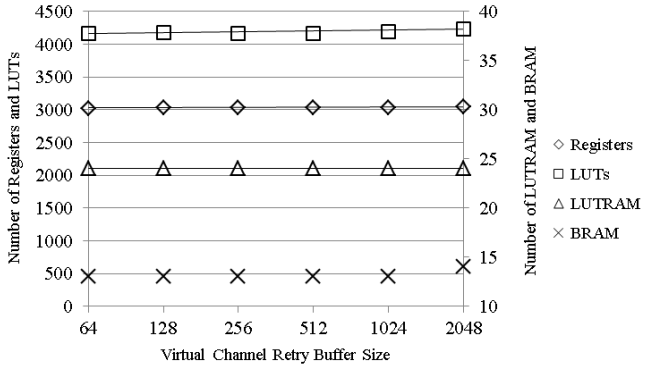

Figure 3 Scaling of FPGA resources with size of VC Retry Buffer

The characteristics for the broadcast and FCT buffers are the same, but they can benefit from their small unit size. If only a small broadcast buffer is used, say two broadcast frames (8 words) as the user application rarely sends broadcasts, then this can be implemented into Look-Up Table RAM (LUTRAM) instead of requiring a full memory element. A similar optimisation exists for the FCT buffer, in a system with a small amount of virtual channels it is unlikely that more than a few FCTs will be waiting for acknowledgement at any one time (aside from directly after link start-up).

III. Experimental Modifications

From the evaluation above we can see that there are several optimisations that can be made by simply modifying the hardware parameters of the SpaceFibre IP. To further this work, it was considered that modification of the IP itself could gain more significant resource reductions by reducing the performance of the interface. The following modifications were explored:

A. Heterogeneous VC Buffer Sizes
B. Single Virtual Channel Interface
C. Unbuffered Retry Scheme

All modified interfaces were compared to the resource requirements of the IP core set to the minimal hardware configuration suggested by StarDundee [4]; this minimal interface was also used as base implementation for each modification.

A. Heterogeneous VC Buffer Size

The SpaceFibre IP has the restriction that all virtual channel buffers are set to the same size, removing the opportunity of optimising the buffer size to the nature of the traffic using the virtual channel. This would typically not be an issue as if an embedded memory element has to be used for the buffer then the whole memory element may as well be used. However if we consider the case where a virtual channel is used exclusively for small and infrequent control packets, then a small virtual channel buffer would be more optimal and could be implemented as a small LUTRAM instead of using embedded memory resources (where most of the embedded memory block would be wasted anyway as only a small buffer is required).

This proposition was examined by modifying one of the virtual channels to use a sub-frame size of only 8 words for its transmit virtual channel buffer (the sub-frame size is allowed on the transmit side if each small packet is terminated with an EOP within the 8 words). Unfortunately as each FCT represents a frame of data (64 words), the receive buffers are bounded to this size. Regardless, this modification resulted in a saving of a Block RAM resource over the reference minimal interface at the expense of a 37% increase in LUTRAM required (from 64 LUTRAMs to 88).

B. Single Virtual Channel

As we saw with the scaling of the number of virtual channels in the interface, the logic for the QoS functionality is inherently complex. Therefore if a SpaceFibre interface was reduced to a *single* virtual channel then we would not require this complexity, but we would keep the flow control and high priority broadcast capabilities; such an interface could be suitable for high data-rate instruments that only require a single high speed link to a processing unit or mass memory. The interface was fixed to a single virtual channel and the QoS and multiplexing was removed. This resulted in savings of 16% in sequential logic, 33% in combinatorial logic and of course the two Block RAMs that the second virtual channel required previously.

C. Unbuffered Retry Scheme

Another source of buffering in the interface is within the Retry layer. Three seperate buffers exist to store data frames, broadcast frames and FCTs when transmitted until an acknowledgement token (ACK) is received from the remote side. If instead a negative acknowledgement token (NACK) is received then the data within these retry buffers is re-sent. In the case where the application can handle some loss of data frames, then this retry buffer is not strictly required. Indeed if a data source is generating data at a similar rate to the line rate of the interface, then some data loss is unavoidable while the retry is being undertaken unless sufficient buffering is provided upstream to handle the pause in transmission.

Instead of resending the frames on receipt of a NACK, the unbuffered retry scheme sends an Error End of Packet (EEP) and spills the data from that virtual channel until an EOP is encountered. This eliminates the need for the virtual channel retry buffer without sacrificing compatibility with a remote interface that may implement the whole codec.

Whilst this modification did remove the virtual channel retry buffer and its associated Block RAM, a small buffer was also required to keep track of which virtual channels had been waiting for acknowledgement when the NACK was received and so which should be spilling. Therefore we see a 3% increase in the amount of sequential elements required, however the simpler retry mechanism does succeed in reducing the number of the combinatorial elements by 14%.

IV. A Minimal SpaceFibre Interface for Instruments

A. Interface Functionality Requirements

The results from the experiments above show that we can modify some parts of the interface to save hardware resources, but simply modifying resource heavy parts of the interface does not drastically reduce enough resource requirements to warrant the reduce in functionality. Therefore a more substantial modification to the investigated interface implementation will be required to produce an even smaller interface. If we consider a minimal interface required for a high data-rate instrument, the functional requirements could be fulfilled using only two virtual channels: one for the high throughput instrument data and a second, much smaller buffer for transmitting housekeeping data and for the instrument commanding. With such a virtual channel partition it can also be seen that only a single channel would be sufficient on the receive side for receiving command packets, eliminating the need for the receive buffer on the high throughput instrument data virtual channel.

Some degree of QoS is required within this interface to ensure that command and housekeeping channel data can be transmitted even if the high throughput channel is saturating the link. We saw previously that the full QoS functionality requires a large amount of resources and so a much simpler but less flexible scheme could be used to cover this requirement, especially as housekeeping data typically only requires a small proportion of the link bandwidth. A similar issue applies to the retry layer; supporting lossless transmission is desired as the application is unknown, but the buffering and control logic requires a considerable amount of hardware resource.

A caveat of any substantial modification of these layers is that they still have to be fully compatible with the full SpaceFibre standard as the remote node has to be assumed to implement a fully featured interface. Most of the modifications discussed involve the higher levels of the transmit side and so we can keep as much as of the lower levels of the interface

unmodified as possible to ensure link-level compatibility with other interface implementations. The receive side of the interface shall also stay largely intact to ensure compatibility, the only modification shall be the reduction of the receive virtual channel level to a single virtual channel. As there are cases where there is no need for an instrument to transmit broadcast packets, the broadcast transmit logic and the associated retry buffer shall also be removed, although by keeping the receive side intact we do not sacrifice the ability to receive broadcast frames.

In summary, these restrictions lead to a transmit interface with the following specifications:

1. A high throughput, single ended data virtual channel.
2. A low throughput command virtual channel.
3. Simple QoS between these channels.
4. Retry scheme to resend data from these channels.
5. No broadcast transmit functionality.
6. FCT transmit and retry support as specified in the standard.

B. Architectural Design

From these specifications, a transmit interface was designed to perform the specified functionality for a minimal hardware resource requirement. A major architectural feature is the combination of the virtual channel and the retry buffers, allowing efficient storage of virtual channel retry data and optimal storage of frames as they are stored pre-framing, removing the need to store an SDF and EDF word for each frame. This results in a retry functionality distributed between the virtual channel and retry layers, requiring a slight modification to the dataflow through the interface as retried packets must also pass through the framing layer, as displayed in Fig 4.

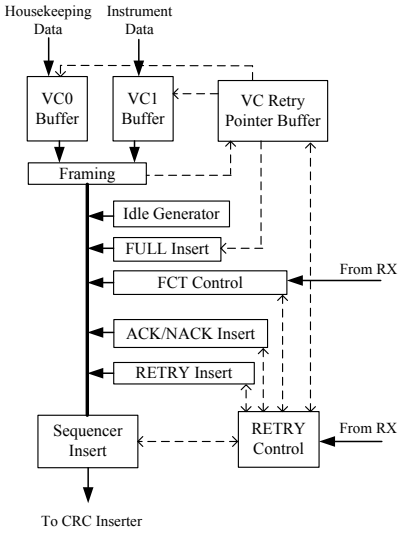

Figure 4 Dataflow of transmit side of instrument SpaceFibre interface

To keep the quality of service as simple as possible it is accepted that all data to be sent from the command and housekeeping virtual channel is of a higher priority than the high throughput instrument data, therefore whenever housekeeping packets are ready to be sent they are always selected over instrument data frames. In the case of the data virtual channel failing as a babbling idiot, this priority scheme ensures that data from the housekeeping channel will always dominate over this babbling idiot data.

Each virtual channel buffer now has three pointers to manage dataflow into and out of the channel: a write pointer, a read pointer and a retry pointer. The read and write pointers are used as a standard FIFO implemented in a circular buffer, the retry pointer however shows how much of the buffer is being used for storage of retry frames; as illustrated in Fig 5. The retry pointer is set by a separate *retry pointer buffer*. This buffer is appended to by the framing layer each time a SDF is added at the start of a new frame with the start address of the packet. The buffer is popped every time an ACK is received by the difference between the ACK sequence number and the previous ACK'd sequence number, thus the retry pointer always points to the start address of the oldest frame that has not been ACK'd yet.

In the case of a NACK, the buffer is popped by the difference in NACK value as with the ACK case and then all stored addresses but the current value of the buffer are flushed, as they will be reinserted into the buffer by the framing layer during the retry. Now the read pointer is set to the address of the retry pointer (i.e. the start of the last frame not ACK'd) and transmission is performed as with a standard data frame. This has the side effect that contiguous small packets may be repacked into a single frame when they may have been originally sent as separate frames, which is why it is important that the retry pointer buffer is flushed. Data sequence within a virtual channel is preserved however, with priority given to retry operations on the housekeeping virtual channel (as with nominal operation).

Flow Control Tokens are also subject to retry requests, but to eliminate the need for a buffer only one at a time is sent, buffered and the sequence number stored. When an ACK is received covering this sequence number, the interface can now handle the next FCT request. In the case of a NACK that covers the FCT sequence number, the buffered FCT is re-sent and the stored sequence number updated. This is unlikely to cause performance limitations on the link as FCTs are typically sent periodically, and with only two virtual channels it is unlikely that multiple FCTs will be requested to be sent in a small time frame; the only time this happens is directly after link initialisation where such a performance drop in FCTs is assumed to be tolerated.

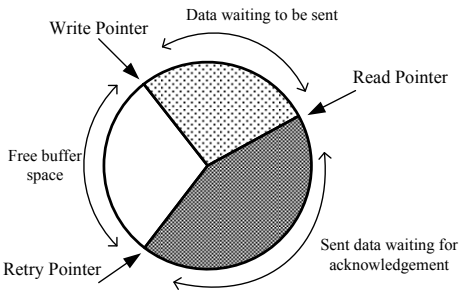

Figure 5 Combined virtual channel and retry buffer structure

C. Implementation Results

The proposed architecture was implemented and integrated into the SpaceFibre IP, replacing all the transmit circuitry up to the CRC inserter of the existing retry layer (the CRC inserter is the last step before the lane layer). The receive side was also restricted to only include a single virtual channel and the FCT arbitration removed. A number of functionality tests were also carried out, in simulation at first but then in hardware with the adapted IP interfaced to the StarFire test unit. This verified that the nominal and retry functionality of the minimal interface behaves correctly and that compatibility with the standard implementation is maintained.

As with the earlier modifications the investigation interface resource requirements are compared with the reference minimal implementation of the SpaceFibre IP core. Table 1 shows these results for the Virtex-6 implementation, where the required numbers of each logical element type are presented:

	Sequential Logic	*Combinatorial Logic*	*LUTRAM*	*Block RAM*
Reference Minimum	2409	2993	64	7
Investigation Minimum	1569	1749	12	5
Reduction	**35%**	**42%**	**81%**	**29%**

Table 1 Instrument interface resource requirements for Virtex-6

These results show that the lightweight optimisation of the interface was successful, both in terms of reducing the buffering required in the interface (shown by the reduction in Block RAM and LUTRAM) and in terms of simplifying more complex parts of the codec (represented by the reduction in sequential and combinatorial elements).

As the primary motive for this study was to propose an minimal interface suitable for implementation within a current space qualified FPGA, implementation for a Virtex-6 is not a realistic use case. Therefore the investigation interface and the minimal standard interface were also synthesised for the MicroSemi RTAX2000 radiation-tolerant FPGA. This FPGA has a significantly smaller number of logic resources available than the Virtex-6 [3] [5] and so it is crucial that the fraction of the overall FPGA resource dedicated to the high speed interface is minimised. Table 2 shows the raw logic resource usage and the percentage of the RTAX2000 total resources required for each interface implementation.

	Sequential Logic	*Combinatorial Logic*	*Embedded RAM*
Reference Minimum	2692	5156	10
RTAX2000 Resource %	25%	24%	15%
Investigation Minimum	1633	3052	6
RTAX2000 Resource %	15%	14%	9%
Reduction	**39%**	**41%**	**40%**

Table 2 Instrument interface resource requirements for RTAX-2000

Due to differences in the FPGA architectures, some variance between the resource requirements are to be expected. Being an antifuse based FPGA, the RTAX does not support small memories embedded within look-up tables, therefore all buffers are implemented in Embedded RAM blocks. This results in a significant saving of RAM blocks in the investigation architecture, with only six buffers required for the SpaceFibre interface. The sequential and combinatorial elements show a similar reduction to the Virtex-6 results, but when this reduction is compared to the total number of resources available in the device we see a very significant reduction in the required resources: from 25% to only 15% of the sequential logic elements and from 24% to 14% of the combinatorial logic elements available on the RTAX2000.

V. Conclusions and Future Work

The SpaceFibre standard contains a number of parameters, such as the number of virtual channels, that can be tailored in an implementation while still maintaining compatibility with other SpaceFibre implementations. This can be used to reduce to the implementation complexity of the SpaceFibre interface. This tailoring was further extended to demonstrate an interface architecture targeting a simple high-speed instrument requiring only one high throughput downstream data channel and one high priority channel for commanding and housekeeping data. The resulting SpaceFibre interface implementation uses only 15% of the RTAX2000 hardware resources.

There are still further optimisations that could be made to the interface however. These primarily concern the receive side of the interface. A buffer exists for holding data frames whilst the CRC is checked before being passed out of the retry layer. Closer integration with the receive virtual channel buffer could be performed here to remove the need for this buffer. Likewise a design could be undertaken to also merge the elastic buffer into the virtual channel buffer, but care must be taken when communicating with the transmit side of the retry layer as the receive side would then be clocked by the recovered clock. Alas the extent of any modifications to the CRC functionality or lane layer (other areas of high complexity) is very limited as these layers are fundamental for compatibility with remote interfaces.

It is important to acknowledge that the reference IP core from StarDundee was created to aid the development of the SpaceFibre standard, and is not as yet optimised to minimise hardware resource requirements.

References

[1] S. Parkes et al. “SpaceFibre Standard Draft F3”, University of Dundee, 10th September 2013.

[2] S. Parkes et al. “A Radiation Tolerant SpaceFibre Interface Device”, 5th International SpaceWire Conference, 2013

[3] Microsemi Corporation, “RTAX-S/SL and RTAX-DSP Radiation-Tolerant FPGAs”, January 2013

[4] STAR-Dundee Limited, “SpaceFibre VHDL IP Core User Manual”, 23rd November 2012

[5] Xilinx Inc. “Virtex-6 Family Overview”, 2012

Network Layer Support in SpaceFibre Protocol

SpaceFibre, Short Paper

Nadezhda Matveeva, Elena Suvorova
Saint-Petersburg State University of Aerospace Instrumentation
SUAI
Saint-Petersburg, Russian Federation
nadezhda.matveeva@guap.ru, suvorova@aanet.ru

***Abstract*—** *The SpaceFibre standard has appeared relatively recently. Also SpaceFibre standard supports several "Quality of Service" mechanisms. It includes best effort, bandwidth reserved, scheduled and priority based qualities of service. It is implemented by means of virtual channels. Standard does not fully describe network layer in the latest version (ECSS Draft F3). The rules for transferring data at the network layer also affect the quality of service.*

In this paper we present analysis and an implementation of the SpaceFibre network layer. The switch matrix's channels quantity connected to port (connection point) is one. Low priority packet transmission can be interrupted, if a packet with a higher priority is received. Interruption rules will be described, data transmission latency characteristics and performance will be evaluated. Analysis and modeling of the proposed network level implementation will be demonstrated. Data packets of different sizes were used during simulation. Number of virtual channels is 4 for the research.

***Index Terms*—** *SpaceFibre, Quality of service (QoS), Network level.*

I. INTRODUCTION

Performance of modern embedded systems depends on network architecture and structure. Existing embedded networks support data transmission with Quality of Service (QoS) [1]. Currently many different standards are widely used in design of network. For example – RapidIO [2], SpaceWire [3] and etc.

For our research we chose different approaches to implementation technology of virtual channels [4]. The first allows transferring data at the same time from different virtual channels of a port. The second – only one virtual channel of a port can transfer data. The third - virtual channel with higher priority can interrupt the transmission of data with lower priority. These approaches are not associated with a specific standard. It can be used in the construction of different embedded networking technologies [5].

We will use SpaceFibre in our case study. SpaceFibre is the modern standard in space industry. This technology also can be used for construction embedded networks.

SpaceFibre provides a coherent quality of service (QoS) mechanism able to support best effort, bandwidth reserved, scheduled and priority based qualities of service [6]. Quality of service parameters [7] that can be provided by routers with SpaceFibre ports depend not only on the SpaceFibre protocol characteristics and port specific implementation but also on a network layer implementation. In this article we analyze different implementation of network layer SpaceFibre.

II. DIFFERENT APPROACH OF NETWORK LEVEL IMPLEMENTATION

In this paper we will compare some approaches of network level implementation. They differ from each other in hardware costs and data transmission principles.

In 1st way of router's network layer structure switch matrix includes a separate channel for connection of each input virtual channel with the correspondent output virtual channel. Quantity of connection points to the switch matrix (hereinafter – connection points) for every port of a router is equal to the virtual channels number in this port, Fig. 1 (only one data transmission direction is represented). This way was recommended by the SpaceWire-RT specification draft [8]. In such router structure data flows can compete with each other only within one virtual channel in output port of router. In this case timing characteristics in the network layer depend only on arbitration rules. In all other cases timing characteristics of data flows are not influenced by the router network layer. However, such router structure results in an essential hardware cost.

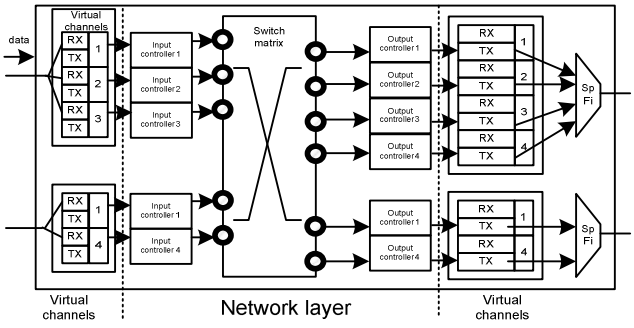

Fig. 1 The first way of router's network layer implementation

According to 2nd way of router's network layer structure, the quantity of connection points for every port is less than number of virtual channels in the port. There is one connection point. In our research we suppose that data flows from every virtual channel can be transmitted via one connection point of the correspondent port, Fig. 2. Hardware cost of this router

structure is essentially less, than hardware cost of the previous one. But in this way, data flows from different virtual channels share switch matrix channels. Therefore, an impact between data flows and corresponding disturbance of its timing characteristics in this case in this router structure is more essential than in the previous one.

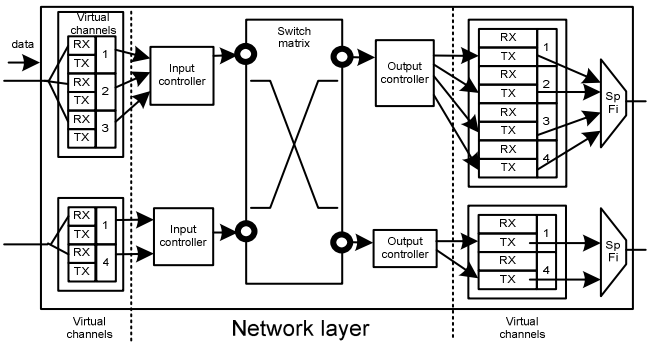

Fig. 2 The second way of router's network layer implementation

The 3rd way of router's network layer structure is similar to 2nd way. The difference between these ways is possibility of lower priority data transmission interruption. Condition of data transmission interruption can be different. Packet transmission can be interrupted after N byte transfer or special time interval after the start of transmission. The following situations can cause interruption of packet transmission: 1) frame with higher priority comes to virtual channel buffer or 2) no data is transmitted during K clock. K and N are software installed parameters. Output ports unavailability or empty virtual channel buffer in specified port can cause data transmission impossibility. When setting N or K parameter, it is necessary to avoid situations when too frequent lower priority data transmission interruption occurs. For example, at the beginning 1 byte of lower priority packet is transmitted, than higher priority packet is transmitted. Thereafter 1 byte of lower priority packet is transmitted and lower priority data transmission interruption occurs again. Interval of data transmission interruption should be comparable to time of channel reset.

III. Network model

Timing characteristics estimation was done on the basis of the models, which are depicted in Fig. 3.

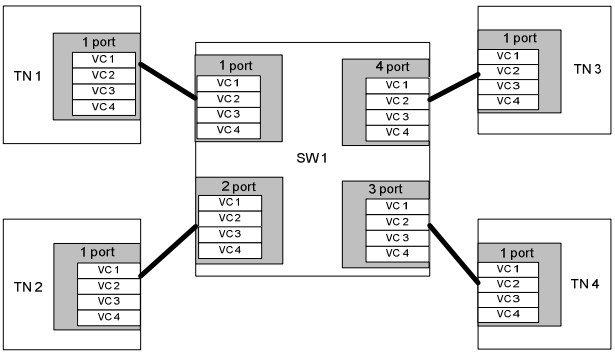

Fig. 3 Network model

The Network model comprises a router with 4 ports, each of which can work with 4 virtual channels. Terminal nodes generate packets in a random time moments. At these random moments the terminal node sends the generated packets to each virtual channel. The destination nodes for each virtual channel are also chosen randomly and can be different for the virtual channels. This configuration can lead to a potential possibility of data packets flow concurrency in the output port.

IV. Simulation results

The network was simulated on the adapted DCNSimulator model [9]. In this case we used the router and node models which comprise only the Virtual Channel and the Network Layers (this gave an opportunity to reduce the simulation time and to obtain more detailed results). The link bandwidth in the model is set to 1 Gbit/s.

The results of the simulation can significantly depend on the router model implementation features such as local clock frequency and link capacity within the router.

Let us consider the case when each virtual channel has its own particular priority level, which corresponds to the virtual channel number: VC1 – the highest priority, VC4 – the lowest. The packet length does not exceed the frame length. Fig. 4- Fig. 8 shows the simulation results for the 1st, 2nd and 3rd way of router implementation for each virtual channel, when size of data packet is 250 bytes. Fig. 9 - Fig. 13 shows the simulation results for the 1st, 2nd and 3rd way of router implementation for each virtual channel, when size of data packet is 750 bytes. In these cases packets for VC4 were sent first, then for VC3, VC2, VC1 successively. The time between packets generation for different virtual channel is 100 ns. N is 256 bytes. Average data transmission delay of high priority (VC1) packets is similar when we use 1st and 3rd way of router implementation.

The 2nd implementation of network layer differs by a large delay value of high priority packet as you can see on figures.

Fig. 14 shows the simulation results for the 1st, 2nd and 3rd way of router implementation for virtual channel 1, when size of data packet is 750 bytes and there are interruptions the transmission of data with lower priority in 3rd way with different N. In this case packet generation time has exponential distribution. Delay is bigger for the 2nd way of the router implementation than for 1st, 3rd way. The smallest average data transmission delay of high priority (VC1) packets is observed in situations where packet transmission with lower priority is interrupted after 32 or 64 bytes transfer.

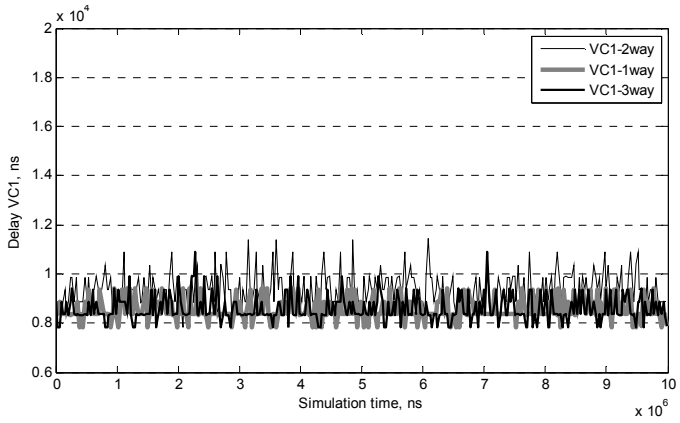

Fig. 4 Comparison of the packet transmission time via VC1 (the packet size = 250 bytes) in case of different implementations of network layer

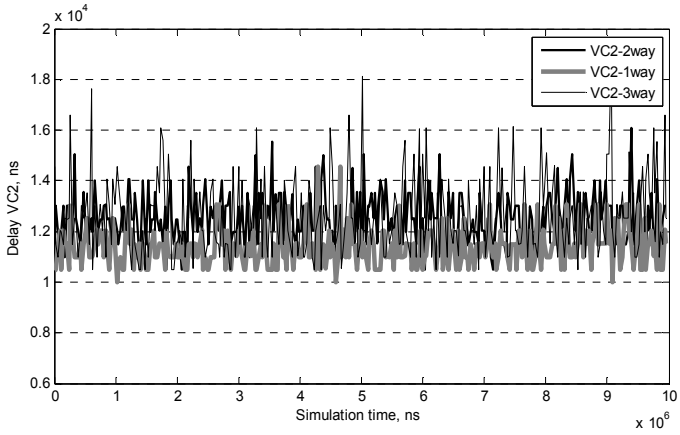

Fig. 5 Comparison of the packet transmission time via VC2 (the packet size = 250 bytes) in case of different implementations of network layer

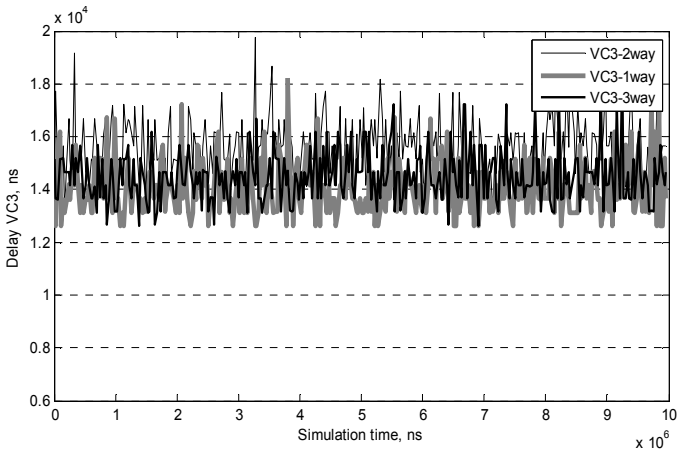

Fig. 6 Comparison of the packet transmission time via VC3 (the packet size = 250 bytes) in case of different implementations of network layer

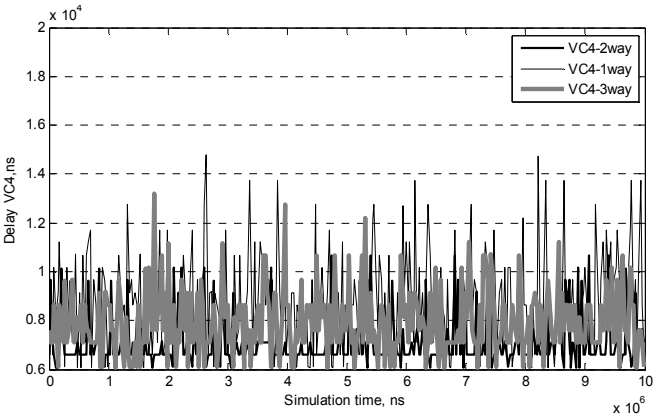

Fig. 7 Comparison of the packet transmission time via VC4 (the packet size = 250 bytes) in case of different implementations of network layer

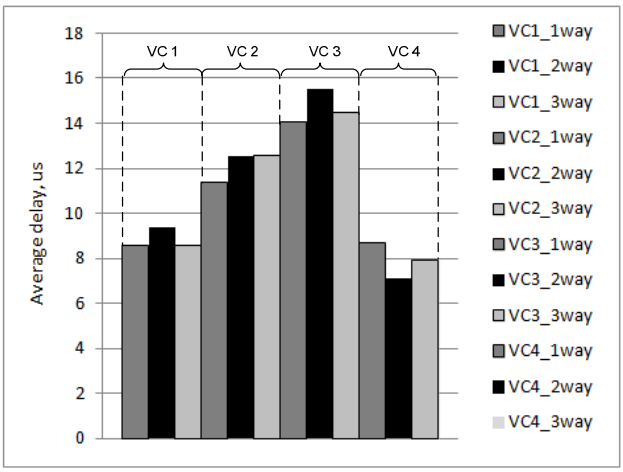

Fig. 8 Bar chart of the average packet transmission time via VC1, VC2, VC3,VC4 (the packet size = 250 bytes)

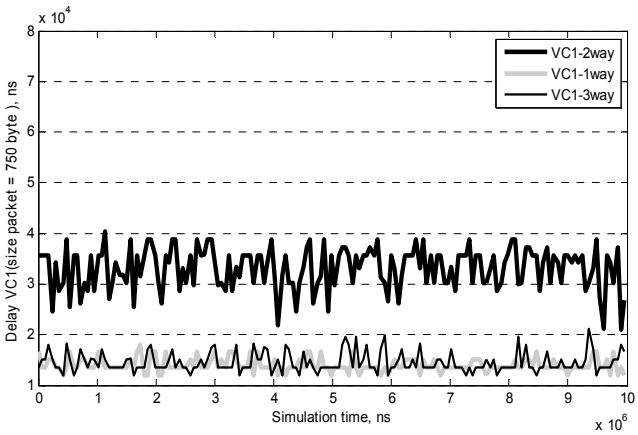

Fig. 9 Comparison of the packet transmission time via VC1 (the packet size = 750 bytes) in case of different implementations of network layer

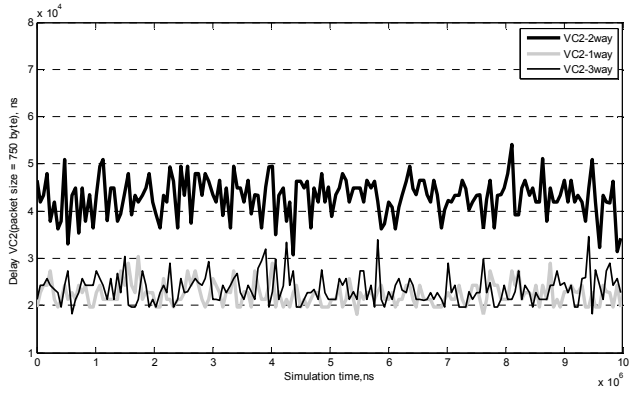

Fig. 10 Comparison of the packet transmission time via VC2 (the packet size = 750 bytes) in case of different implementations of network layer

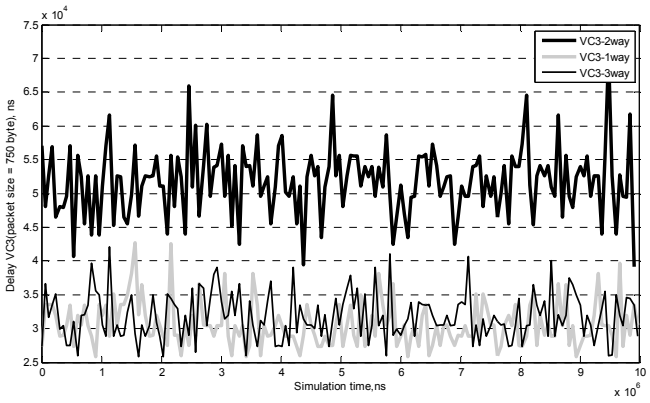

Fig. 11 Comparison of the packet transmission time via VC3 (the packet size = 750 bytes) in case of different implementations of network layer

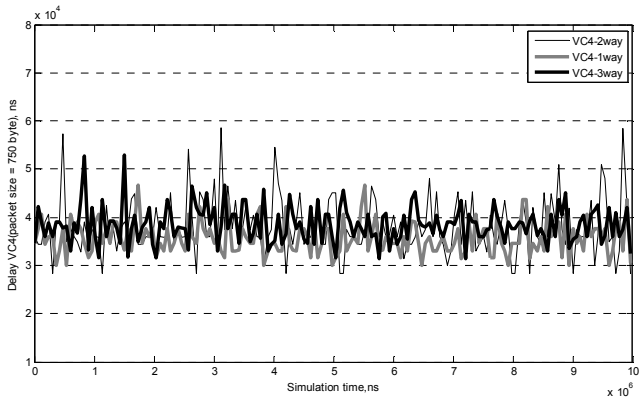

Fig. 12 Comparison of the packet transmission time via VC4 (the packet size = 750 bytes) in case of different implementations of network layer

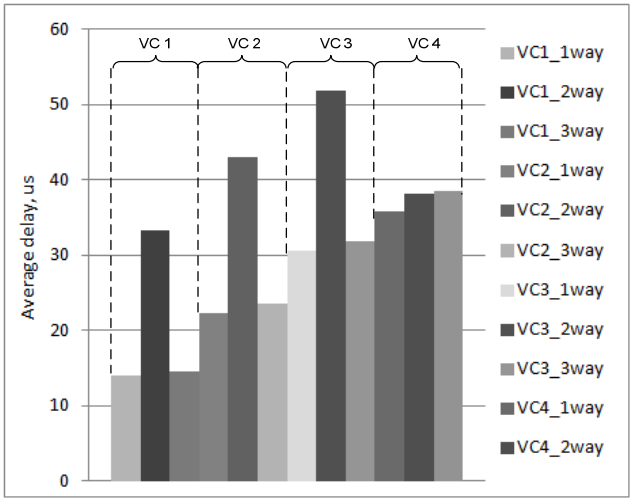

Fig. 13 Bar chart of the average packet transmission time via VC1, VC2, VC3, VC4 (the packet size = 750 bytes)

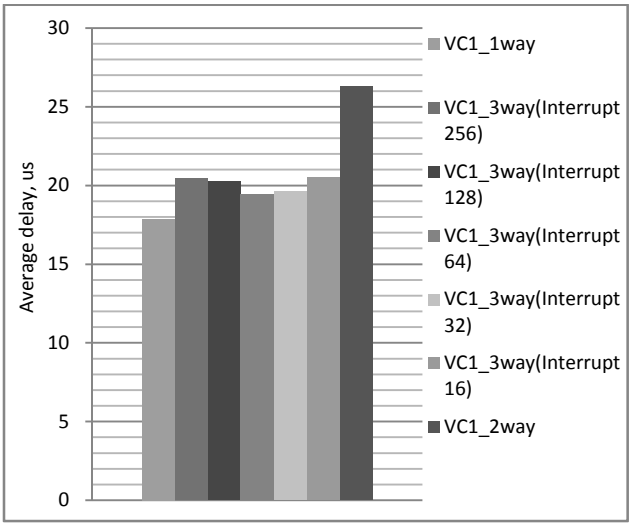

Fig. 14 Bar chart of the average packet transmission time via VC1 (the packet size = 750 bytes). Exponential distribution of packet generation time.

V. CONCLUSION

The comparison of the achievable timing characteristics for different ways of router implementation showed that if the packet size is smaller than the frame size then the average packet transmission time for 3rd way is almost similar to 1st way. The 1st way of router structure hardware is essentially constrained [10].

Delay of the high priority traffic grows faster for the 2nd way of the router implementation. Therefore, the 2nd way of the router implementation can be used for the networks with the packet length shorter than frame size. In this case it will provide scheduled, bandwidth reserved and priority qualities of service. The packet lengths larger than the frame size while using the 2nd way of the router implementation result in degradation of the timing characteristics in comparison with the 1st and 3rd way. This degradation of characteristics grows proportionally to the packet's length of the virtual channels of

low priorities. Consequently, the 2nd way of the router implementation in networks where long packets are transmitted is possible only when there are no hard real time requirements. The 3rd way of the router implementation essentially decreases these disadvantages. The average packet transmission time and achievable link utilization in this case are almost similar to the 1st way of the router implementation.

Delay is 10% bigger for the 2nd way of the router implementation than for 1st way, when the packet length shorter than frame size and delay is 1% bigger for the 3rd way of the router implementation than for 1st way. Delay is 50% bigger for the 2nd way of the router implementation than for 1st way, when the packet length longer than frame size and delay is 17% bigger for the 3rd way of the router implementation than for 1st way. Therefore, the achievable characteristics for the scheduled service and delay value for this 2nd way of router implementation are lower.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under contract n° 14.578.21.0022

REFERENCES

[1] *S. Balandin, M. Gillet, "Embedded Network in Mobile Devices", International Journal of Embedded and Real-Time Communication Systems (IJERTCS), Issue 1(1), pp 22-36. 2010.*

[2] *D. Bueno, "Simulative analysis of the RapidIO embedded interconnect architecture for real-time, network-intensive applications", in Local Computer Networks, 29th Annual IEEE International Conference, Nov. 2004, pp. 710 – 717.*

[3] *S. Parkes, "SpaceWire for Adaptive Systems", ahs 2008 NASA/ESA Conference on Adaptive Hardware and Systems, pp.77-82.*

[4] *J. Duato, S. Yalamanchili and L. Ni, Interconnection Networks. An engineering approach. San Francisco: Morgan Kaufmann Publishers, 2003.*

[5] *J. Fleischmann, "Prototyping networked embedded systems", Computer., vol.32, Feb. 1999, pp. 116 – 119.*

[6] *S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, D3.2-SpaceWire-RT Updatated Specification. Annex 1 SpaceFibre Standart, Febrary 2013*

[7] *W. James Dally, Principles and Practices of Interconnection Networks. San Francisco: Elsevier, 2004.*

[8] *S. Parks, C. McClements, M. Dunstan, A. Ferrer, A. Gonzalez. SpaceFibre.SpW WG, October 2012*

[9] *A. Eganyan, E. Suvorova, Y. Sheynin, A. Khakhulin, I. Orlovsky. DCNSimulator – Software Tool for SpaceWire Networks Simulation. International SpaceWire Conference 2013, June 2013*

[10] *N. Matveeva, E. Suvorova, V.Olenev, I. Lavrovskaya, I. Korobkov, A. Eganyan. SpaceFibre Quality of Service Features Support in the Network Level. International SpaceWire Conference 2013, June 2013*

SpaceFibre Demonstrator: Demonstration and Testing

SpaceFibre, Short Paper

Paul Rastetter, Tim Helfers *(Authors)*
Astrium GmbH
Munich, Germany
Paul.Rastetter@astrium.eads.net

A. Antonakou, G. Dramitinos, C. Papadas
Integrated Systems Development S.A.
Athens, Greece;
Steve Parkes
University of Dundee,
Dundee, United Kingdom

Abstract — Currently Astrium GmbH and Intergrated Systems Development (ISD) S.A. are planning the development of a demonstrator for SpaceFibre. The SpaceFibre demonstrator will be used to execute functional performance tests and EMC (Electro Magnetic Compatibility) tests. University of Dundee is program prime contractor and provides Astrium with the SpaceFibre IP core.

The work is shared between the two partners in the following way:

- Astrium: Prime Contractor and Technical Coordination; FPGA Design; EMC Testing
- ISD: Development of Demonstrator Board including housing, development of test bed and functional performance testing

The driving requirements for this development are:

- SpaceFibre performance, while implementing it into space equivalent components
- Design and MAIT of the demonstrator in such a way that representative EMC testing is possible

I. Demonstrator Test-bed

The demonstrator test-bed will include the following items:

- STAR Fire unit from STAR-Dundee Ltd,
- personal computer connected to the STAR Fire unit via USB executing the SpaceFibre Link Analysis software,
- standard logic analyzer providing PODs able to interface standard connectors,
- oscilloscope able to monitor the eye diagram of the receiving differential signals using contact-less probes and
- clock generator used to provide an external clock signal

The STAR Fire embedded link analyser that exposes the decoded 8B/10B signals on the respective connector and the Logic Analyser that will capture these signals will enable monitoring of frame transfers and analysing traffic. The external clock generator will allow system validation under various traffic conditions.

Each test will utilize one or more pieces of the above mentioned equipment. Moreover, each test may utilize either a single demonstration board or multiple demonstration boards in a daisy chain configuration.

The test-bed configuration for board-level, functional and performance testing is shown in figure 1.

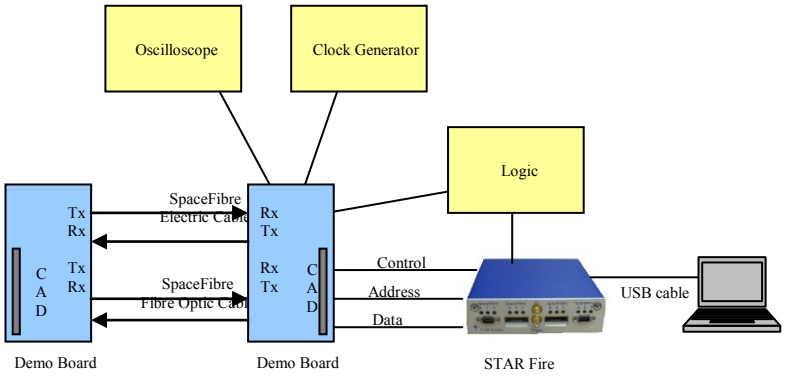

Figure 1: Demonstrator Test Bed

II. Functional and Performance Tests

For the functional and performance tests three different test modes will be implemented:

- Virtual channel loopback
- Packet generator and packet checker
- External host interface

The functional testing will be done in several steps using the different test modes.
The performance test will show the upper limit of the speed of the SpaceFibre Demonstrator.

III. EMC Testing

The diagram of the test setup for the EMC testing is shown in figure 2. The board located in the EMC chamber is accommodated in a mechanical housing (box), which is just penetrated by the SpaceFibre and power supply connector. The data traffic is generated by the setup externally of the EMC chamber, which can be controlled via the STAR Fire test equipment. The Demonstrator Board within the EMC chamber just loops back the data. In this way, data traffic is generated in each direction, without the need for another external interface. Therefore, no additional external interface is required on the demonstrator board, penetrating the housing.

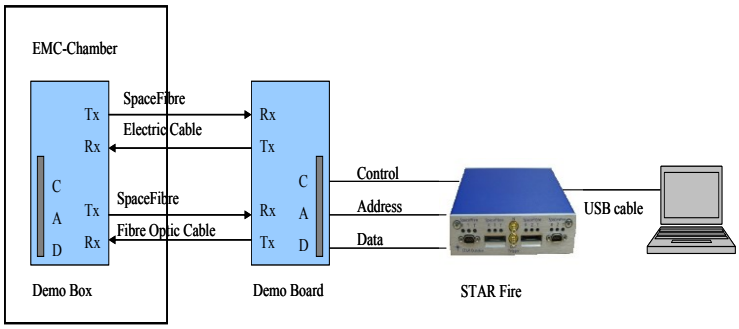

Figure 2: EMC Demonstrator Test Setup

The following EMC tests are planned:

- Conducted Emission tests
- Radiated Emission tests, Electric field
- Conducted Susceptibility Test
- Radiated Susceptibility Test, Electric field
- Electro-static Discharge

Advanced Oversampling Techniques for the SpaceFibre

SpaceFibre, Short Paper

Vladimir Goussev, Dmitri Skok, Mikhail Maksimovskij, Tatiana Solokhina, Jaroslav Petrichkovich
RnD ELVEES
Moscow, Russia
vgoussev@elvees.com, dskok@elvees.com, mmaksimovskij@elvees.com, tanya@elvees.com

***Abstract* — The SpaceFibre provides numerous advantages over the SpaceWire: QoS, FDIR, high transmission rate, galvanic isolation, lower cable mass. At moderate speeds, e.g. 10-200 Mbps, the SpaceFibre interface implementation can be significantly simplified with oversampling techniques, which does not require PLLs and analog CDR blocks. It makes possible fully digital implementation of the SpaceFibre interface in a simple ASIC or FPGA.**

This paper presents an oversampling technique based on digital signal processing. The technique allows to enhance transmission rate, operational distance, phase/amplitude distortion tolerance and higher noise immunity in compare with the traditional edge detection method.

***Index Terms* — SpaceWire, SpaceFibre, oversampling, clock and data recovery.**

I. Introduction

The successor of the SpaceWire – SpaceFibre [1], a.k.a. SpaceWire2 - brings a number of advanced features to the spacecraft aboard networking: Quality of Service, Fault Detection, Isolation and Recovery, Low-latency signaling, Multi-lane connection, High speed, Low mass cable. Many of the above features are provided by the usage of single differential pairs (Rx and Tx) per lane direction with the NRZ signaling and 8b/10b encoding at the physical level.

Typical implementation of the NRZ signaling requires specially designed analog circuits, e.g. PLL-based clock-data recovery [2-4]. Unfortunately, these circuits are not always available for space-grade FPGAs and ASICs technologies. To avoid this limitation, an oversampling SpaceFibre mode was suggested at lower speeds (up to 200-400 Mbps) [5]. The oversampling technique can be designed as a digital domain circuitry, which simplifies SpaceFibre implementation and allows to combine the functionality of the SpaceFibre and the design simplicity of the SpaceWire in the space-grade FPGAs and ASICs.

Another benefit of the suggested oversampling mode is that it opens room for digital signal processing of oversampled data to enhance performance of the communication in terms of transmission distance, transmission rate, immunity to EMI noise or immunity to transmission media quality. An advanced approach based on digital signal processing of single-bit oversampled data is presenter in this paper.

II. Digital Oversampling Technique

The digital oversampling technique for the NRZ signaling is presented on Fig.1 (architecture block diagram) and Fig.2 (waveform).

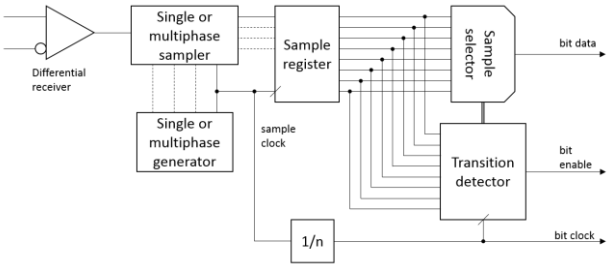

Fig. 1. Oversampling architecture block diagram

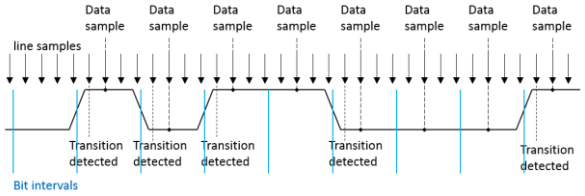

Fig. 2. Oversampling waveform

The differential receiver converts analog transmission line signal to digital domain, so all further processing is performed with digital circuitry. The key element of the circuitry effecting on the receiver performance is the transition detector.

In the simplest case, the transition detector treats signal transitions as a time base for next data samples using known oversampling ratio – the relation between transmission bit rate and sampling rate. For example, if the transmission rate is 200 Mbps and the sampling rate is 1 GHz then the oversampling ratio is 5, so once a transition is detected next 3rd, 8th, 13th etc samples are selected as data samples as long as next transition is detected. A slight modification of this method does not use data selection, as time intervals between the consequence transitions are enough to reconstruct data bits.

As almost everything simple and straight forward, the above methods are not good on practice because of transition jitter and multiple transition at bit interval boundaries caused by inter-symbol interference and signal noise. Robust digital oversampling methods suggest different techniques for transition filtering and bit interval boundaries calculation [6].

One of the most important question is oversampling ratio, which is a trade-off between transmission bit rate, transition performance (distance, noise immunity, BER) and transition detector complexity. Larger oversampling ratio provides more options for transition detector to achieve better performance and/or have simpler implementation. At other side, the maximum sampling rate depends on the technology used for implementation, so it directly limits transmission bit rate. The minimum oversampling ratio can be estimated using the eye diagram, as it shown on Fig.3 for the SpaceFibre – there have to be at least one sample in the eye opening.

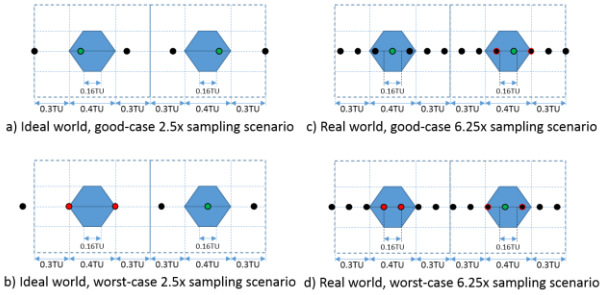

Fig. 3. SpaceFibre oversample ratio

As it can see from the figure, the minimum SpaceFibre oversampling ratio should be greater than 2.5 in the ideal case (zero-noise differential receiver, jitter-less generator). In the real world, the minimum oversampling ratio of up to 6 is required, depending on real values of the differential receiver noise and generator jitter.

III. Suggested Digital Signal Processing Technique

Let the following assumptions:

- The transmitter sends 8b/10b (or any other DC-balanced code with the limited run length of 0's and 1's) data B_n at the transmission bit rate F_{bit};
- Transmission media is described by the linear causal operator whose pulse response function is h(t).
- Reasonably good initial estimate of h(t), h'(t) is provided: h(t)=0, t<0; h(t)<ε, t>T_h.. Since h(t) never becomes identity zero, we chose some finite reasonably small ε.
- The receiver samples the output of the media at the sample rate F_s, returning the sign of its input V_{in}, so that '0' denotes V_{in}<0 and '1' – $V_{in} \geq 0$, F_s>F_{bit};
- Nominal relation F_{bit}/F_s is known, but the exact ratio is unknown in advance and is time-varying.
- Optionally, the receiver input is affected by the additive Gaussian noise with the standard deviation σ.

Then the signal at the input of the receiver I(t) is the unity amplitude rectangular pulse of width $1/F_{bit}$ modulated by B_n convolved with h(t):

$$I(t) = \Xi(t,B) \equiv \left\{ \sum_{k=0}^{N} (2b_k - 1) \cdot U\left(\frac{t}{F_{bit}} - k \right) \right\} * h(t) \qquad (1)$$

Where:

$$U(t) = \begin{cases} 0, t < 0, \\ 1, 0 \leq t \leq 1, \\ 0, t > 1. \end{cases} \qquad (2)$$

is rectangular pulse and "*" denotes the convolution.

The example of the measured U(t)*h(t) sampled with the frequency of 128 F_{bit} is shown on Fig.4.

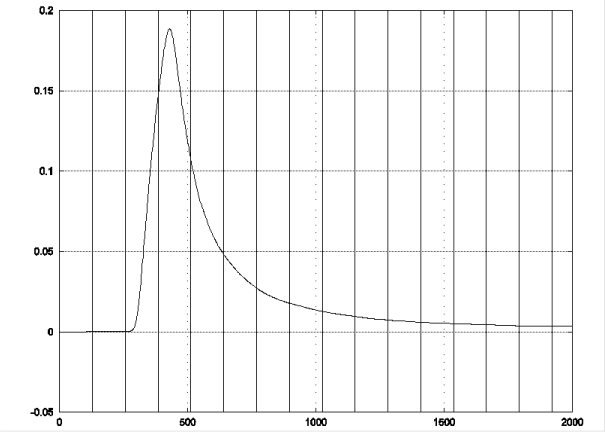

Fig. 4. Bit response relative to the bit interval

The receiver determines the sign of its input at frequency F_s, resulting in the sequence C={c_n}:

$$c_n = S(I, n, \tau) \equiv sign\left(I\left(\frac{n}{F_s} + \tau \right) \right) \qquad (3)$$

Where:

$$sign(x) = \begin{cases} 0, x < 0, \\ 1, x \geq 0. \end{cases} \qquad (4)$$

Let denote the concatenation of two sequences $A = \{x_n\}_{n=a}^{b}$ and $B = \{y_k\}_{k=c}^{d}$ by {A, B}, i.e.:

$$\{A, B\} = \{x_a, \ldots, x_b, y_c, \ldots, y_d\} \qquad (5)$$

The goal is, given the C and the hint {b_k, ..., b_{k+p}}, to determine {b_n} starting with n=p+1.

Let define the error functional:

$$E(f,C,\tau)\equiv\sum_{k=k_1}^{k_2}e_k\,, \tag{6}$$

Where:

$$e_k=\begin{cases}0, iff\left(\tau+\dfrac{k}{F_s}\right)\cdot(c_k-0.5)>0,\\ f\left(\tau+\dfrac{k}{F_s}\right)^2, otherwise.\end{cases} \tag{7}$$

A. Noiseless case.

In the absence of noise, E(I, C, 0) = 0 and E(I, C, d) > 0 for $|d| > \varepsilon > 0$. Also:

$$E\left(I,\{c_{0:k},\hat{c}_k,c_{k+1:n}\},0\right)>0 \tag{8}$$

for any k, provided that B and h(t) are "good enough" in some sense.

B. Initial phase estimate.

As previously mentioned, before we can start receiving data bits, we need a hint $\{b_k - b_{k+p}\}$. In practice, this is the case, because the communication begins with the known sequence of symbols, "comma", in the case of SpaceFibre. What we need to do is to determine the initial phase of the sequence, i.e. to determine the position of the next bit to decode relative to C.

This can be done in the following way. As h'(t) is given, a fragment of the signal P'(t) input to the receiver can be constructed as the convolution of the rectangular pulse of duration 1/Fbit, modulated by the $\{b_k, ..., b_{k+p}\}$, and h'(t) (operator Ξ' is similar to Ξ in (1), but uses h'(t) instead of h(t)):

$$P'(t)=\Xi'\left(t,\{B_k,...,b_{k+p}\}\right) \tag{9}$$

Since T_h can span many bit intervals $1/F_{bit}$, the beginning of P'(t) is affected by the unknown bits b_n, n<k. Similarly, the tail of the P'(t) is affected by b_m where m>k+p. Let N_h is T_h expressed in the bit intervals: $N_h = [T_h\ F_{bit}]$. To get the pattern P(t), that is defined only by the known bits from $\{b_k, ..., b_{k+p}\}$, just cut off the head of length N_h/F_{bit} and the tail of the same length from P'(t): $P(t)=P'(t+N_h/F_{bit})$, $t<p/F_{bit}$. Since we need reasonably long useful bit pattern to reliably determine initial phase, say, 20 bits, it follows than p should be $p>20+N_h/F_{bit}$. Let $u=p-N_h/F_{bit}$ is the number of "useful" bits that define the pattern P(t). Then the number of the receiver samples for the duration of P(t) is:

$$l=\left\lfloor u\frac{F_s}{F_{bit}}\right\rfloor \tag{10}$$

With P(t) at hand, find the minimum $E(P, \{c_s, ..., c_{s+l}\},\tau)$, where $0 \le \tau \le 1/F_{bit}$ and $0 < s < D_{srch}$, D_{srch} being the depth of the search. Then, τ and s define the position of the start of the 1'st bit in P(t) in terms of index in C and time offset.

C. Next bit extraction.

To extract the next bit from the sequence C, let consider the two families of hypotheses:

$$\begin{aligned}H_0(t)&\equiv\Xi(t,\{B_k,'0',T_d\}),\\ H_1(t)&\equiv\Xi(t,\{B_k,'1',T_d\})\end{aligned} \tag{11}$$

T_d denotes all the 2^d combinations of d bits.

Since the operator h(t) is causal, we need only finite d. In practice, with realistic h(t), we only need d equal to 1 to 3, since later bits influence rapidly vanishes. The following cases are possible:

1. There exists exactly one **n** and the corresponding sample c_n, such that all hypotheses in $H_0(t_n)$ have the same sign, and all hypotheses in $H_1(t_n)$ have the opposite sign. Then, the value of c_n unambiguously identifies the right family of hypotheses and the value of the next bit.
2. There are no **n** the above is true. It means that no receiver sample hits the eye opening (or no open eye exist). It can also occur when the oversampling ratio F_s/F_{bit} is insufficient for the given h(t). For this case, the same sequence C can correspond to at least two different B, and no further processing of C can select the right one.
3. More than one **n** satisfy the above condition. The case when all such **n** resolve to the same bit value is trivial. The opposite can take place when h(t)≠h'(t) or in the presence of noise (σ > 0). The resolution of this is discussed in chapter F.

The key concept discussed above is illustrated on Fig.5.

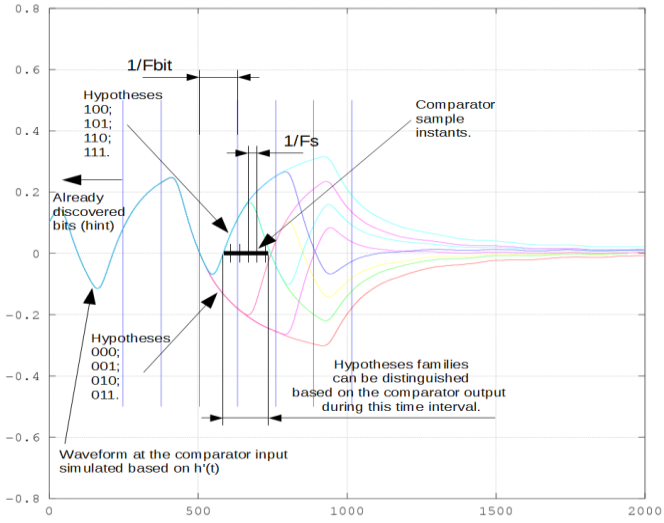

Fig. 5. Bit extraction concept

When the next bit is discovered, it is attached to the string of known bits and the step repeats.

D. Phase and frequency lock

As the oversampling rate Fs/Fbit is not known exactly and varies over time, we must adjust it accordingly to keep phase synchronization. This can be done by adding a small delta to the current value of the oversampling rate and phase offset depending on the sign of the instant phase error. To get the latter, take a pattern P(t)) = Ξ(t, B) of m recently discovered bits B and compare E(P, C, -1/10F_{bit}) and E(P, C, 1/10F_{bit}). If the former is greater than the latter, then decrease the oversampling rate and phase by the small step, else increase both.

The number of bits in B should provide at least one value transition. From the properties of the 8b/10b, 10 bits would be the adequate amount.

E. h'(t) extraction.

From implementation perspective, it may be more efficient to store the bit response r'(t)=U(t/F_{bit})*h'(t) instead of h'(t) alone. It can be done in the form of vector r'_n of the sampled values of r': r'_n=r'(n/(128*F_{bit})). Then, the computation of Ξ in the sampled form would be:

$$\xi_n(B) = \sum_{k=0}^{N} (2b_k - 1) \cdot r'_{n-128k} \qquad (12)$$

One of the simplest algorithms to make r'_n to converge to r_n would be, whenever the computed value of e_k is nonzero, adjust the corresponding r'_n by the small delta in the direction that decreases e_k.

F. Noise estimation and handling.

In the presence of noise with the standard deviation σ, the probability of incorrect value of the comparator output is equal to that of instant noise value exceeds the distance from zero to the corresponding hypothesis value at the given point of time. Since both hypotheses are deterministic in the assumption that hint bits are decoded correctly, the reliability of each receiver sample can be computed. And vise versa, measuring the receiver error rate at particular samples and knowing their corresponding reliability, one can estimate noise sigma. So that the decoding algorithm would continually estimate the noise level and adjust weight of different comparator samples accordingly.

IV. Conclusion

In the 'classic' blind oversampling technique we use the fixed phase in the bit interval to sample data. This is OK for the environment, where additive random noise is small enough and can be neglected. The proposed method provides a 'dynamic phase' of sampling, i.e. the received samples set is chosen on the bit-by-bit basis, depending on the de-facto media properties and the previous bit pattern. This provides more samples to decode every single bit, together with the information of the reliability of each sample. As only a little fraction of data bits is defined by one or two samples, total BER in the environment with noise and dispersion can be improved by an order of magnitude or so.

As a side result, the method provides background measurement of the pulse response of the media and the noise level.

Although the efficient hardware implementation left beyond the scope of the paper, rough estimate shows that the complexity is not expected to be too high and forms the order of dozen 8-bit additions and table lookup per sample. The memory requirement basically defined by the storage of h'_n, some 1 K bytes or less.

The method can be applied with the minimum modifications to the multi-level sampling case (ADC instead of the comparator) as well.

References

[1] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, “SpaceFibre Standard Draft F3”, University of Dundee, September 2013

[2] R. Walker, “Clock and Data Recovery for Serial Digital Communication”, Agilent Laboratories, ISSCC Short Course, February 2002

[3] B. Razavi, “Challenges in the design high-speed clock and data recovery circuits”, IEEE Communications Magazine, Volume 40, Issue 8, August 2002, pp. 94 – 101

[4] M. Hsieh, G. Sobelman, “Architectures for Multi-Gigabit Wire-Linked Clock and Data Recovery”, IEEE Circuits and Systems magazine, fourth quarter, 2008

[5] S. Parkes, “SpaceWire-RT Update”, SpaceWire Workgroup meeting #19, presentation, October 2012

[6] S. I. Ahmed, Tad A. Kwasniewski, “Overview of oversampling clock and data recovery circuits”, Carleton University, 2009

Onboard Equipment & Software (Short)

The draft ECSS SpaceWire Backplane Standard

SpaceWire Onboard Equipment and Software, Short Paper

Alan Senior, John-Paul Coetzee
Thales Alenia Space (UK)
Bristol, United Kingdom
alan.senior@thalesaleniaspace.com
john-jaul.coetzee@thalesaleniaspace.com

Jørgen Ilstad
European Space Agency, ESTEC
Noordwijk ZH, Netherlands
jorgen.ilstad@esa.int

***Abstract*— The SpaceWire standards are maintained and issued formally as ECSS documents (e.g. ECSS-E-ST-50-12C) and this means that equipment designed by different agencies is interoperable, which has significant benefits.**

SpaceWire is mainly used between instrument units, however to facilitate a high level of integration of onboard systems it is proposed that an ECSS SpaceWire Backplane standard should be created and adopted, the backplane offering power, signal and impedance-matched connectivity for high-speed serial links such as SpaceWire and SpaceFibre.

The SpaceWire Backplane standard will assist in the aim of creating a common onboard infrastructure to be used across many different mission applications by encouraging design reusability at the sub-unit (PCB plug-in module) level. A key advantage of a SpaceWire Backplane is the scalability of the design.

A draft SpaceWire Backplane ECSS standard has been created by TAS-UK as part of an ESA contract. It is based around the Smiths Connectors Nexus modular connector which can be populated with a variety of different contact inserts for power, signal and high-speed data and thus can be tailored to a particular application.

The standard will specify the physical dimensions, connectors and electrical interfaces of the unit backplane and plug-in module. Units and plug-in modules will then be interoperable, so for example a module produced by one vendor will correctly fit into a unit from another vendor and interface correctly with the backplane electrical signals. It permits both non-redundant and redundant units to be built and does not dictate the backplane SpaceWire network architecture. Recommendations are made to assist in the creation of reliable fault-tolerant systems.

It is intended that this ECSS document takes advantage of specifications already existing, namely other ECSS and also ANSI/VITA VPX standards. The principal VITA standards are referenced include VPX (VITA 46), OpenVPX (VITA 65) and a new draft standard SpaceVPX (VITA 78). It is anticipated that there will be a harmonisation activity in future between the SpaceVPX and this ECSS document before the formal ECSS standard release.

This paper is a walk-through of the ECSS Backplane standard, it describes the thought processes and rationale behind it and the anticipated advantages of adopting it.

***Index Terms*— SpaceWire, SpW, SpaceFibre, SpFi, ECSS, SpaceVPX, Backplane, Networking, Avionics, Spacecraft Electronics.**

I. Introduction

A draft SpaceWire (SpW) [1] Backplane standard has been created using the ECSS drafting rules; the document is one output of an ESA contract performed by Thales Alenia Space UK (TAS-UK, formerly SEA Space Division). It is anticipated that the draft will evolve as part of a future ESA contract.

The draft does not currently constitute an official document, although it may at a future date be submitted to ECSS for publication. It is anticipated that there will be a harmonisation activity between the SpaceVPX specification and this ECSS document before the formal ECSS standard release.

To maximize flexibility a particular network architecture is not mandated, however a design based on the outputs of the ESA Modular Architecture for Robust Computing project [2] is recommended. The SpW backplane may be either passive or active [3].

II. Avionics Applications

In a spacecraft avionics unit the Printed Circuit Boards (PCBs) or modules are typically connected together via a backplane (Fig.1).

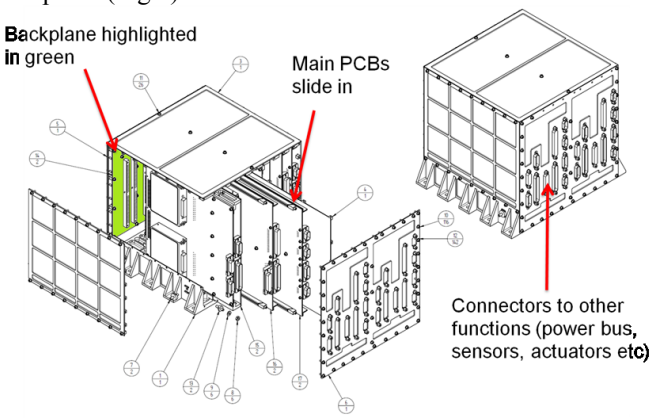

Fig. 1. Backplane within a Spacecraft avionics unit

The electrical interfaces at the backplane interface used are typically non-standard, consisting of parallel busses, discrete lines at different voltage levels and device/technology dependant busses. The mechanical interfaces also vary between equipment suppliers.

It is anticipated that a SpW backplane standard will assist in creating a common onboard infrastructure to be used across many different mission applications by encouraging design reusability at the sub unit (PCB plug-in module) level.

A key advantage of employing SpW is the scalability of the design and the simplification of Assembly Integration and Test (AIT) activities (Fig. 2).

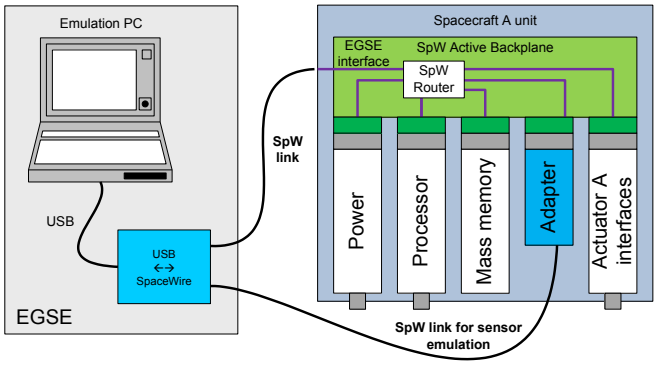

Fig. 2. A SpW backplane simplifies AIT activities

The modules are commonly:

- Power supply – provides internal unit power but may be commanded (e.g. on/off) and provides housekeeping telemetry (e.g. on/off and trip status, voltage, current, temperature).
- Processor – provides the computing resource for complex decision making and data processing. Gathers telemetry and controls system elements.
- Mass Memory – provides a repository for science data, application software images and intermediate products from data processing.
- Sensor (input) interfaces – provides interfaces to multiple sensors types (e.g. switches, temperature/voltage/current sensors, scientific sensors).
- Actuator (output) interfaces – provides outputs for control purposes (e.g. heaters, thrusters pulses, on/off signals).

Clearly all these module types have different electrical interface requirements that need to be supported by the standard.

III. Backplane Standard - Summary

The aim is that the standard specifies the physical dimensions, connectors, thermal and electrical interfaces of the unit and plug-in module. Units and plug-in modules designed to the standard will then be interoperable, so for example a plug-in module produced by one vendor will correctly fit into a unit from another vendor and interface correctly with the backplanes electrical signals.

The standard permits both non-redundant and redundant units to be built, it does not dictate the backplane SpW network architecture but provides recommendations to assist in the creation of reliable fault tolerant systems.

The standard may be tailored for the specific characteristics and constraints of a space project in conformance with ECSS-S-ST-00, however the aim should be that interoperability is not compromised.

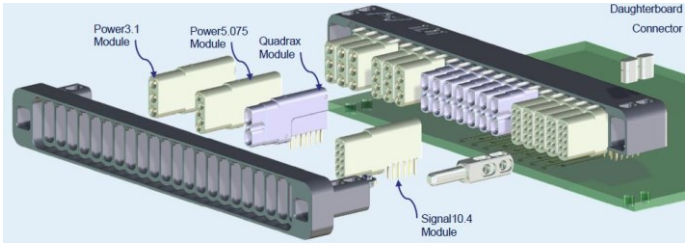

Fig. 3. Smith Connectors (Hypertac) module Nexus connector

The standard is based upon the Smiths Connectors HYP_6890 "Nexus" connector [4] shown in Fig. 3, it has with 22 insert bays, additional connector designs are anticipated to be available in due course. The key advantage of this connector is that it can be populated with a variety of different contact inserts, thus it can be tailored to a particular application.

IV. Connector Designations

The Smiths Connectors HYP_6890 contains 22 insert bays, there are 4 connector insert variants used in the backplane standard, plus one blank insert (Fig. 4). The pin numbers are allocated as defined in Fig. 5.

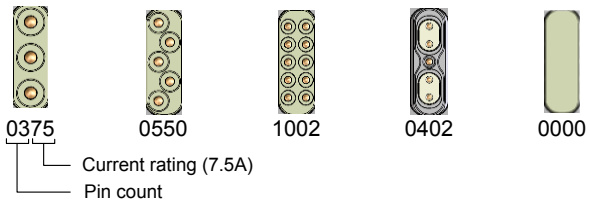

Fig. 4. Connector insert designations

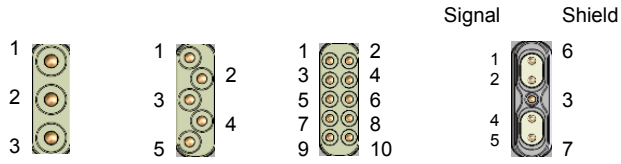

Fig. 5. Plug insert pin designations

The module interface is via either 1 or 2 connectors designated as shown in Fig. 6 and Fig. 7 respectively.

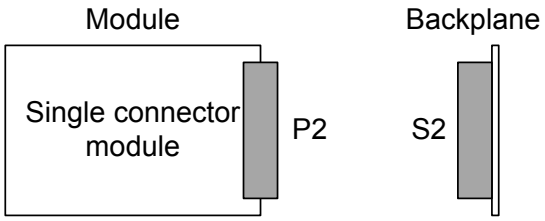

Fig. 6. Single connector module

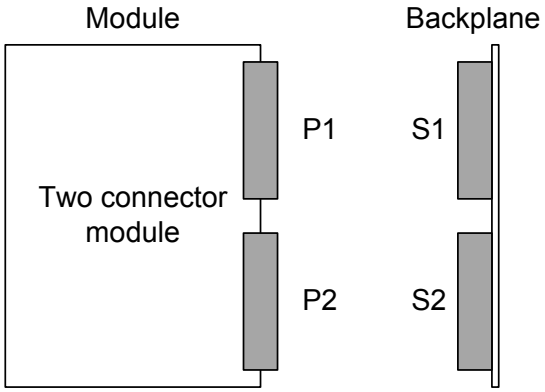

Fig. 7. Two connector module

An example module level pin designation is presented in Fig. 8.

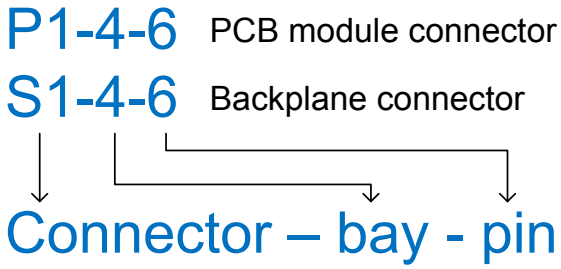

Fig. 8. Two connector module

V. Module Interfaces

The insert configurations have been defined for 3 module types:

- Power
- Router
- Cluster (input/output module)

The Power Module configuration is presented in Fig. 9. This provides:

- Power connections (7.5A and 5A rated)
- 2 SpW links
- LVDS clock and sync signals
- Low rate serial input/output (I/O)
- Logic I/O
- Analogue I/O

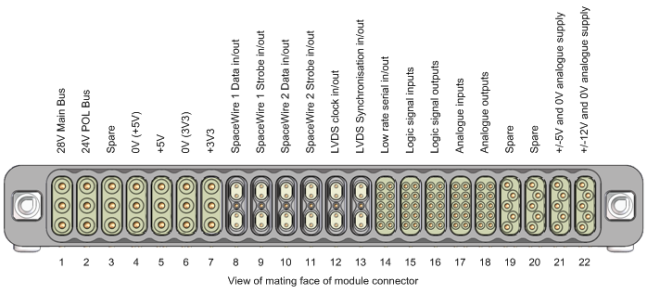

Fig. 9. Power Module insert allocation

The Router Module interface (Fig. 10) has a lower number of power and discrete I/O pins but supports 8 SpW links in a single connector to support its role as a SpW router function. Using two connectors on a double Eurocard means the module could support 16 SpW links.

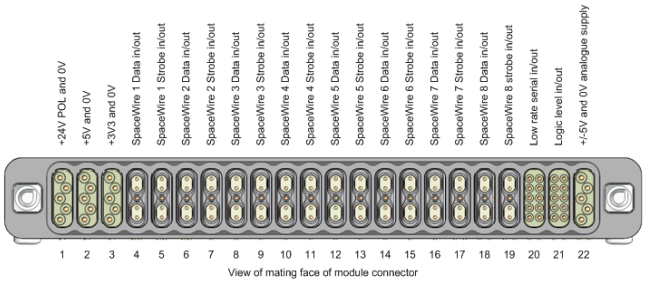

Fig. 10. Router Module insert allocation

The Cluster Module (Fig. 11) provides:

- Power
- 4 SpW links
- LVDS clock and sync signals,
- Logic discrete I/O
- Analogue discrete I/O

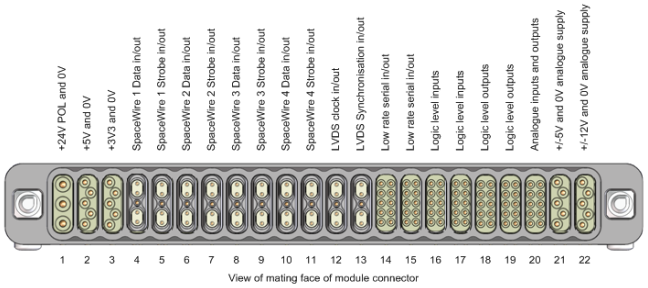

Fig. 11. Cluster Module insert allocation

The pin allocations of each insert type are defined in the standard, four examples are given in Table I. Compliance with this pin allocation will ensure compatibility at the insert level so that other combinations of insert configurations could be defined.

TABLE I. EXAMPLE INSERT PIN ALLOCATIONS

Function	Insert type	Insert pin	Signal allocation
28V main bus	0375	1 2 3	+28V nominal NC 0V (+28V)
+5V and 0V	0505	1 2 3 4 5	0V (+5V) +5V +/-10% 0V (+5V) +5V +/-10% 0V (+5V)
SpW X S in/out	0402	1 2 3 4 5 6 7	SpW_X_Sout+ SpW_X_Sout- Shield SpW_X_Sin+ SpW_X_Sin- Sheild sheild

VI. MODULE PHYSICAL

The Module physical size is based around the Single and Double Eurocard sizes (Fig. 12) and a non-standard "half height" module size. The mechanical and thermal interfaces specified will be elaborated in a future release of the standard.

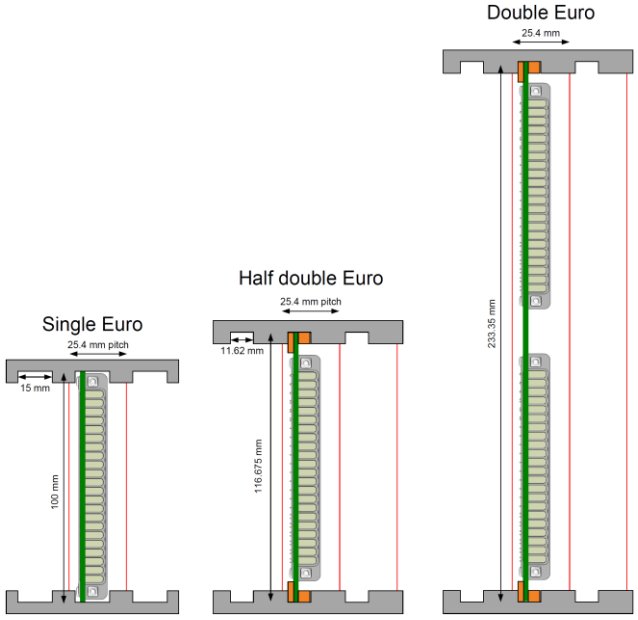

Fig. 12. Chassis and module slot details

VII. FUTURE STANDARD DEVELOPMENT

It is intended that standard will take advantage of specifications already existing, namely the ECSS and ANSI/VITA VPX standards. The principal VITA standards that are potentially applicable are VPX (VITA 46), OpenVPX (VITA 65) and a new draft standard SpaceVPX (VITA 78). At the time of writing the SpaceVPX standard is at an early draft and is not on general release.

It is also noted that additional details will need to be added to this ECSS standard to specify the tailoring of the VPX specifications for the SpW Backplane application, this tailoring being based on real needs, mechanical and thermal design, analysis and testing.

VIII. KEY ADVANTAGES OF A SPW BACKPLANE

Employing a SpW backplane within an Avionic unit has the following key advantages:

- SpW is a well specified and supported standard
- Common electrical interface for all modules
- Compatibility between vendors
- Reduction of interconnection count to a board
- Support devices available (Router etc.)
- IP Cores are available (SpW, RMAP)
- Test equipment is readily available and well supported
- Software interface specification is easier…
- It permits a scalable architecture
- High rate data transfer rates (compared to Mil-Std-1553, UART, CAN, SPI, RS422 etc)

REFERENCES

[1] ECSS-E-ST-50-12C, "SpaceWire – Links, nodes, routers and networks", Issue 2, 31st July 2008

[2] A. Senior, W. Gasti, O. Emam, T. Jorden, R. Knowelden, S. Fowell, "Modular Architecture for Robust Computation", International SpaceWire Conference 2008

[3] A. Senior, P. Worsfold, "A SpaceWire Active Backplane Specification for Space Systems", International SpaceWire Conference 2010

[4] A. Senior, K. Boxshall, J. Ilstad, S. Sharma, "A Modular Connector for High Speed Backplanes", ESA Space Passive Components Days 2013

Integrating STAR-Dundee SpaceFibre Codec with TI TLK2711

Onboard Equipment and Software, Short Paper

Bruce Yu, Alberto Gonzalez-Villafranca, Albert Ferrer
STAR-Dundee Ltd
STAR House, 166 Nethergate
Dundee, DD1 4EE, UK
bruce.yu@star-dundee.com

Chris McClements, Steve Parkes
Space Technology Centre, University of Dundee,
Dundee, DD1 4EE, UK
sparkes@computing.dundee.ac.uk

***Abstract*—The SpaceFibre Codec IP (beta version) was released by STAR-Dundee at the end of 2013. The SpaceFibre standard and the codec IP are designed in the way that it shall work with TI TLK2711-SP – a space qualified SERDES device [1]. This paper presents the work where the Codec IP and the TLK2711 are used to implement a SpaceFibre link. Firstly the SpaceFibre Codec and the TLK2711 device are introduced, especially the power-on reset and signal detection operations of the TLK2711 for they are fundamental for the SpaceFibre link. Experiments on link initialisation are presented with results and analysis.**

***Index Terms*— SpaceFibre Codec IP, TLK2711, SpaceFibre Link, Link Initialisation**

I. Introduction

SpaceFibre is a very high speed serial communications link which is being designed for use on board spacecraft. As SpaceFibre is compatible with SpaceWire at packet level, a SpaceFibre link can transfer a SpaceWire packet but at a much higher speed. It also provides a broadcast mechanism similar to SpaceWire time-codes but offering much more capability. SpaceFibre is a complementary technology to the currently popular SpaceWire, and applications developed for SpaceWire can be readily transferred to SpaceFibre.

SpaceFibre is designed to have a link speed of 2.5 Gigabits per second, as is achievable with current space qualified technology. It is possible to reach even higher speed, 20 Gigabits per second, with future technology, and multi-laning. Beside the high performance in speed, SpaceFibre has more worthy features, such as low latency, integrated Quality of service (QoS), and integrated FDIR capabilities.

A SpaceFibre Codec VHDL IP core has been developed at STAR-Dundee to evaluate and validate the SpaceFibre standard. A beta version of the Codec IP core was released around the end of 2013. The Codec IP is able to operate with an external SerDes device with minimal glue logics, including the Texas Instruments TLK2711-SP Wizard Link device. Together with a Microsemi Rad-Tolerant RTAX-2000 device, the Codec IP and the TLK2711-SP are ready to build a flight qualified SpaceFibre System.

TLK2711-SP is a Space qualified component, with flight heritages. Its commercial counterpart is TLK2711A, and they are functionally equivalent.

For the HPPDSP (High Processing Power Digital Signal Processor) project, high-speed data I/O interfaces are desired for which the SpaceFibre technology is a perfect fit. Designed for this project, the prototyping board is equipped with a Xilinx Virtex-4 FPGA and three TLK2711A devices, which are used for the implementations of three SpaceFibre interfaces. The STAR-Dundee SpaceFibre Codec IP has been successfully implemented on the FPGA, connected to the TLK2711A devices. Each of the interfaces has a number of virtual channels.

This paper firstly introduces the IP core and the TLK2711 device. Then the integration design is presented. Finally some experimental results are given and analysed.

II. STAR-Dundee SpaceFibre Codec IP

The STAR-Dundee SpaceFibre Codec IP core was developed as part of the standard development for its evaluation and validation. It is in the form of VHDL source codes, and it is highly configurable giving flexibility through generics, such as the number of virtual channels.

The CODEC is organized in layers that are defined in the standard, with interfaces between each layer. It doesn't include the physical and serialisation layers. For the encoding layer, it can be configured to include or exclude the 8B10B encoding/decoding module, or more specifically to have a special interface to the TLK2711 device. This enables the IP core with capability to connect with different technologies.

SpaceFibre Codec can transmit and receive SpaceWire packets encapsulated within Virtual Channel data frames, and also Broadcast frames and control words used to provide the QoS and the FDIR capabilities. This information is passed to the SpaceFibre interface via the Virtual Channel interface, the Broadcast interface, and the Management interface as shown in Fig. 1.

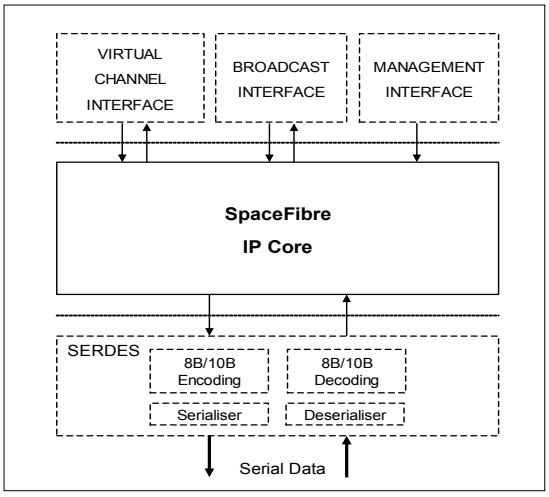

Fig. 1. Overview of STAR-Dundee SpaceFibre Codec IP Core

III. TI TLK2711 DEVICE

The TI TLK2711-SP Wizard Link device is a Space qualified multigigabit transceiver. The device contains both a transmitter and receiver, performing parallel-to-serial and serial-to-parallel data conversion. This device offers data rates from 1.28 to 2.0 Gigabits/s (at a link speed of 1.6 to 2.5 Gigabits/s).

The transmitter takes in 16-bit wide serial data, encodes it using 8B/10B encoding and serialises it for transmission over a VML differential signal pair. The receiver takes the serial data, de-serialises it, and performs 8B/10B decoding to provide the 16-bit parallel data.

A. TLK2711 Transmitter

The parallel data input to the transmitter comprises two bytes of data (TXD0-7 and TXD8-15) along with two control/data flags (TKLSB and TKMSB respectively). The control/data flags are high when the corresponding data byte contains a control code (K-code) and low when it contains data. The two data bytes and the control/data flags are latched into an 18-bit register on the rising edge of the TXCLK signal.

The TXCLK signal must be a continuous clock with a frequency in the range 80 to 125 MHz. It drives most of the transmitter circuits. There is a clock synthesiser which multiplies up TXCLK by 20 to provide the clock to drive the parallel to serial converter. The clock synthesiser also provides a reference clock for the clock recovery circuitry in the receiver.

To mitigate signal degradation on copper transmission media, two levels of pre-emphasis may be selected using the PRE input. When low the pre-emphasis is 5%, when high it is 20%.

The ENABLE signal is normally asserted to allow the TLK2711 device to operate. When de-asserted, the device is put in a power down mode with substantially reduced power consumption, as only signal detection circuit is active which draws less than 15 mW. In the power down mode, the serial transmit pins (TXN), the receiver data bus pins (RXD0-15) and RKLSB are tri-stated. But TXCLK clock still needs to be provided in power-down mode.

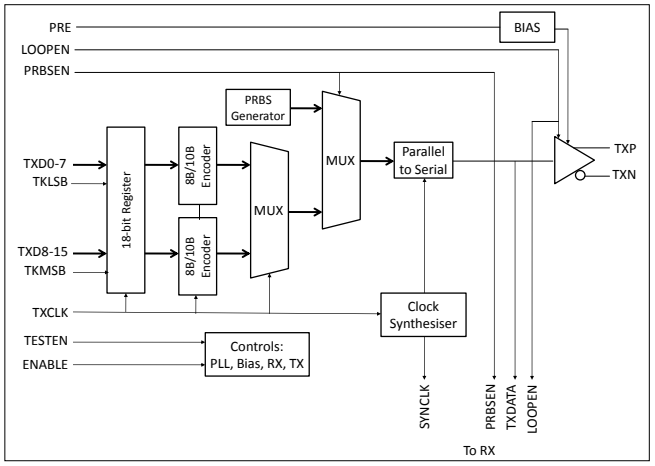

Fig. 2. TLK2711 Transmitter Block Diagram

B. TLK2711 Receiver

The received signal is fed via a pair of multiplexers to a serial to parallel convertor and to an interpolator and clock recovery block. The interpolator and clock recovery block recovers the received clock, to provide bit and word synchronisation. Bit synchronisation is achieved using a phase locked-loop (PLL) that takes the transmit bit clock from the transmitter (SYNCLK) as a reference and provides an output frequency locked to the transitions on the received serial bit stream.

The serial data is converted to a correctly aligned pair of 10-bit codes. The two 10-bit codes are decoded by a pair of 8B/10B decoders, each providing an 8-bit data byte and a control/data flag RKMSB and RKLSB). These signals are registered in an 18-bit register.

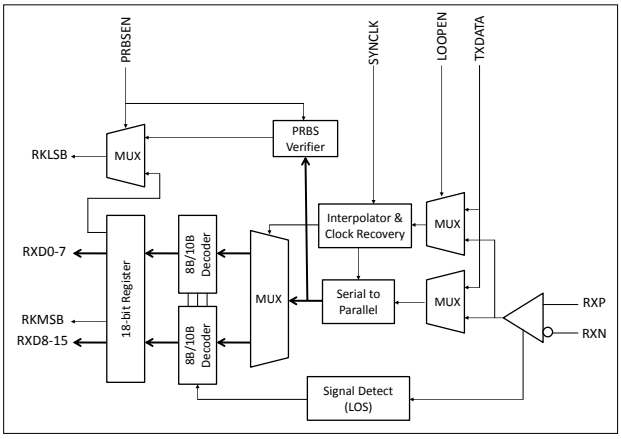

Fig. 3. TLK2711 Receiver Block Diagram

C. Power-on Reset

Upon application of minimum valid power, the device goes through a power-on reset process. When ENABLE pin is asserted high from a power-down mode, the device also goes

into power-on reset process before the normal operation begins.

During power-on reset, RXCLK is held low; the receiver data bus pins (RXD0-15) and RKLSB/RKMSB are in high-impedance state; the serial transmit pins (TXP/TXN) are high impedance as well.

The length of the process depends on the TXCLK frequency, but is less than 1 ms.

D. Loss of Signal (LoS) Detection

Loss-of-Signal detection is intended to be an indication of error conditions like a detached cable or no signal being transmitted, where the incoming signal no longer has sufficient voltage amplitude to keep the clock recovery circuit in lock. When loss of signal is detected the RXD0-15, RKMSB and RKLSB signals are all set high. This represents an invalid K-code on both bytes so can be safely decoded to mean loss of signal.

In power-down mode, the signal detection circuit is still active, and the RKMSB pin indicates the presence or otherwise of a signal on the receiver inputs. This can be used to provide an auto-start capability on a bi-directional serial link (similar to that used for SpaceWire). To save power when there is no data to send or to provide warm redundancy, the link can be put in the power down mode (ENABLE de-asserted).

This signal detection circuit enables the auto-start capability of a SpaceFibre link. When one end of the link has data to send it can enable its TLK2711 device and start sending data. The other end of the link, in power-down mode, detects that there is now a signal on the receiver inputs (RKMSB goes HIGH indicating that there is no longer loss-of-signal). The TLK2711 device at that end of the link can then be enabled and the link begins normal operation.

IV. SpaceFibre Interfaces on HPPDSP

HPPDSP project requires the I/O data interfaces having a very high speed, for which the SpaceFibre technology has been adopted. There are three SpaceFibre ports on a HPPDSP prototyping board. For each port, there is a double-deck eSATA connector as shown in Fig. 4. The upper deck is connected to MGT RocketIO on the Xilinx Virtex-4 FPGA. The lower deck is connected to a TLK2711 device. For this project, the lower decks are in use.

Fig. 4. Picture of SpaceFibre Ports on HPPDSP Unit

For each SpaceFibre interface, there are a number of virtual channels (VC), for instance four VCs (VC0 – VC3). The VC0, connected to a RMAP Target (and a RMAP initiator on one interface), is used to access the Configuration Bus on the FPGA design for configuration and control purpose. The VC1, VC2 and VC3 are connected to the IO DMA bus for data I/O transmission at high speed.

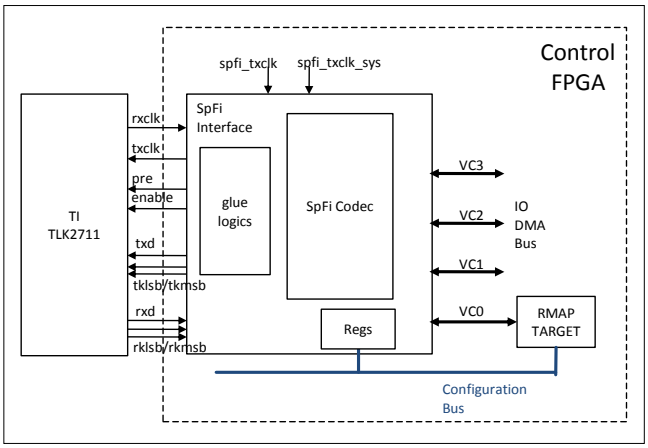

Fig. 5. Block Diagram of SpaceFibre Interface on HPPDSP

V. Experiments

Two HPPDSP prototyping boards are used for the experiments. The SpaceFibre interfaces can be configured to "Disabled", "Start", or "Auto-Start", using the Configuration Bus.

A. Loss of Signal (LoS) Detection

This experiment checks the operations of the signal detection circuit, under circumstances of forced no-signal and forced signal on the link.

The no-signal scenario is simulated with cable unplugged. When the TLK2711a device is enabled, the receiver outputs K31.7 on both MSB and LSB. When the device is disabled, the RK_MSB is set low.

After the rising edge of the ENABLE pin, the outputs by the receiver are not reflecting the true state, as shown in Fig. 6. One can see a short false-positive pulse on the signal detection, which is about 6.2 us. It is rational to conclude this is due to the power-on reset process.

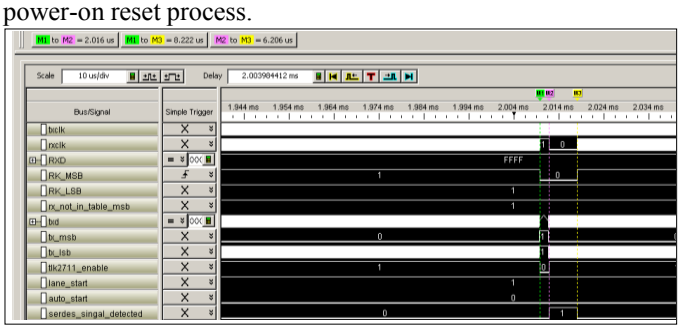

Fig. 6. Signal Detection with Forced No-signal Scenario

The forced signal scenario is simulated with the other end of the link set to "Start". When the TLK2711a device is enabled, the receiver doesn't outputs K31.7 on both MSB and LSB. When the device is disabled, the RK_MSB is set high. After the falling edge of the ENABLE pin, one can see a short low pulse on the signal detection in Fig. 7. This is because the simulated scenario is not perfect, and for that period the other end of the link was going through a reset cycle.

Fig. 7. Signal Detection with Forced Signal Scenario

B. Link Initialisation

An eSATA cable connects a SpaceFibre interface on one board to a SpaceFibre interface on the other board. This experiment has six different test cases. For each one, the same test is carried out three times. The time taken for this end to connect is recorded.

TABLE I. TEST CASES FOR EXPERIMENTS ON LINK INITIALISATION

Case	*This End*	*Remote End*	*Time to Connect*
1	Then Start	Started	~40us
2	Then Auto-Start	Started	~40us
3	Started	Then Start	~35us after receiving first signal
4	Started	Then Auto-Start	~35us after receiving first signal
5	Then Start	Auto-Started	~53us
6	Auto-Started	Then Start	~53us after receiving first signal

Test Case 1 and Test Case 2 are essentially the same test. Test Case 3 and Test Case 4 are essentially the same test. When one end is started, it tries to connect so it sends signal which can be picked up by the other end. Therefore "Then Start" and "Then Auto-Start" are not making any difference.

For Test Case 1, before this end is set to "Start", signal has been detected on the link. As soon as this end is set to "Start" at marker M1 in Fig. 8, the TLK2711 device is enabled.

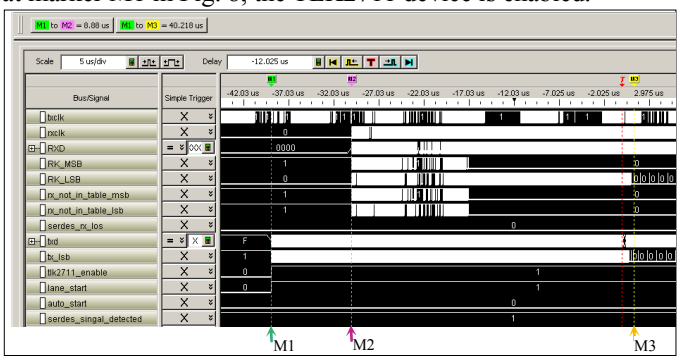

Fig. 8. Link Initialisation under Test Case 1

Then the power-on reset is in process, until there is the RXCLK at marker M2 in Fig. 8. Following that, it takes some further time for the device to synchronise and lock with the incoming serial data and to find a comma to align on the word boundary. Then it takes about 18us for the SpaceFibre Lane initialisation state machine to go through various states and get connected. In Fig. 8, the point that the link is connected is at maker M3, where it starts to send out IDLE frames.

For Test Case 3, this end has been started, and the TLK2711 device has been enabled. It is at marker M1 in Fig. 9 when the first signal is received. After about 16us, the device synchronised and locked with the incoming serial data and found a comma to align on the word boundary. Then it takes another about 18us for the link to get connected at maker M3 where it starts to send out IDLE frames.

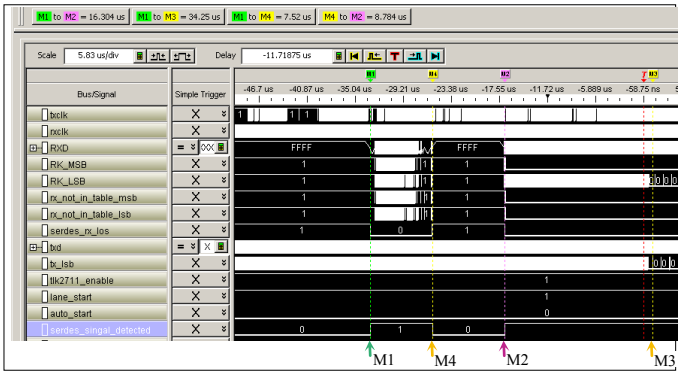

Fig. 9. Link Initialisation under Test Case 3

For Test Case 6, this end has been set to Auto-Start. The TLK2711 device is enabled as soon as signal detected on the link that is at marker M1 in Fig. 10. After going through the power-on reset process, at yellow marker M4, it detects a LoS and therefore the TLK2711 device is disabled. At marker M2, signal is detected again and so the TLK2711 device is enabled. Then similarly it takes 16us plus 18us for the link to get connected at maker M3. The reason for the LoS detected may due to the remote end was in the power-on reset process.

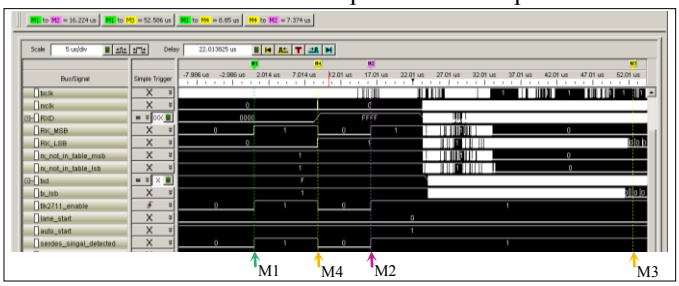

Fig. 10. Link Initialisation under Test Case 6

ACKNOWLEDGMENT

The authors would like to acknowledge the support of ESA for the HPPDSP project (ESA contract number 4000102660), which contributed to the work described in this paper.

REFERENCES

[1] S. Parkes, and C. McClements, "SPACEFIBRE CODEC: USE OF THE TLK2711-SP," Proceedings of the 4th International SpaceWire Conference, San Antonio, 2011

[2] S. Parkes, A. Ferrer, A. Gonzalez, and C. McClements, "SpaceFibre Specification Draft F1", May 2013

[3] Texas Instruments, "TLK2711-SP Data Sheet: 1.6-Gbps to 2.5-Gbps Class V Transceiver", Reference Number SGLS307D, July 2006, Revised July 2009.

MARC – Lessons Learnt

SpaceWire Onboard Equipment and Software, Short Paper

Alan Senior, John-Paul Coetzee
Space Division
Thales Alenia Space - UK
Bristol, United Kingdom
alan.senior@sea.co.uk, john-paul.coetzee@sea.co.uk

Wahida Gasti
European Space Agency, ESTEC
Noordwijk ZH, Netherlands
wahida.gasti@esa.in

***Abstract*— This paper describes the lessons learnt during the development and testing of the Modular Architecture for Robust Computing (MARC) demonstration system. It is principally written from a hardware perspective.**

The MARC system is designed for satellite avionics applications. The network and power architectures are based on established spacecraft redundancy concepts and provide tolerance to single point failures. The MARC architecture is designed to provide a scalable solution that can meet the demanding needs of future missions, the SpaceWire network can be expanded to include new functions and to provide duplicate paths to achieve the level of redundancy needed for a particular mission.

An important aspect of the demonstrator hardware is that the key components are space qualifiable parts; permitting the design to be upgraded to a fully space qualified system with minimal changes, in particular the hardware design uses the ESA Atmel AT697F processor and SpaceWire 10X router developments. The ESA SpaceWire RMAP IP Core is also used for all module network interfaces, being implemented within FPGAs.

The lessons learnt include the experiences with implementing the RMAP IP Core, VHDL synthesis problems, power consumption issues and the need for detailed internal unit interface specifications. Additional technology developments, such as radiation and fault tolerant Point of Load converters that are required for migration of the design to flight are also identified.

Lessons were also learnt regarding parallel Hardware and Software developments to reduce development timescales whilst eliminating diverging design compatibility.

***Index Terms*— SpaceWire, SpW, MARC, Avionics, RMAP.**

I. Introduction

The Modular Architecture for Robust Computing (MARC) system (Fig.1) is an innovative hardware development that unifies future spacecraft processing system requirements to create a SpaceWire (SpW) [1] network based scalable, fault tolerant, high performance capability and robust system solution suitable for both Spacecraft platform and data handling applications [2].

The design comprises a SpaceWire Active Backplane and a set of plug-in Modules with SpaceWire interfaces; this scalable architecture permits optimisation of the system to suit different applications.

The developed demonstration system uses many recently developed European technologies such as the SpaceWire RMAP IP Core the LEON2FT processor and the SpaceWire 10X Router as well as commercial technologies such as DDR and FLASH within the Mass Memory Module. An important aspect of the electronics design is the use of technologies that have a component level route to a radiation tolerant flight system.

Fig. 1. MARC Demonstrator

The potential applications for MARC are extremely broad, encompassing single spacecraft with modest platform and instrument requirements, to high data rate instruments mounted on multiple formation-flying spacecraft.

Avionics applications include:

- Platform command and control (with SpW replacing legacy Mil-Std-1553)
- Science spacecraft data handling and payload processing
- Deep space missions requiring a high level of autonomy
- Planetary robotic systems
- Complex and/or multiple payloads

The project has completed the hardware build and test phase and the MARC Demonstrator is ready to be used for the development of flight software. The MARC hardware is designed to support software services based on the Spacecraft Onboard Interface Services standards (SOIS).

The SpaceWire network architecture building block is shown in Fig. 2. , this is called a "Cluster". The 8 Port Routers were implemented with the Atmel AT7910 device. The Module SpaceWire interfaces are implemented with the ESA developed RMAP SpW IP core within a Microsemi ProASIC 3 FPGA. TAS-UK was an Alpha tester of the RMAP IP Core.

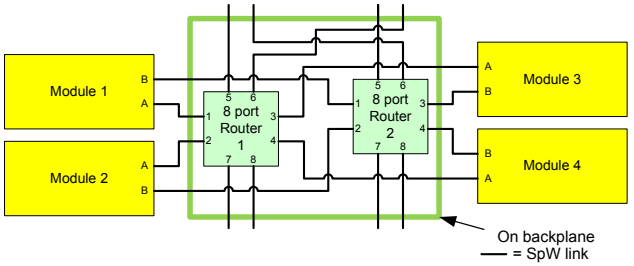

Fig. 2. Network Cluster comprising 2 Routers and 4 Modules

II. SpaceWire RT

The MARC hardware supports the SpaceWire Real-Time (RT) protocol to the following extent:

- Non-RMAP SpW RT packets bypass the RMAP IP Core on the Core Computing Module (CCM)
- SpW RT packets can be exchanged with another CCM
- Other modules are RMAP compatible but are not "time slot" aware however this is not an issue as they are slaves in the demonstrator
- The CCM software manages all transfers and respects time slots
- CCM FPGA incorporates a SpW RT packet transmit and receive buffer
- Hardware flags have been added added to indicate the SpW RT message has been sent

The SpW RT draft protocol was implemented in software, this consumed a significant percentage of the processing budget and reduced the responsiveness of the system.

Lesson learnt: the functions required to implement complex SpW protocols should be partitioned into the hardware and software domains to increase system performance.

III. SpW and RMAP IP Core

The ESA SpW IP Core was available prior to the start of the MARC project and a compatible RMAP IP Core was under development. TAS-UK reviewed the specification for the RMAP IP Core and noted that when mated with the SpW IP Core other packet protocols were not supported, this limitation was not compatible with the MARC system design. One option open to TAS-UK was to design a "protocol sorter" that could be inserted between the two IP cores so that non-RMAP packets could be routed through the same SpW interface. Fortunately the ESA RMAP IP Core development was at a stage where it could be updated to permit other protocols to be used.

Lesson learnt: Protocol IP Core extensions to the SpW IP Core should not preclude the use of other protocols.

IV. SpW IP Core integration into the ProASIC FPGA

Integration of the RMAP IP Core into the ProASIC 3 FPGA with the support logic for the processor and other interfaces was relatively straightforward initially. Meeting the timing requirements to achieve a 200Mbps speed however was non-trivial and required many synthesis iterations with adjustments to the placement.

Lesson learnt: Choose the FPGA I/O pins and verify the placed design meets the timing constraints before allocating pins at schematic level. Choosing I/O pins in physical proximity to the clock pin resources can assist in meeting high performance requirements.

V. RMAP Initiator Error on Timeout

When the MARC system is operating normally SpaceWire RMAP messages are initiated and replies are received within the RMAP Initiator watchdog time-out period and the system operated reliably.

During the MARC system Failure Detection Isolation and Recovery (FDIR) testing random SpaceWire nodes attached to the active backplane are de-powered, in this situation an RMAP command to that node was routed through the backplane network and then be blocked, since there is no packet sink. In this situation, it was found that no further RMAP messages could be initiated by the sending node and the MARC system had to be power cycled to recover. Clearly the failure of the system to recover was not acceptable.

Lengthy software investigations ensued to demonstrate that that it was definitely the hardware that was not behaving correctly. The RMAP Initiator was configured to support 36 outstanding transactions, however during the software tests it was noted that up to 56 outstanding transactions could be reported.

The RMAP IP core has been subjected to both simulation and hardware testing at Star Dundee. This involved the initiation of RMAP transactions with both no timeout and a fixed timeout without any anomalous behaviour.

At TAS-UK the same test bench command script was modified to initiate 36 transactions with infinite timeout. The bug then manifested itself as a functional error in a back-annotated gate-level simulation with no apparent associated timing problems. The fault signature changed almost on a per-synthesis basis suggesting that there were timing violations that were not being checked or reported. It is believed that the complex initiator processes and enumerated type state machines were not being correctly synthesized despite being written in valid VHDL, thus it was concluded that the Synplify synthesizer being used by TAS-UK had an undocumented bug. The RMAP IP Core design was produced by STAR Dundee

using the Mentor Graphics tool suite which employs a different synthesizer and it compiled the design correctly.

The Mentor Graphics tool suite was not available to TAS-UK, so the complete MARC VHDL code design for the Processor Module was released to the STAR Dundee to investigate and perform a re-build on behalf of TAS-UK. This recompiled design operated correctly in simulations and the MARC Demonstrator then correctly recovered in the FDIR test scenario. The final design achieved 200Mbps SpW data rates reliably.

Lessons learnt: Use the same design and synthesis tools as the IP Core creator to avoid possible synthesis bugs. Ensure the test benches check all all corner and boundary cases for the particular IP Core configuration used. Ideally IP Cores designed for Space applications should be validated using the different synthesis tools commonly used for Space electronics design.

VI. Power dissipation

The SpW active backplane used four AT7910 8 port routers these were connected in a network topology that provided 28 point to point SpW link at 200Mbps. The power consumption of the backplane was approximately 15 Watts, this included all of the power supply regulators and support devices such as buffers and oscillators.

Lesson learnt: A rule of thumb based for this particular setup is that each SpW link has an associated dissipation of approximately 0.5W at 200Mbps.

VII. LEON2FT Maximum Clock Rate

The AT697F data sheet cites a maximum processor clock rate of 100MHz. A worst case analysis indicated that 0ns access time RAM would be needed to operate with zero wait states. The MARC system was therefore operated with an 80MHz processor clock rate.

Lesson learnt: Data sheet clock rates are not always achievable in all conditions in a real system when the worst case is considered.

VIII. POL converters and supply voltages

The supply voltage required by integrated circuits is dependent on the semiconductor technology and device characteristics. Typical voltages for digital devices are 5V, 3.3V, 2.5V and 1.8V. The trend towards higher speed parts has led to smaller semiconductor feature sizes, the thinner oxide layers employed have lower breakdown voltage and hence lower supply voltages are employed. The high speed and high transistor count of these modern devices leads to a high current requirement at relatively low supply voltages.

Providing all the different supply voltages at the currents required whilst maintaining the voltages within the required tolerance at the device pins is not trivial. Linear regulators may be used but are inefficient if the input and output voltages are significantly different. As an example generating a 2.5V regulated supply at 1A with a 5V input leads to a dissipation of 2.5W in the regulator.

The solution is to supply each module with a higher voltage and use Point of Load (POL) DC-DC converters onboard the module. The MARC system uses a 24V backplane bus to keep the backplane voltage drops to an acceptable level. POLs however are not commonly available to generate voltages lower than 5V from a 24V input supply. Typically the available terrestrial and space qualified POLs operate with a 5V input.

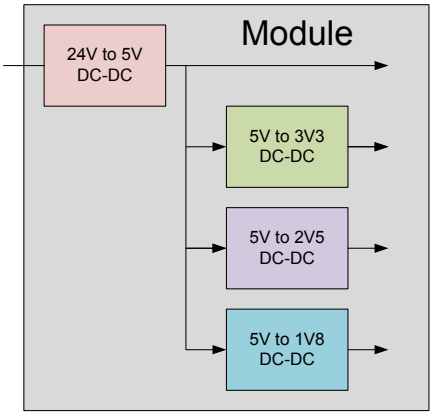

Fig. 3. Module power supply voltage generation

Lesson learnt: Allow for 2 stages of DC-DC conversion on the Module, for example 24V to 5V and then 5V to 1.8V.

IX. Test and Integration

The SpW active backplane [3] interface for each Module consisted of two SpW links and a single power rail. The lack of bespoke interfaces at the backplane interface and the ability to connect EGSE to spare ports of the backplane SpW network permitted module level debugging using available off the shelf SpW test equipment. Modules in development could be emulated easily and the system status could be monitored without creating a bespoke hardware test environment.

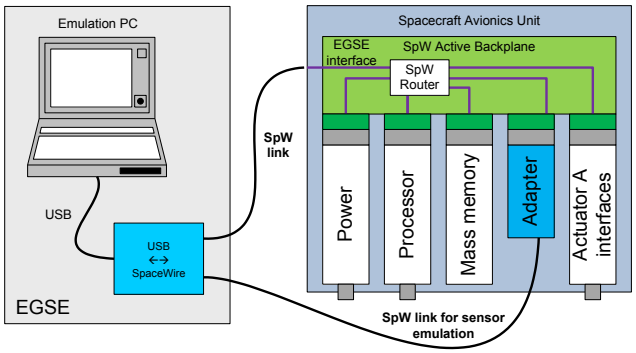

Fig. 4. A SpW backplane simplified MARC integration and test activities

The use of SpW at the backplane interface permitted a PowerPC module supplied by another vendor to be integrated into the SpW network without difficulty.

Lessons learnt: A SpW backplane has significant advantages during integration and test activities, simplifying module test and system level debugging.

X. PARALLEL HW AND SW DEVELOPMENTS

The hardware and software development activities for MARC overlapped to shorten the complete system development schedule. The software was initially developed on a RASTA system until the MARC hardware demonstrator became available. When the software was integrated with MARC it was found that there were significant incompatibilities between the hardware and software, this resulted in an extended software development schedule.

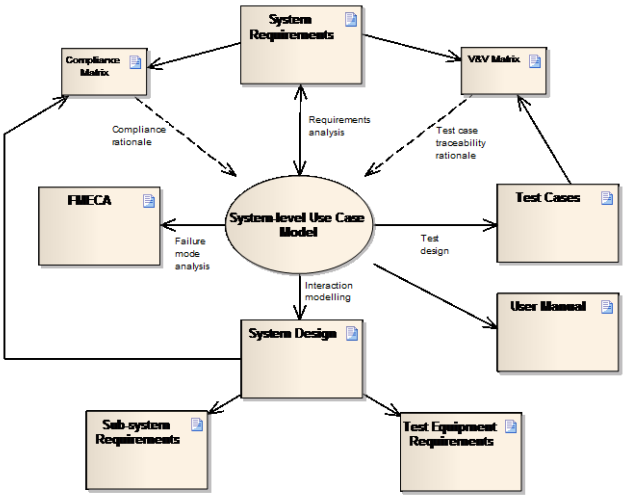

Fig. 5. Role of system-level Use Case Model

There were a number of reasons for this situation arising:

- unclear and misunderstood requirements
- few team members have a full end-to-end understanding of what the system does and why
- too much focus on sub-system design and physical interfaces in a way that is divorced from the top level requirements
- the view that "Functionality is a software implementation detail"

This situation could have been alleviated by using 'Use Case Models' which identify 'the system' and the 'actors' (roles played by users and external systems).

A 'Use Case' [4] describes the behaviour required of the system to achieve a particular user goal in a story-like narrative structure that effectively communicates system vision in scope and detail. The key advantage of this style is that the system behavior can be understood by software, hardware and "non-engineering" team members.

The Use Case Model may also be used for behavioural analysis to refine the requirements, highlight inconsistencies and to identify failure modes.

Lesson learnt: Produce Use Case Models to assist in the development of compatible hardware and software for complex systems.

[1] ECSS-E-ST-50-12C, (SpaceWire – Links, nodes, routers and networks, Issue 2, 31st July 2008

[2] A. Senior, W. Gasti, O. Emam, T. Jorden, R. Knowelden, S. Fowell, "Modular Architecture for Robust Computation", International SpaceWire Conference 2008

[3] A. Senior, P. Worsfold, "A SpaceWire Active Backplane Specification for Space Systems", International SpaceWire Conference 2010

[4] Alistair Cockburn, "Writing Effective Use Cases", Addison-Wesley, 2000.

An RTEMS Port for the AT6981 SpaceWire-Enabled Processor : Features and Performance

Onboard Equipment and Software, Short Paper

David Paterson
STAR-Dundee Ltd.
Dundee, UK
david.paterson@star-dundee.com

David Gibson, Steve Parkes
Space Technology Centre
School of Computing
University of Dundee
Dundee, UK
davidgibson@computing.dundee.ac.uk,
sparkes@computing.dundee.ac.uk

***Abstract*— The Atmel AT6981 is a complex system-on-chip based on a SPARC LEON2-FT core, and which provides a number of peripheral devices including three multi-function SpaceWire engines and a router.**

The RTEMS real-time operating system is widely used in spacecraft systems in many roles. Its long history and open source availability make it an ideal choice for many applications. RTEMS has already been ported to many platforms, including some based on the SPARC LEON2 processor.

The process of porting RTEMS to the AT6981 is described, and the performance, both for general data processing and for SpaceWire traffic handling, is examined.

***Index Terms*— Relevant indexing terms: SpaceWire, Spacecraft Electronics, Real-Time Operating System, RTEMS.**

I. Introduction

The requirements for spacecraft on-board data handling are continually increasing in terms of demands on both processing power and network bandwidth. This has driven the development of ever more powerful and capable data processors and network controllers.

The Atmel AT6981 [1] combines a high-performance, fault-tolerant processor with multiple SpaceWire engines and a SpaceWire router, providing both data processing and network control in a single package. The inclusion of on-chip memory and a range of other peripherals and network interfaces make it a highly capable device, suitable for use in a wide range of applications.

Along with the requirements for increased processing capabilities, there is also a need for a reliable software environment to support real-time scheduling of the data handling tasks. The RTEMS operating system is an ideal candidate for this role, having proven its reliability and usefulness in use on many missions, as well as having been widely adopted in non-spaceflight applications.

Although RTEMS has been ported to LEON2-based platforms, each target system has a different configuration, so an AT6981-specific port is required in order to make full use of the device's capabilities. Porting RTEMS essentially requires the development of a target-specific Board Support Package (BSP) together with additional device drivers for the target's peripherals, and these are integrated into the RTEMS source tree in order to build the target-specific version.

II. The AT6981 System-On-Chip

Based on a SPARC V8 LEON2-FT processor running at 200 MHz, the AT6981 is ideally suited for SpaceWire-based applications with the inclusion of three powerful and flexible SpaceWire engines. Each engine contains an RMAP initiator, RMAP target and three general-purpose transmit/receive DMA channels. These SpaceWire engines are connected to a SpaceWire router which has eight external ports, providing extensive network connectivity.

Additionally, the AT6981 includes up to 1 MByte of on-chip EDAC-protected SRAM, controllers for CAN, MIL-STD-1553 and Ethernet, as well as general purpose I/O, UARTs, timers and other commonly-required interfaces.

A diagram of the AT6981 is shown below in Figure 1.

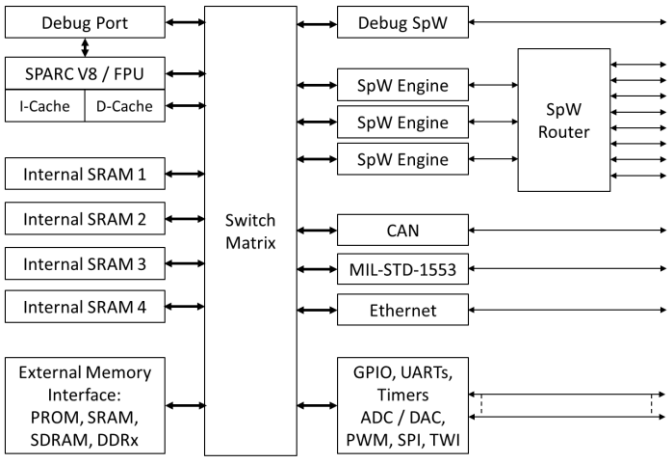

Fig. 1 – AT6981 Functional Block Diagram

The SpaceWire subsystem of the AT6981 is built around three highly capable, semi-autonomous engines which can offload much of the work involved in sending and receiving SpaceWire traffic from the main processor.

A diagram of the SpaceWire subsystem is shown below in Figure 2.

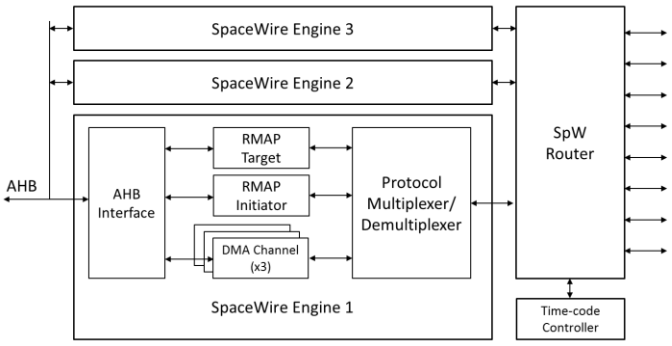

Fig. 2 – AT6981 SpaceWire Subsystem

The SpaceWire router has three internal ports and eight external ports connected through a switch matrix which allows multiple simultaneous connections between inputs and outputs.

The router is also connected to a time-code controller which indicates received time-codes, and can generate time-codes based on internal counters, hardware interrupts or on command from the processor. The time-code controller also handles and can generate distributed interrupts.

For packets being transmitted, the protocol multiplexer/ demultiplexer selects packets to be sent to the router, using a fair arbitration scheme. For received packets, the first four bytes of each packet are checked against configurable patterns and masks to determine the correct destination – RMAP target, RMAP initiator, or one of the three DMA channels.

The RMAP target accepts RMAP commands from a remote system, performs read or write operations over the AHB bus to local memory, and optionally returns a reply packet. The target supports all RMAP commands, and includes a 16 byte buffer for verified write commands.

The RMAP initiator transmits RMAP commands to read or write memory or registers on a remote system, transferring data to or from local memory via the AHB bus. The initiator is controlled by a table of transaction requests stored in memory, allowing it to transmit multiple commands and validate replies to them without processor intervention.

The three DMA channels can each transmit and receive SpaceWire packets from or to local memory. Transmitted packets can consist of one or more data chunks, allowing for separate storage of packet headers, while received packets are stored contiguously in memory. As with the RMAP initiator, transmit and receive operations are controlled by configuration tables, minimising processor overhead.

The DMA channels can also transmit and receive RMAP [2] and PUS [3] packets, using hardware CRC-8 and CRC-16 computation respectively.

With three identical SpaceWire engines, and eight external ports from the router to access the spacecraft's on-board network, the AT6981 provides a very high level of capability for data handling. The ability of these engines to operate autonomously means that this is achieved with minimal load on the main processor.

III. The RTEMS Real-Time Operating System

The RTEMS operating system has been designed specifically for use in real-time embedded environments, providing a full range of essential support features for real-time software, including mission-critical and safety-critical applications.

RTEMS has been under continuous development since the late 1980's, and has evolved over that time into a highly reliable and capable system. It has been used in a wide range of application areas, such as networking, automotive, medical, hi-fi systems, particle accelerators and, most importantly, spacecraft systems [4].

Real-time systems are differentiated from other software applications by the requirement that they must respond to events within specified time constraints – "A real-time system is one whose logical correctness is based on both the correctness of the outputs and their timeliness." [5].

Real-time requirements may be divided into two broad categories :-

- Soft real-time – in which a missed deadline does not compromise the integrity of the system or result in a catastrophic event.
- Hard real-time – in which a missed deadline causes the work performed to have no value or to result in a catastrophic event.

RTEMS is designed to handle both of these types of constraint, and implements a number of different task scheduling options to allow for flexibility in system design, and for both hard and soft real-time tasks, and variations of them, to run in the same system.

The RTEMS system is structured using a layered approach, as show in Figure 3.

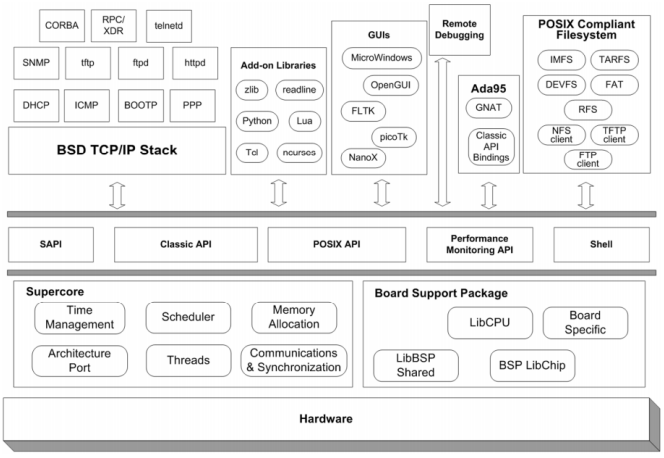

Fig. 3 – RTEMS Architecture

Although the architecture diagram shows a large number of components, the configuration mechanisms invoked when building an RTEMS system ensure that unused parts of the code base are not included in the final executable.

A number of APIs are available, including a POSIX compliant API supporting a large part of POSIX 1003.1b, such as process and thread creation and control functions and object types (semaphores, mutexes, condition variables etc.), file and directory management and memory management.

At the lowest level of the RTEMS architecture, the interface to the target hardware is managed through the Board Support Package, and this is discussed in more detail in the next section.

IV. Porting RTEMS to the AT6981

RTEMS is already in widespread use for many spaceflight applications, and is seen as a reliable and easy-to-use operating system environment for the implementation of flight software. Porting RTEMS to the AT6981 extends the range of devices which are supported, and provides a very capable hardware-software combination for many on-board data handling and communications applications.

As stated previously, RTEMS has been ported to the LEON2 in some basic configurations. However, in order to make full use of the features of the AT6981, a more complete, target-specific port was needed. This provides not only the basic integration of the processor into RTEMS, but also drivers for the built-in peripheral devices.

The first fundamental step in the process of porting RTEMS to any new target is identifying which components need to be developed, and which parts of existing ports, or parts of "standard RTEMS" can be used :-

- Does a BSP for this board exist?
- Does a BSP for a similar board exist?
- Is the board's CPU supported?

In this case, although the CPU (SPARC LEON2) is supported, no really similar board or device has yet been ported, so only some basic, shared interrupt handling code could be used. More of the common code from RTEMS, mainly related to system initialisation, could be included, but most of the BSP would have to be developed "from scratch" (albeit, based on the design and structure of other, similar BSPs).

For an initial, basic BSP, a small number of modules must be implemented :-

- Initialisation (board start-up)
- Clock driver
- Console driver
- Timer driver (optional)

The initialisation code is responsible for ensuring that the processor board is correctly initialised following a system power-on or reset. Registers and memory areas are set to known states, the stack is set up and interrupts cleared to ensure correct and reliable operation of the operating system and the application software.

The clock driver provides a reliable time reference to the RTEMS kernel, so that all primitives that require a clock tick work correctly.

The console driver, effectively a UART driver, is primarily for use in debugging and for system status report messages.

The timer driver is used by timing and benchmark tests, and although optional in the basic BSP, can be useful in determining system performance, and identifying areas which may need optimisation.

Once the basic BSP had been implemented and tested, confirming that RTEMS was operating correctly, the next step was to develop drivers for the peripheral devices on the AT6981, beginning with the SpaceWire subsystem.

In order to simplify the API for users of this feature, it was decided to write two separate device drivers, one for DMA channel management, and one for the RMAP targets and initiators. This separation also reflects the fact that these parts of each engine can operate independently.

The structure of an RTEMS device driver is relatively simple, and involves implementing a standard set of device operations – open, close, read, write and control, which map directly to the API functions typically available in most high-level language support libraries. Additionally, an initialisation function must be provided, and this is called during the RTEMS start-up sequence to carry out any device-specific initialisation which might be required.

At present, only the SpaceWire device drivers have been written, but additional drivers for other peripherals will be added in the future.

V. Performance

The testing of the RTEMS port was carried out on the STAR-Dundee AT6981 Prototype Card which contains an FPGA into which is programmed the LEON2 core, 128 KBytes of on-chip SRAM, the SpaceWire subsystem (as shown in Figure 2) and 256 MByte of DRAM. The AT6981 Prototype Card is shown in figure 4.

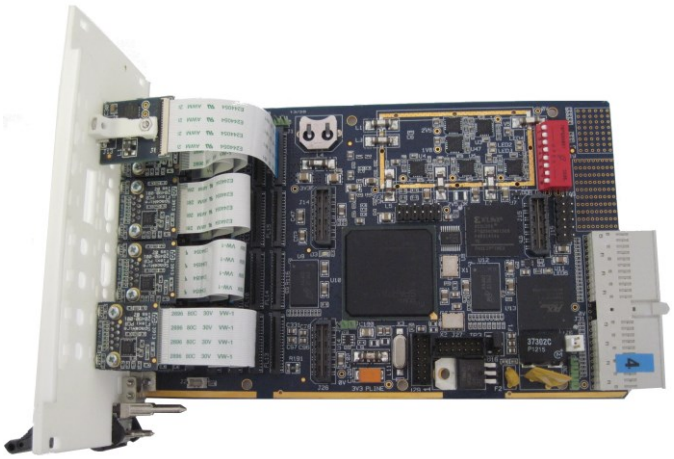

Fig.4 – STAR-Dundee AT6981 Prototype Card

The target clock speed for the production versions of the AT6981 will be 200 MHz, which should provide at least 150 MIPS Dhrystone performance, and at least 40 MFLOPS Whetstone performance. However, the Prototype card processor clock speed is limited to 30 MHz, so the measured performance is expected to be approximately one-sixth of the production device.

The SpaceWire clock on the prototype card runs at the full 200 MHz, so link speeds of up to 200 Mbit/s are supported.

Testing is still ongoing, but it should be possible to transmit and receive packets at the maximum data rate via all three SpaceWire engines simultaneously, provided they are routed through different external ports. The autonomous operation of the engines should require minimal processor overhead in handling these transactions, so processor performance is not expected to be a limiting factor in normal operation.

Final, measured performance figures will be given in the oral presentation of this paper.

VI. Conclusions

The AT6981 provides a high-performance system-on-chip solution to the ever-increasing demands for on-board data processing and network bandwidth. The flexibility and autonomous nature of the three SpaceWire engines allows for its use in a wide range of network configurations and operating modes.

RTEMS has already gained wide acceptance for use as an environment for spacecraft software, and porting RTEMS to the AT6981 extends the range of hardware which supports it. This will provide additional options to designers and developers of on-board data handling systems, providing a reliable platform on which to implement any required application software.

The open-source nature of RTEMS makes it relatively easy to configure, and to port to new target hardware. Although the AT6981 port of RTEMS currently provides only a basic BSP and drivers for the SpaceWire engines, additional drivers will be developed in the future, increasing the usability of this versatile hardware / software combination.

References

[1] “Atmel Space Rad-Hard Processors” Atmel-41005B-AEROSpaceRadHardProcessor_E_A4_052013.

[2] ECSS-E-ST-50-52C, “SpaceWire – Remote Memory Access Protocol”, European Cooperation for Space Data Standardization, February 2010

[3] ECSS-E-ST-70-41A “Ground Systems and Operations – Telemetry and Telecommand Packet Utilization”, European Cooperation for Space Data Standardization, January 2003

[4] “RTEMS Applications”, RTEMS Wiki - http://www.rtems.org/wiki/index.php/RTEMSApplications

[5] Phillip A. Laplante, “Real-Time Systems Design and Analysis”, Third Edition, John Wiley & Sons / IEEE Press, 2004

Test & Verification (Short)

MOST: Modeling of SpaceWire & SpaceFibre traffic

NOT PERMITTED TO PUBLISH PAPER

Automatic Performance Tracking of a SpaceWire Network

SpaceWire test and verification, Short Paper

Kai Stohlmann
Institute of Space Systems
German Aerospace Center
28359 Bremen, Germany
Kai.Stohlmann@dlr.de

Görschwin Fey
Institute of Space Systems
German Aerospace Center
28359 Bremen, Germany
Goerschwin.Fey@dlr.de

Daniel Lüdtke
Simulation and Software Technology
German Aerospace Center
38108 Braunschweig, Germany
Daniel.Luedtke@dlr.de

Abstract— Verification of complex networks, especially meshed networks created of routers, can become quite difficult. There are several parameters influencing the actual data throughput, e.g., congestion in the network, transmission rates at the inputs of the network or between routers as well as the reception rate of data. An appropriate model is required to evaluate the network performance. This model can be defined at different levels of detail whereas the more detailed levels are considered to be more precise with respect to the real hardware behavior. The objective of this paper is to define such a model and to provide random constraint stimuli for an automatic tracking of the network performance. The model consists of a meshed network with a fixed topology using HDL descriptions of the SpaceWire routers in our system. However, this approach can also be applied to other network topologies. Precisely mimicking input and output data streams that are applied to the network is crucial to identify inefficient data paths. These data streams are generated dependent on predefined high level constraints (e.g. packet lengths, transmission rates) that are transferred into lower level constraints to finally create the required stimulus. The quality of the system, dependent on a specific data stream, is determined among others by tracking and analyzing the transition time of packets from source to destination.

Index Terms— Network, Functional verification, Random constraint verification, Universal Verification Methodology (UVM)

I. Introduction

Accessing and monitoring signals or data traffic in embedded hardware can become very difficult but is often required. To address this problem the paper shows how monitoring/tracking of network traffic and related DUV (Device Under Verification) behavior is applied during simulation by use of a hardware verification language in combination with a random constraint verification approach. Since hardware debugging is usually harder than debugging based on simulation, we are creating a verification environment whenever possible. The OBC-NG prototype (On-board Computer - Next Generation) [1] which is developed by the German Aerospace Center will serve as the DUV. Because its network is heavily meshed it becomes even more complex and difficult to follow the data traffic and to monitor the behavior of the real system. These monitoring capabilities are important for debugging as soon as the actual data traffic differs from the expected data traffic that can be caused either by faulty DUV parts or by faulty interactions between DUV parts due to suboptimal configuration. Because the DUV might change its topology during development, it was intended to create a verification environment that scales with the used network topology.

The remainder of this paper is structured as follows. Chapter II shows the used DUV in more detail, which is important to understand the structure of the surrounding test environment. The following Chapter III introduces the concepts of a typical UVM (Universal Verification Methodology) [2] test environment that is responsible for driving the DUV dependent on user constraints. The related stimuli generation that is used inside the test environment is explained in Chapter IV. Performance and behavior tracking is presented in contrast to the test environment in Chapter V. Finally, Chapter VI gives some conclusions.

II. The device under Verification (DUV)

The OBC-NG prototype will be used as the DUV with a basic structure shown in Fig. 1. The prototype consists of four identical nodes that are able to perform the same tasks. The main idea is that faulty nodes can be replaced by any other node as well as load balancing can be performed in an optimal way which is described in detail in [1]. However, for this paper only the structure of the network is important.

Every node consists of a processing unit (PU0 to PU3) that is connected to a related SpaceWire router written in VHDL (R0 to R3). The router was designed according to the related standard [3]. The configuration of it, e.g. routing table entries, is performed by use of the RMAP protocol [4]. The router itself was independently tested by random constrained verification and can be considered as functional correct. All nodes are connected to each other via the routers.

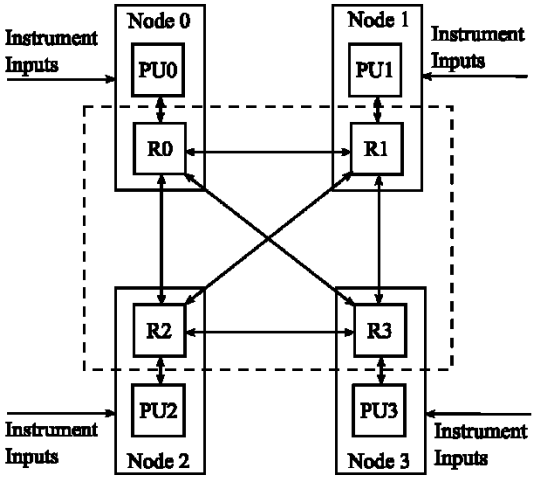

Fig. 1. DUV Structure

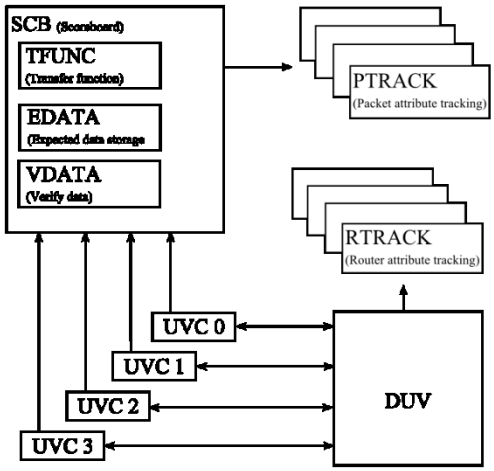

Fig. 2. Basic verification environment

Note there is only one connection at every single node to provide the processing units with data that might come from instruments or other subsystems. This is not a technical limitation of the system. The number of links can be increased. The processing units forward data through the routers to other nodes if required.

Hardware/software co-verification of the routers together with the respective PUs would slow down simulations drastically. Thus, it is reasonable to model data traffic between PUs and related router by random processes. The part of the system that fully simulated in the verification environment is enclosed in the dashed box shown in Fig. 1.

III. VERIFICATION ENVIRONMENT

Whenever a verification environment needs to be created it must be decided what test methodology should be followed and which language will be used.

If one considers hardware description languages (HDL) like VHDL (Very High Speed Integrated Circuit Hardware Description Language) or Verilog as insufficient for test environments, one can select a language that focuses on verification. An option is SystemVerilog, which is standardized [5] and supported by the most common simulation tools. Further, a framework called UVM based on SystemVerilog is developed by the main EDA vendors, provided for free and used for the test environment that is explained in the following. Reasons for us to select UVM were the object-orientated approach that increases the reusability or extension of existing verification components as well as the ability to drive the simulation by random constrained stimulus.

Fig. 2 shows the basic structure of the test environment that is used to drive the DUV and to track all necessary information related to data traffic and DUV behavior. The DUV is driven by a set of UVCs (Universal Verification Component). A UVC is attached to every interface that is provided by a DUV. As described before, the PUs are not embedded into the verification environment. Instead, the UVCs will drive the routers as shown in Fig. 3.

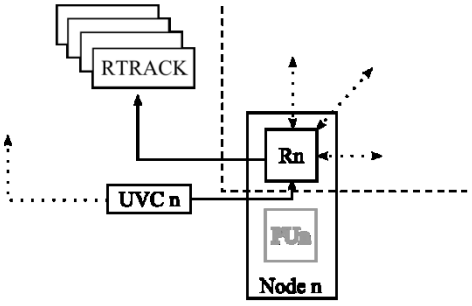

Fig. 3. UVC interaction with the DUV

Because the DUV consists of four nodes it is required to instantiate a single UVC for each router. During simulation, the UVC generates predefined SpaceWire packet objects to apply them sequentially to the DUV. The SCB (Scoreboard) shown in Fig. 2 provides functionality for comparing results or modifying data. For most packets that are transferred into the router an expected output packet is created by a transfer function inside TFUNC and stored inside EDATA. TFUNC also checks whether an expected packet needs to be created or not. E.g., in case of a routing table configuration with no response, a comparison of actual and expected data is not possible. However, if data is transferred from UVC0 over R0 to R3 to UVC3 it is possible to compare sent data at UVC0 with received data at UVC3. In case a UVC receives data, the

compare function inside VDATA is used to ensure that the packet was transferred correctly. PTRACK objects capture the performance properties of packets. Because these properties are tracked at every output, it is required to instantiate these module four times. The properties of the router are captured by the RTRACK objects, which have to be instantiated for every router separately. In case the topology changes, it is possible to add or remove the required amount of tracking modules.

IV. STIMULI GENERATION

As described in the previous chapter, UVCs are responsible for generating and applying data that is transferred over the network. This generation and randomization of data must be controlled in a way that it behaves as close as possible to the PUs in the real system. Additional, randomization control must be exchangeable for reusability and the application of different tests during simulation.

Fig. 4. shows the generation flow of "SpW packets" from the highest level down to the DUV. The "PU traffic reference" provides additional information about the expected traffic behavior generated by the PUs in the real system, e.g.:

- Distribution of different packet lengths
- Distribution of logical addresses
- Content of payload
- Transfer speed

However, this information is provided in plain text by people that are responsible for the PUs and without the requirement of having knowledge about SystemVerilog or the test environment. To refine the information for the randomization process of the "SpW packet", a constraint file (specific constraints) is defined where the high level constraints are defined in SystemVerilog by the verification engineer. The "specific constraints" influence the randomization every time the Sequencer is requesting "SpW packets" by accessing "specific sequence". "specific sequence" is derived from "base sequence" and will restrict the "base constraints" in a way how it is required for a specific test. If, e.g., "base sequence" creates "SpW packet" with payload lengths between 1 to 1000 bytes, a "specific sequence" could be created with allowed payload lengths between 500 to 600 bytes. Sequences can be seen as containers where packets can be created and randomized depending on defined constraints. If different behavior between tests and simulation runs is required, it is often sufficient to exchange only constraint files instead of changing the code of the test environment.

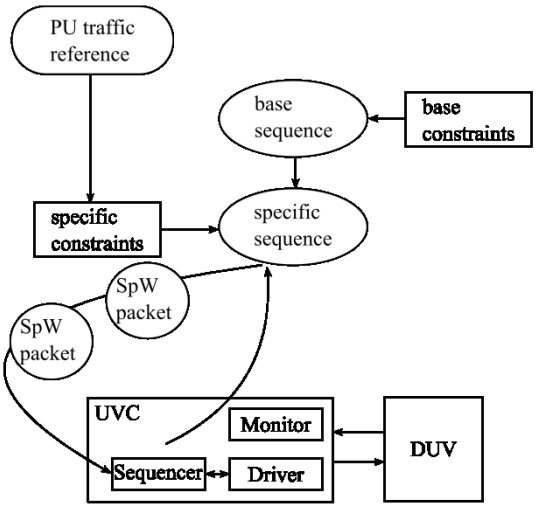

Fig. 4. Stimulus generation flow

Whether a packet needs to be generated depends on the Driver. If the Driver has applied previously requested packet content completely to the DUV, it will request a new one until the maximum amount of executed packets is reached. If the Drivers of all UVCs have finished the packet application, the simulation stops after a predefined drain time. This time is required to let the DUV work until the last packets are processed.

The Monitor passes back the packets to the SCB (Scoreboard). Packets could be sent from drivers to scoreboard directly but this is not recommended because the UVC can also act in a passive way. In that case the driving part of the UVC is deactivated and replaced by, e.g., real HDL designs. But the monitoring of the DUV interface shall still be in place for checks and coverage purposes independent of the Driver activity. This replacement would happen, for instance, if we would decide to use a HDL description of our PUs to create the data traffic for the routers. This kind of replacement will take place every time HDL subsystems are integrated to a bigger system. Once connected they will exchange data over interfaces that were previously connected to UVCs during separate subsystem tests. Unfortunately, this flexibility is associated with additional effort since the monitor must reassemble packets.

The "SpW packet" whose content finally drives the DUV is a class-based object that contains all the packet information like addresses, payload content, end of packet marker etc. but also additional information like delays, packet drop probabilities or system time information, which is important for the performance analysis described in the following chapter.

V. PROPERTIES AND PERFORMANCE TRACKING

If the UVCs are submitting data to the network according to the behavior of the real PUs, it is necessary to check that the network is able to handle the traffic. To identify suboptimal

behavior and optimization possibilities the following properties are tracked during the simulation:

Packet Latency. It provides the transition time of the first byte of each packet from insertions into the network and reception at the destination. Only the first byte is considered to keep the latency determination independent from packet size.

Packet Jitter. The jitter is defined as the difference between the maximum and the minimum transition time of packets. As for the packet latency, only the first byte of a packet is considered for the measurement.

RX Router Buffer Full. If data paths through the network are saturated, receiving (RX) buffers become full. Either this can be caused by a UVC that provides too much data for a specific destination or too many packets are transferred over the same data path.

TX Router Buffer Empty. To optimize the traffic load of the network it is useful to identify buffers and links that are rarely or never used. If no or little traffic is sent over an output port of a router, this can be derived from the status of the TX buffers.

Router Closed Packet. The router has the option to close packets by EEP (Error End Of Packet) in two cases:

1. A connection time between input and output exceeds a predefined threshold.
2. A maximum amount of packet bytes during transfer between input and output was exceeded.

All these attributes are monitored and tracked during the whole simulation by the PTRACK and RTRACK modules shown in Fig. *2*. The PTRACK module is responsible for tracking packet latency and the packet jitter. This module does not need to be connected to signals inside the DUV. Instead, it gets data captured by the Monitor inside the UVC. How the PTRACK tracking is performed is described in the following.

Every time an arbitrary UVC and its related Driver insert a packet in the network, the Monitor reassembles a copy of this packet and sends it to the SCB (Scoreboard). Depending on the packet type, the SCB decides whether it is stored or not. In general, only RMAP (Remote Memory Access Protocol) configuration packets are not stored. In addition to the packet content, two attributes are stored:

- System time related to the moment the first byte was inserted into the network
- Source UVC

The packet plus additional attributes is stored until the packet is received at the destination UVC. Now the related Monitor captures and reassembles the received packet. If the received packet is not a reply related to an RMAP configuration, the Monitor triggers the tracking function inside the SCB to pass the captured packet further to the PTRACK module.

The PTRACK module internally creates a dynamic data structure to store all required time information. This requires extracting the previous stored expected packet. As mentioned before, the expected packet was extended among other values by its network entry time. We know now when the packet entered and left the network. Now all information is available to create or extend the structure with the following attributes:

- Longest transition time
- Shortest transition time
- Average transition time
- Jitter
- Logical address
- Source UVC

All transition times and jitter values are related to a specific data path. For each path, a separate tracking needs to be performed. The example in Fig. 5 shows that UVC 3 can be reached by use of three logical addresses: 56, 67 and 38. With R0 as source, three data paths are possible to reach R3:

1. R0 to R1 to R3
2. R0 to R3
3. R0 to R2 to R3

In this example the PTRACK module for UVC 3 would create three entries, one for each data path. Distinguishable by source UVC number and Logical address.

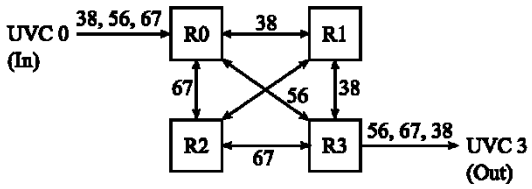

Fig. 5. Logical path example

At the end of the simulation, all tracked entries are printed at the simulator console.

The example applied logical addressing. If path addressing is used, the data path of the packet is not implicitly given by two values (Source UVC and logical address) as it is the case for logical addressing. Path address handling is currently not supported by the performance-tracking framework. This would require the implementation of an algorithm that can reconstruct the path of a packet after reception.

In case of exceptional high packet latencies, congestion or other irregularities, it should be possible to find the cause by observing the router attributes. The RTRACK modules track them. These modules are connected through interfaces to access all router signals no matter where or how deep they are located inside the HDL design. This kind of connection allows to add tracking functionalities if required. One just have to add the signals that should be observed and extend the RTRACK class.

The RTRACK modules start observing with the beginning of a simulation. For the buffer signals RX full and TX empty the following attributes are tracked:

- Trigger amount
- Longest time active
- Shortest time active
- Average time active

Every time a buffer signal is triggered, the related tracking structure updates their values. Note that capturing is performed for every router input and output port separately inside each RTRACK module. The information how often a router closed a connection with an EEP is only tracked by counting the occurrences. Because the amount of available buffers per router is known during the whole simulation, it is not required to extend the tracking structure in a dynamic way, as it is the case for the PTRACK module.

VI. Conclusions

This paper presented an approach to evaluate the network performance of the OBC-NG prototype depending on the network configuration and constraints that model the incoming traffic into the network. We gave an overview of the involved system components beginning with the DUV and finishing with the performance/attribute tracking that is embedded into the test environment. The full access to all signals inside the DUV is a huge advantage compared to the real hardware.

The presented approach can help to evaluate and optimize different network configurations and topologies for a given application.

However, we are aware of the fact that the results strongly depend on the quality of the input traffic model. To determine sufficient traffic models is subject of future research. In addition, the current implementation of the performance tracking does not fulfill the requirements of credible statistics [6]. We do not perform, for instance, confidence determination yet. This requires a much higher effort in terms of memory and arithmetic functions in such a hardware-related environment.

Unfortunately, it is not possible to provide test results in this paper. It is because the DUV network configuration and the traffic model are not finalized so far. Therefore, the application of the test environment to the real system is planned as the next work.

References

[1] D. Lüdtke, K. Westerdorff, K. Stohlmann, A. Börner, O. Maibaum, T. Peng, B. Weps, G. Fey und A. Gerndt, “OBC-NG: Towards a reconfigurable on-board computing architecture for spacecraft,” in Aerospace Conference, 2014 IEEE, 1-8 March 2014.

[2] Accellera Systems Initiative, “Universal Verification Methodology (UVM) 1.2 Class Reference,” June 2014.

[3] ESA Requirements and Standards Division ESTEC, “SpaceWire – links, nodes, routers and networks,” 2008, ECSS-E-ST-50-12C.

[4] ESA-ESTEC Requirements & Standards Division SpaceWire, "SpaceWire - Remote memory access protocol," 2010, ECSS‐E‐ST‐50‐52C.

[5] IEEE Computer Society and the IEEE Standards Association Corporate Advisory Group, “IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language”, 21 February 2013.

[6] K. Pawlikowski, H.-D. Jeong und J.-S. Lee, “On credibility of simulation studies of telecommunication networks,” Communications Magazine, IEEE, pp. vol.40, no.1, pp.132-139, Jan. 2002.

Recording SpaceWire Traffic

SpaceWire Test and Verification, Short Paper

Stephen Mudie, Chris McClements, Alan Spark,
Stuart Mills, Alex Mason
STAR-Dundee Ltd
Dundee, Scotland, UK
stephen.mudie@star-dundee.com

Martin Dunstan, Steve Parkes
School of Computing
University of Dundee
Dundee, Scotland, UK
mdunstan@computing.dundee.ac.uk
sparkes@computing.dundee.ac.uk

To support the validation and debugging of complete SpaceWire systems, STAR-Dundee Ltd have developed a SpaceWire Recorder. Using STAR-Dundee SpaceWire technology and the latest solid state data storage technology, the SpaceWire Recorder is capable of unobtrusively recording traffic on up to four links in both directions at a maximum aggregate data rate of 600Mbit/s. The maximum amount of data that can be recorded is limited only by the size of the solid state disks used. A Traffic Viewer software application provides a simple means of operating the recorder, as well as displaying and managing the large volume of SpaceWire traffic that can be recorded.

***Index Terms*— Relevant indexing terms: SpaceWire, Networking, Spacecraft Electronics, Recorder**

I. Introduction

Viewing SpaceWire traffic on a complete SpaceWire system for validation and debugging purposes can be challenging. One solution may be to use multiple SpaceWire Link Analyser Mk2s, each connected on a different link and each configured to capture data at the same time via external triggers.

A SpaceWire Link Analyser Mk2 will unobtrusively capture very detailed information regarding SpaceWire traffic on a single SpaceWire link. The timing information of every SpaceWire character is captured along with a trace of the data and strobe signals. The amount of data captured however is limited by the Link Analyser memory size, the units are not time synchronized and each Link Analyser will have a separate instance of software running, making it very difficult to interpret the operation of the SpaceWire system.

To resolve this problem STAR-Dundee has developed a SpaceWire Recorder. The SpaceWire Recorder is a standalone unit capable of recording SpaceWire traffic on multiple links unobtrusively to a hard disk. It is supplied with software that controls recording and displays the recorded traffic in a single application, allowing data on all links to be viewed simultaneously. The recording size is limited only by the hard disk size meaning large volumes of SpaceWire traffic can be recorded over long periods of time. Entire recordings can be viewed in software as opposed to only part of the recording.

II. Hardware

The SpaceWire Recorder is a standalone PC. It consists of a CompactPCI rack containing a power supply, one solid state disk (SSD) carrier, a processor board and the STAR-Dundee SpaceWire Recorder cPCI card.

Fig. 1. SpaceWire Recorder

By default the SpaceWire Recorder comes with two 480GB solid state disks. One disk is responsible for storing SpaceWire traffic recordings and the other holds the system files such as the operating system. The recordings disk is held within a SSD carrier providing easy access. The system files disk is attached directly to the processor board.

A powerful processor board accompanies the SpaceWire Recorder with an Intel Core i7 and 8GB RAM. Amongst the I/O there are two DisplayPort ports and a VGA port allowing three monitors to be used, plus gigabit Ethernet making remote connection possible. A rear transition module accompanies the processor board. This allows access to I/O from the back of the system.

III. SpaceWire Recorder cPCI Card

STAR-Dundee have developed a cPCI card capable of many different configurations. The SpaceWire Recorder is one of the first products to make use of this. The SpaceWire Recorder cPCI card has eight SpaceWire interfaces used to

unobtrusively record SpaceWire traffic in both directions on four links. Memory on-board the SpaceWire Recorder cPCI card allows the traffic to be captured and spooled very quickly to disk. Four external triggers allow the recorder to integrate with external equipment. Each external trigger can be configured as either an input or output trigger. These allow the user to control recording in response to an input signal or generate an output signal when an event of interest occurs. A dedicated trigger button allows the user to force a trigger providing further control over recording. The status of the SpaceWire interfaces, external triggers and trigger button are indicated by LEDs.

Fig. 2. SpaceWire Recorder cPCI Card

IV. Software

The SpaceWire Recorder comes with all the necessary software pre-installed. This consists of Windows Embedded Standard 7, the board support packages required by the processor board, STAR-System (including the STAR-System PCI Driver) and the Traffic Viewer GUI software.

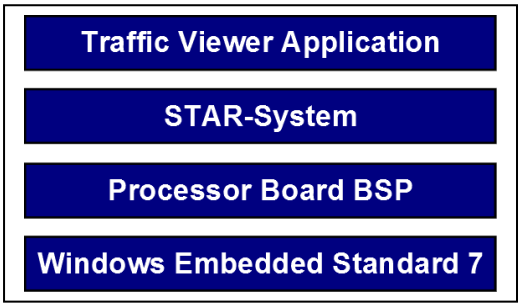

Fig. 1. Software Layers

Windows Embedded Standard 7 delivers the performance, reliability and flexibility of Windows 7 in a form specific to the requirements of the SpaceWire Recorder.

Developed by STAR-Dundee, STAR-System is a high performance suite of software designed to work with all future and a range of current STAR-Dundee devices. STAR-System includes numerous modules used by the SpaceWire Recorder, including the STAR-System PCI Driver. The fast data rates at which the SpaceWire Recorder can record are partly achievable thanks to the performance of the STAR-System PCI Driver.

V. Traffic Viewer

The Traffic Viewer is a GUI application that allows the user to control the SpaceWire Recorder and display and manage recordings.

The user can configure the recording directory, the maximum recording size and the maximum recording time. Start and stop buttons control recording. Once a recording is complete it is displayed. Each column in the view represents the SpaceWire traffic in one direction of a SpaceWire link. The left most column shows the recording time.

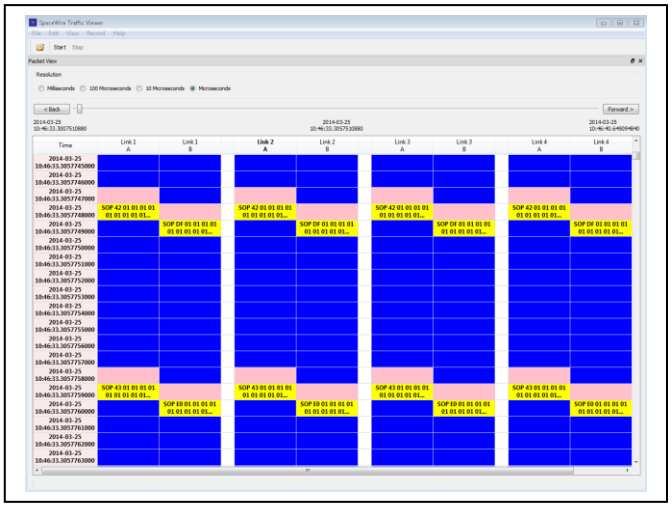

Fig. 2. Traffic Viewer

10ms of recorded SpaceWire traffic is loaded into the display at any one time. The user can specify the timing resolution of the display: 1us, 10us, 100us and 1ms. To seamlessly load another section of the recording, the time slider at the top of the view is used. Left and right of the time slider is the recording start and end time. To quickly navigate the recorded traffic the user can specify a specific time relative to the start of the recording or use the built-in search capabilities. Users can search for a data pattern, a time-code value, a specific error, the start of a packet, an EOP or an EEP.

Double clicking a packet opens a dialog that shows the packet in greater detail. It shows the time at which it was captured, the packet duration and the packet data.

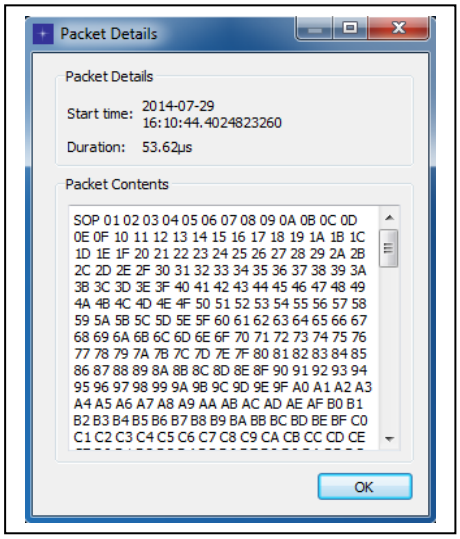

Fig. 3. Packet Dialog

VI. Capabilities

Using the SpaceWire Recorder SpaceWire Traffic can be recorded at high speed on many links over a long period of time. The maximum amount of data that can be recorded is only limited by the size of the solid state disks in use.

The SpaceWire Recorder records data, time-codes and link errors. Using the Traffic Viewer application, recording can start and stop at the click of a button. Alternatively a recording can automatically stop when the recording disk is full, a specified amount of data has been recorded to disk or a pre-defined period of time has elapsed since the recording was started.

Recorded SpaceWire traffic is displayed in the Traffic Viewer. Search capabilities make it easy to navigate large recordings and identify the SpaceWire traffic of most interest. Recordings are automatically saved to be viewed at a later date.

VII. Performance

To measure the recording performance of the SpaceWire Recorder a SpaceWire EGSE was used to generate data in both directions of all four links. The SpaceWire EGSE is a SpaceWire equipment emulator capable of full real-time performance. Once configured using a unique SpaceWire specific scripting language, it operates independent of software, capable of saturating a SpaceWire link with data at a 200Mbit/s link speed, i.e. no Nulls between data characters.

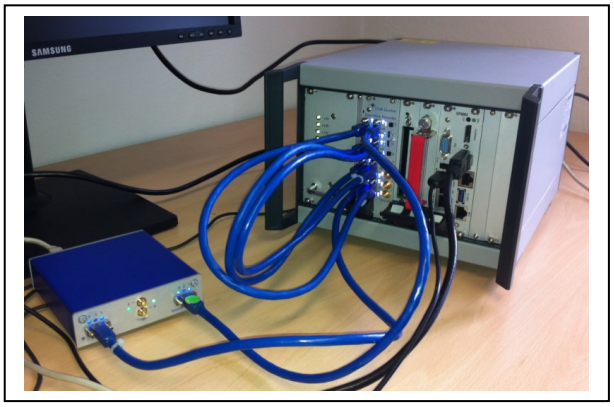

Fig. 4. Performance Test Setup

The SpaceWire EGSE was used to generate packets of a specific size consisting of random data at a fixed link speed over a prolonged period whilst recording was enabled. If no hardware buffer overflow was detected, the test was started again with an increased link speed. This incremental process was performed until a hardware buffer overflow was detected signifying the maximum recording speed was exceeded. Internal statistics monitoring within the SpaceWire Recorder software provided detailed information regarding the recording data rates achieved and the usage of the SpaceWire Recorder spooling buffers.

The SpaceWire Recorder is capable of recording to disk at an aggregate data rate of 600Mbit/s. However the speed at which it can record to disk differs depending on the size of the recorded SpaceWire packets. The table below plots the aggregate recording data rate achieved for different packet sizes.

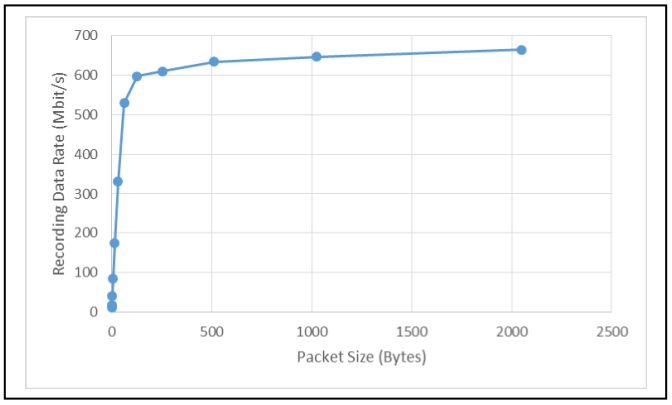

Fig. 5. Packet Size vs Aggregate Recording Data Rate

The best performance achieved was an aggregate recording data rate of 664.78Mbit/s whilst recording continuous 2048 byte packets. The worst performance achieved was a data rate of 11.24Mbit/s whilst recording continuous 1 byte packets which corresponds to a packet rate of 1.4M packets/s. The aggregate recording data rate drops considerably when recording continuous packets of size less than 64 bytes. The

main reason for this is the recording time-stamp overhead associated with each packet.

If the rate at which data is transmitted from the SpaceWire devices connected to the SpaceWire Recorder exceeds the rate at which it can be recorded then a capture overflow will occur. If this happens, the Traffic Viewer application will stop recording automatically and alert the user.

VIII. Future Work

The SpaceWire Recorder hardware has capabilities currently not fully implemented in software. The Traffic Viewer application currently does not support:

- Triggering: start recording when an event of interest occurs e.g. link error
- Filtering: disable or enable recording of time-codes and specific errors (currently enabled by default)
- Link statistics: view the average bit rate of each bi-directional link

New views of recorded SpaceWire traffic will also be added to the Traffic Viewer. The SpaceWire Recorder is currently being used to help validate the SpaceWire Plug and Play (PnP) protocol. Feedback from this and other users will be used to improve existing features and guide the development of new features.

IX. Conclusion

The SpaceWire Recorder is an essential tool for the validation and debugging of an entire SpaceWire network. It serves a different purpose from a SpaceWire Link Analyser Mk2, which is designed to capture a much smaller, yet more detailed, amount of SpaceWire traffic on a single SpaceWire link.

The SpaceWire Recorder unit is built around a high performance SpaceWire Recorder cPCI card complimented by solid state disks and a powerful processor board. Combined with the STAR-System PCI Driver and the Traffic Viewer software application, the SpaceWire Recorder has impressive capabilities and delivers exceptional recording performance. Large quantities of SpaceWire traffic over multiple links can be recorded for long periods of time. The maximum aggregate recording data rate achieved whilst testing performance was 664.78Mbit/s with 2048 byte packets. Recorded SpaceWire traffic can be viewed and managed using the Traffic Viewer application.

References

[1] STAR-Dundee, SpaceWire Recorder User Manual, v1.01.

[2] STAR-Dundee, http://star-dundee.com/products/spacewire-recorder, SpaceWire Recorder, STAR-Dundee Website.

[3] STAR-Dundee, http://star-dundee.com/products/spacewire-link-analyser-mk2, SpaceWire Link Analyser Mk2, STAR-Dundee Website.

[4] STAR-Dundee, http://star-dundee.com/products/spacewire-egse, SpaceWire EGSE, STAR-Dundee Website.

High Speed Test and Development with the SpaceWire Brick Mk3

Test & Verification 1, Short Paper

Stuart Mills, Pete Scott, Steve Parkes
STAR-Dundee Ltd.
Dundee, Scotland, UK
stuart.mills@star-dundee.com, pete.scott@star-dundee.com, steve.parkes@star-dundee.com

***Abstract*—The original STAR-Dundee SpaceWire-USB Brick has provided a simple yet powerful interface to SpaceWire networks for a number of years. STAR-Dundee's SpaceWire Brick Mk3 provides all the features of the original Brick, but with better performance, better software, better documentation and the same high quality support. It will replace the Brick with a product which can be used to very easily perform numerous SpaceWire test and development activities, and at very high speeds.**

***Index Terms*—SpaceWire, USB, Brick, Interface, Router, STAR-Dundee, Spacecraft Test and Development Equipment, STAR-System.**

I. Introduction

The original SpaceWire-USB Brick [1] has been serving the SpaceWire community for over ten years. It is an excellent learning tool for those new to SpaceWire, but it is also used by more experienced engineers to develop and test new SpaceWire devices and networks.

The software provided with the Brick was developed to provide the highest possible throughput, and is capable of transmitting and receiving concurrently from/to a PC over a USB 2.0 cable at the full 160 Mbits/s data rate achievable on a 200 Mbits/s SpaceWire link.

The Brick and its successor the Brick Mk2 [2] do have their limitations, however. Both devices are restricted by the throughput constraints of USB 2.0, which means that a maximum combined throughput of around 360 Mbits/s is achievable.

This paper introduces the replacement for these devices – the Brick Mk3. This device will be released later this year (2014) and includes all the capabilities of the Brick Mk2, plus a number of improvements. It is connected to the PC using USB 3.0, which offers greatly improved performance when compared to USB 2.0. The paper describes the advantages of using USB 3.0 for SpaceWire test and development equipment, introduces the new features in the Brick Mk3 hardware and software, and shows some typical scenarios in which the Brick Mk3 can be used. It concludes with a summary of the benefits of using the Brick Mk3 for SpaceWire test and development.

II. Bus Comparison

The buses most commonly used to connect additional devices to a PC or rack are PCI and related technologies, and USB. Devices can also be connected over a TCP/IP network, e.g. using Ethernet or wireless. Each bus offers different capabilities, with advantages and disadvantages of each. For this reason STAR-Dundee offers PCI [3], PCI Express (PCIe) [4], CompactPCI (cPCI) [5] and USB [2] [6] SpaceWire interface and router devices.

Previous STAR-Dundee USB devices have included a USB 2.0 connection [7]. A new version of USB, USB 3.0 [8], was released in 2008, offering higher data rates than the previous version. As this version of USB has gained market share and is now provided in most new PCs, STAR-Dundee has released the new Brick Mk3 with support for USB 3.0.

To highlight the benefits of using USB 3.0 in the Brick Mk3, the remainder of this section compares each of the buses mentioned above, concentrating on the advantages of USB 3.0 for SpaceWire test and development.

A. Throughput

Both PCI and cPCI offer full-duplex data signalling rates of approximately 1 Gbits/s [9], while PCIe provides close to 2 Gbits/s per lane [10]. USB 2.0 is slower in comparison, providing 480 Mbits/s, half-duplex [7]. One of the advantages of USB 3.0 is that it provides full-duplex data signalling at rates of up to 5 Gbits/s [8].

Although it is not possible to achieve user data rates at the full signalling rates of each of these buses due to protocol overheads, STAR-Dundee software and hardware is designed to obtain rates as close as possible to the maximum achievable. The overheads of the USB protocol are slightly higher than each of the PCI protocols, which have very small overheads. However, the high data signalling rates of USB 3.0 means that this is unlikely to have an effect on the Brick Mk3's performance. Initial investigations with the Brick Mk3 suggest that the device will be more than capable of ensuring both SpaceWire links on the device can concurrently transmit and receive packets at the maximum rate possible.

In comparison, Gigabit Ethernet devices offer a data signalling rate of 1 Gbits/s [11]. However, TCP/IP devices have greater overheads than the other buses, due to the use of the TCP/IP protocol suite in addition to the bus's own protocol overheads, e.g. that of Ethernet in the case of Gigabit Ethernet.

B. Latency

Latency values of each of the buses are difficult to compare, due to the different natures of each bus, but both PCI and PCI Express provide the best latency of the buses being discussed. USB 2.0 latency is not as good as that of the PCI buses. However, one of the improvements to USB 3.0 was to the latency that could be achieved, particularly when large amounts of data are transferred. Initial investigations with the Brick Mk3 suggest latency is slightly better when transmitting and receiving SpaceWire packets over USB 3.0 in comparison to USB 2.0.

TCP/IP devices offer much poorer latency than the PCI and USB buses. Latency of TCP/IP devices will also degrade with each additional hop across the network that is required to reach the device.

C. Characteristics

Each of the buses considered provide advantages in different circumstances. For example, cPCI devices can be used in a rack system, while TCP/IP devices can be accessed from another location on the network.

One advantage of USB is that it very easy to connect and disconnect devices to/from a PC. Unlike the PCI buses, USB devices can be connected to laptops, in addition to desktop and rack PCs, and can be added or removed while the operating system is running. Although not all PCs support USB 3.0 as yet, the Brick Mk3 can also be used in older USB 2.0 ports.

III. Hardware Features

The Brick Mk3 hardware is an evolution of previous STAR-Dundee USB devices. It includes all the new features added to the Brick Mk2 when it replaced the original SpaceWire USB Brick. These include link speed and state change event signalling, the ability to inject errors on the link and support for the STAR-System software suite (see section IV).

Fig. 1. SpaceWire Brick Mk3

The Brick Mk2 includes an improved interface mode when compared to the original Brick, with independent channels for data and configuration. The Brick Mk3 improves upon this with the ability to operate as a true interface, with independent channels for each link and a further channel for device configuration. This allows the device to be configured while simultaneously transmitting and receiving on both links. The Brick Mk3 can also be used in router mode, as with other STAR-Dundee interface devices. In this mode it offers three external ports which are transported over the USB port in parallel.

Another improvement in the Brick Mk3 is in the options available for setting the link speed. Both the Brick Mk2 and the Brick Mk3 allow the link speed to be set by specifying multipliers and divisors, with the divisor being any value in a large range. The Brick Mk2 limited the multiplier to be from a small list of values, but the Brick Mk3 uses the same method provided by the SpaceWire PCIe of allowing the multiplier to be any value in a large range.

Work has also been performed to improve the physical characteristics of the Brick Mk3. The box which houses the Brick Mk3 is very different from previous iterations of the Brick, in a blue metal case with the device type clearly visible on the top (see Fig. 1). The SpaceWire connectors are mounted side by side, rather than on top of one another. This makes it much easier to insert and remove SpaceWire cables, and to view the LEDs above each port.

The Brick Mk3 features hardware designed to prevent any single point of failure causing damage to equipment interfaced to the SpaceWire or Trigger ports. A FMECA report is available on request which provides further details on this protection.

As with previous iterations of the Brick, the Brick Mk3 is USB powered, with only a single USB cable required to connect the device to a PC to provide power and a data connection. Although the Brick Mk3 takes advantage of the benefits of USB 3.0, it can also be used in older USB 2.0 ports, with only the throughput and latency that can be achieved affected. When used in a USB 2.0 port, performance is similar to that of the Brick Mk2.

IV. Software Support

Software support for the Brick Mk3 hardware is provided by STAR-Dundee's software suite, STAR-System. This suite can be used with all of STAR-Dundee's recent and planned future interface and router devices, including the SpaceWire Brick Mk2 [2], Router Mk2S [6], PCIe [4], PCI Mk2 [3] and cPCI Mk2 [5].

STAR-System consists of a number of layers. At the bottom are the device drivers for communicating with the hardware. STAR-System includes Windows and Linux drivers for communicating with STAR-Dundee USB devices, and these were updated to add support for the Brick Mk3.

Above the drivers is the STAR-System core and the APIs for interacting with the devices. These are designed to be generic, and not specific to any device, so only some very

minor changes to the core's internals were required to support the Brick Mk3.

At the top of the stack are the user applications. STAR-System includes both command-line test applications and Graphical User Interface (GUI) applications, covering many typical ways in which a SpaceWire interface or router device is used. GUI applications are provided to:

- Type in the bytes of packets and have these transmitted
- Receive packets and display their bytes
- Specify complex packet formats and have these transmitted at high rates
- Receive packets at high rates and compare their format to specified complex packet formats
- Configure the properties of devices, including their routing tables
- Inject errors on a link

Again, these applications are all designed to be generic and to work with all device types. Only some minor changes were required to the Device Configuration application to support the features specific to the Brick Mk3. The other applications needed no changes to work with the Brick Mk3.

The STAR-System drivers, APIs and applications are all designed to provide very high data rates and low latency. When tested with the Brick Mk3 in internal loopback mode (i.e. not accessing SpaceWire), STAR-System applications were capable of transmitting and receiving at rates of approximately 1 Gbit/s, i.e. 2 Gbits of data crossed the USB link every second.

The release of the Brick Mk3 will coincide with the release of version 3.0 of STAR-System. This will include a number of improvements from the last release, including:

- A new Time-code GUI application for transmitting and receiving time-codes and configuring device settings related to time-codes
- Numerous improvements to the existing GUI applications
- The SpaceWire CUBA Software, a command-line application previously provided with the original Brick for transmitting and receiving RMAP commands and SpaceWire packets
- Context sensitive help in all GUI applications
- More detailed documentation

The main change to STAR-System version 3.0, however, is in the internal core of the software. A great deal of work has been done on improving the performance of the software stack, reducing CPU usage and latency, and increasing throughput for all supported STAR-System devices. These were areas in which STAR-System already excelled, but improvements were identified which would be beneficial on real-time operating systems and in low resource environments. A pleasant side-effect of making these changes is that they are also beneficial when using STAR-System on standard PCs running Windows or Linux.

V. Using The Brick Mk3

The sections above have described the individual improvements to the Brick Mk3 and some of the features provided. This section describes how these features can be used in typical SpaceWire test and development activities.

A. Checking Data Received From an Instrument

The Brick Mk3 is capable of receiving packets at very high rates. When testing a SpaceWire instrument, the Brick Mk3 can be combined with the STAR-System Sink application to not only receive data from the instrument at high rates, but also check that the packets are in the correct format, and record the instrument data to file.

A SpaceWire camera is likely to transmit packets which contain more fields than just the image data. There is likely to be address information at the start of the packet and there may be a checksum or CRC at the end. The STAR-System Sink application allows you to specify the format of the packets that are expected to be received. It can then check each field in the received packets is in the correct format, and write individual fields, or the full packet, to file.

Fig. 2 shows an example packet format for a camera configured using the Sink's Packet Format dialog. The Sink expects to receive a single address byte of 0xfe, followed by a 16-bit sequence number. The Sink will check that the address byte is correct and the sequence number increments in each packet. After the sequence number is the image data which is expected to be 1 MByte. As there's no way to know what image the camera will be sending, the content of this field is not checked.

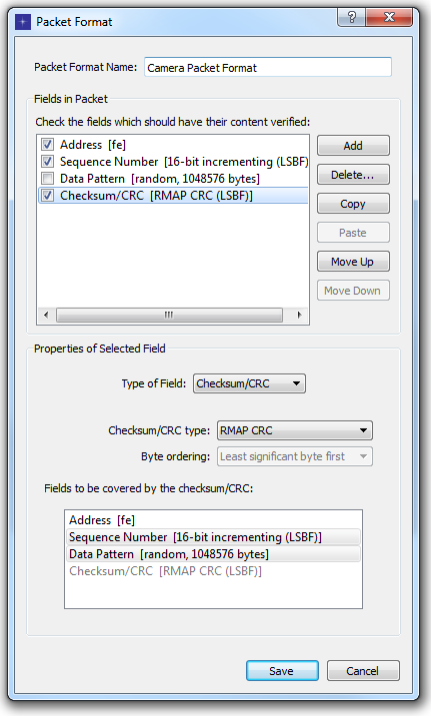

Fig. 2. Sink Packet Format

Finally the packet ends with a CRC. The properties of the CRC are shown in the Packet Format dialog screenshot. The CRC being used is the RMAP CRC, although a number of different CRCs and checksums are supported. The CRC in this example covers the sequence number and image, although it could be set to cover any of the fields in the packet. The Sink will check the CRC is correct in each received packet.

A separate dialog in the Sink application allows packets, or individual fields in packets, to be recorded to file. The format in which each packet or field is written to file is then specified in the dialog shown in Fig. 3. Each of the bytes in the field can be written to the file numerically as text, with spaces or another separator between each value. The field in each packet can also be written to one large file. For the camera's images in the screenshot, we have chosen to write the images to file as binary data, with a new file used for each image. Assuming these files are in an appropriate format, it should then be possible to open the files received from the camera and view them in a photo viewer.

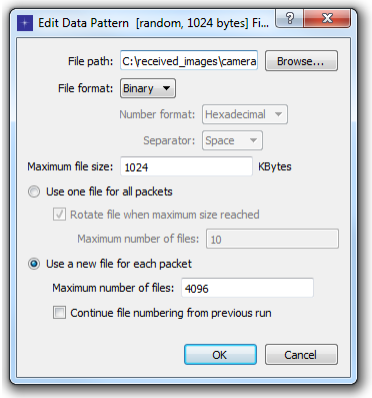

Fig. 3. Sink Recording to File

The Sink application provides many other features, and has a partner application, the Source, which can be used to transmit packets. It uses the same packet formats as the Sink, so the camera packet format specified here can also be used in the Source to simulate the camera.

B. Configuring a SpW-10X (AT7910E)

The Device Configuration application can be used to configure STAR-System devices such as the Brick Mk3, providing an interface for setting link speeds, routing tables and viewing error status information. The application can also be used to configure routing devices over a SpaceWire network, using a device such as the Brick Mk3 to communicate with the devices on the network. Supported routing devices include the AT7910E, the ESA SpaceWire Router [12].

When working with a spacecraft network containing AT7910E devices, the Brick Mk3 can be connected to the network and used to check the status of these devices, as shown in Fig. 4. In this screenshot, the Device Properties of an AT7910E are on display, showing the general properties of the device, and providing the option to configure settings which affect the entire device. Tabs are provided for each port, including the configuration, SpaceWire and external ports, showing the current error status, and allowing the links to be started, stopped, etc. The final tab provides the ability to configure each routing table entry of the device.

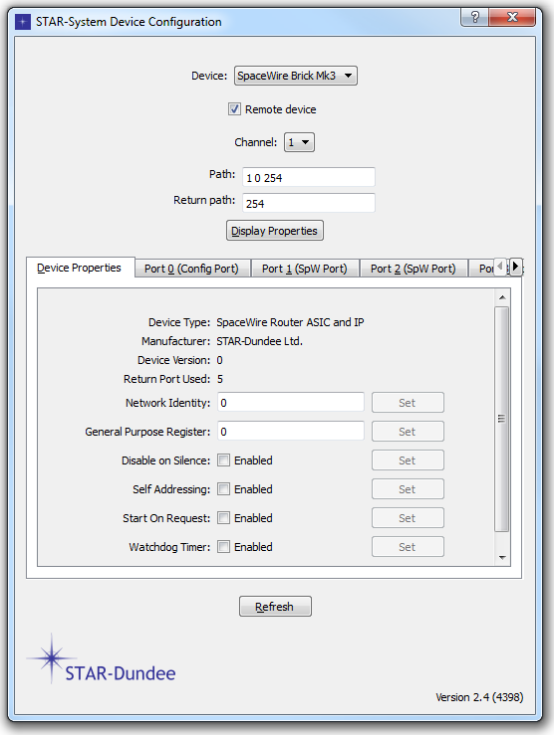

Fig. 4. Device Configuration of an AT7910E via a Brick Mk3

C. Acting as a Time-code Master

When experimenting with time-codes, e.g. for SpaceWire-D development, the Brick Mk3 can very easily be enabled as a time-code master. The STAR-System Time-code application includes a tab for enabling the device as a time-code master, see Fig. 5. The frequency at which time-codes are to be generated can be entered in hertz, and the Brick Mk3 will be enabled as a time-code master once the Enable button is clicked.

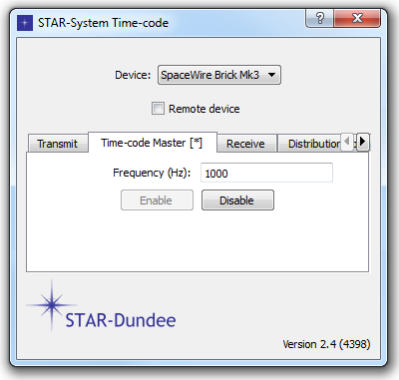

Fig. 5. Enabling a Brick Mk3 as a Time-code Master

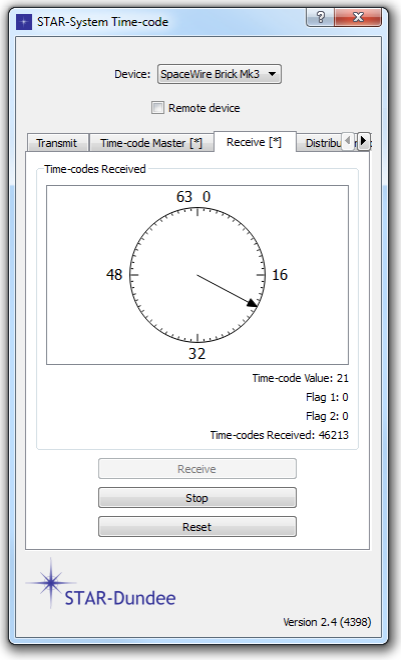

Fig. 6. Receiving Time-codes With a Brick Mk3

To see the time-codes which are being generated, or which are being received from another time-code master on the network, the Time-code application includes a tab for receiving time-codes, shown in Fig. 6. This shows each time-code's value as it received on a clock, as well as displaying the value numerically along with the values of the time-code flags.

The Time-code application also includes tabs for transmitting individual time-codes and for specifying which ports time-codes should be routed out of.

VI. Summary

The SpaceWire Brick Mk3 is a powerful interface and router device, which offers the capability to transmit and receive at the maximum speed that can be achieved on a 200 Mbits/s link, on two links concurrently while also configuring the device. In other words it is capable of transmitting and receiving at 160 Mbits/s on both SpaceWire links, while also reading and writing registers on the device, giving a total combined data rate of greater than 640 Mbits/s. It can also transmit and receive packets with latencies which are better than can be achieved with the SpaceWire Brick Mk2. This is possible because of the improved throughput and latency provided by USB 3.0, and because of the inclusion of a true interface mode in the device.

Combined with the comprehensive STAR-System software suite, the Brick Mk3 product can be used to perform many of the tasks required during SpaceWire test and development.

References

[1] STAR-Dundee, "SpaceWire-USB Brick", http://www.star-dundee.com/products/spacewire-usb-brick, 2014.

[2] STAR-Dundee, "SpaceWire-USB Brick Mk2", http://www.star-dundee.com/products/spacewire-usb-brick-mk2, 2014.

[3] STAR-Dundee, "SpaceWire PCI Mk2", http://www.star-dundee.com/products/spacewire-pci-mk2, 2014.

[4] STAR-Dundee, "SpaceWire PCIe", http://www.star-dundee.com/products/spacewire-pcie, 2014.

[5] STAR-Dundee, "SpaceWire cPCI Mk2", http://www.star-dundee.com/products/spacewire-cpci-mk2, 2014.

[6] STAR-Dundee, "SpaceWire Router Mk2S", http://www.star-dundee.com/products/spacewire-router-mk2s, 2014.

[7] Compaq et al, "Universal Serial Bus Specification," Revision 2.0, April 27, 2000.

[8] Hewlett-Packard Company et al, "Universal Serial Bus 3.0 Specification (including errata and ECNs through May 1, 2011)," Revision 1.0, June 6, 2011.

[9] PCI-SIG, "PCI Local Bus Specification", Revision 3.0, February 3, 2004.

[10] PCI-SIG, "PCI Express Base Specification", Revision 3.0, November 10, 2010.

[11] IEEE Computer Society, "IEEE Standard for Ethernet", IEEE Std 802.3-2012, December 28, 2012.

[12] Atmel, "AT7910E, SpW-10X SpaceWire Router, Datasheet", http://www.atmel.com/Images/doc7796.pdf.

SpaceWire-to-GigabitEther and SpaceWire backplane

SpaceWire test and verification, Short Paper

Shigeyuki Arase, Iwao Fujishiro
Shimafuji Electric
8-1-15 Nishikamata,Ota-ku,Tokyo144-0051,Japan
arase@shimafuji.co.jp

Masaharu Nomachi
Osaka University
1-1 Machikaneyama,Toyonaka,Osaka560-0043,Japan

Abstract— **In 2006, we developed SpaceWire platform named SpaceCube cooperation with JAXA and NEC. After the success of SpaceCube project, we developed number of SpaceWire products. Some examples of this innovation include several kind of the SpaceWire interface boards, SpaceWire router and SpaceCubeMK2. These developments included the support and cooperation of JAXA, OSAKA University, Japan Space Systems and NEC. In this paper we describe architecture, functions and usage about our new products which are the SpaceWire-to-GigabitEther and the SpaceWire backplane. The equipment used as verification for ASTRO-H of JAXA**.

***Index Terms*—Backplane**

I. Introduction

The SpaceWire-to-GigabitEther is the bridge unit to convert between TCP/IP protocol and SpaceWire. This unit's notable feature is high-speed, therefore this unit does not install software such as OS. Rather, it is designed only as hardware by FPGA. The second feature of this unit is a lightweight and small size, so it is used a satellite component test .

On the other hand, SpaceWire Backplane with the flexibility and scalability features is developed by Osaka University and JAXA. We developed the SpaceWire Packet Recorder which adopted this backplane. This SpaceWire Packet Recorder is capable of testing SpaceWire network component, and recording large scale SpaceWire network system.

This paper describe architecture and feature of the SpaceWire-to-GigabitEther and the Packet Recorder.

II. SpaceWire-to-GigabitEther

We developed the SpaceWire-to-GigabitEther unit which is the unit to convert SpaceWire and TCI/IP protocol. This unit has 4 SpaceWire ports (all port Max 200M bps) and the 4 ports total link rate is achieved at a theoretical maximum speed 800Mbps.

The IP (hardware logic) which include MAC, TCP stack on FPGA was also developed originally, so one of the future is flexibility to modify and version up this unit.

Fig.1. SpaceWire-to-GigabitEther

TABLE1 Specification

Ether	10/100/1000BASE x 1Port
SpaceWire	Number of Port:4Ports Link Speed:200MBps (Max) Link Status LED:4 led's
FPGA	Spartan6
memory	128MB (DDR2 SDRAM)
size	136mm x 75mm x 25,2mm
power	5V/1.5A (Typ.)

We experienced the SpaceWire-to-GigabitEther unit accentual transfer speeds using a port to 4 ports of Space Wire. We got almost logically full speed from this result.

Use a port Use 2 ports Use 3 ports Use 4 ports

Fig.2. SpaceWire transfer speed

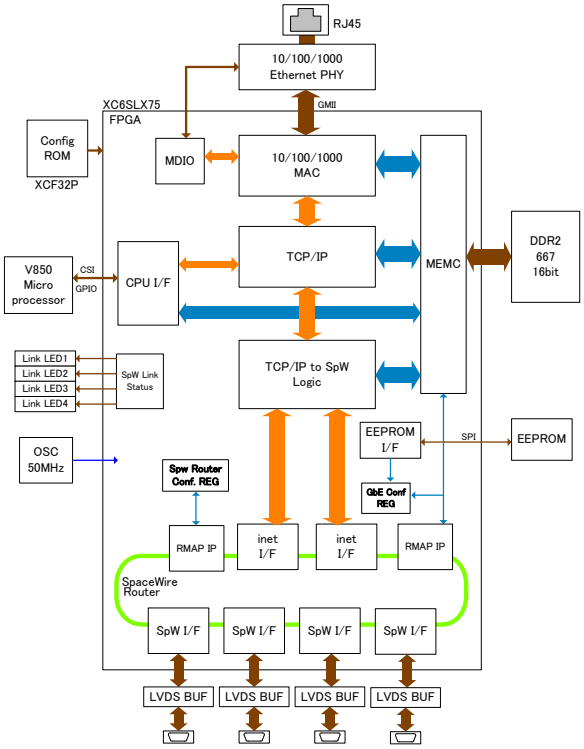

Fig.3.Block diagram

This is the example to connect the SpaceWire-to-GigabitEther unit to target board (The SpaceWire DIO2). It can send RMAP command using sample software on PC. It also sends RMAP command after generate RMAP header information and data to control target board via SpaceWire.

The user can download sample software and develop its own application using the sample software source code from Shimafuji Electric Inc. web page.

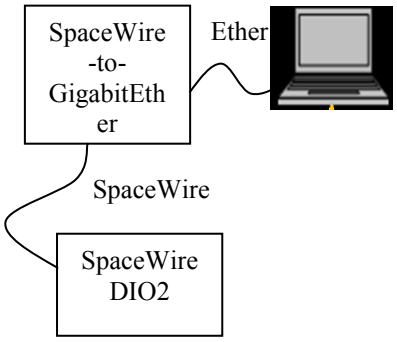

Fig.4. Connection example

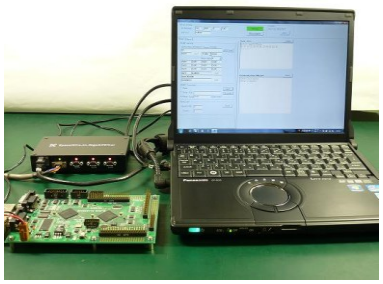

Fig.5. Connections

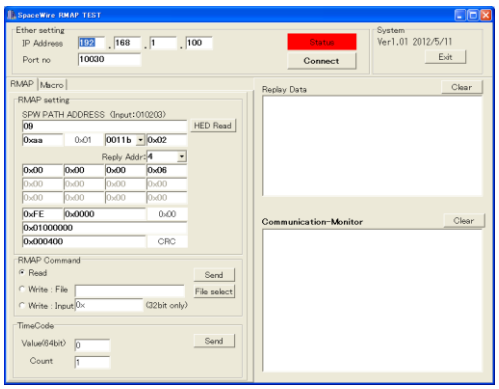

Fig.6Sample software screen

III. SpaceWire Backplane

The SpaceWire Traffic Generator / the SpaceWire Packet Recorder is consist of MCH/SpaceWire Router board, SpW Traffic Generator board / SpW Packet Recorder board and backplane. There are 2 type of backplane, 6slot type which can hold 6 SpW Packet Recorder boards and 12 slots type.

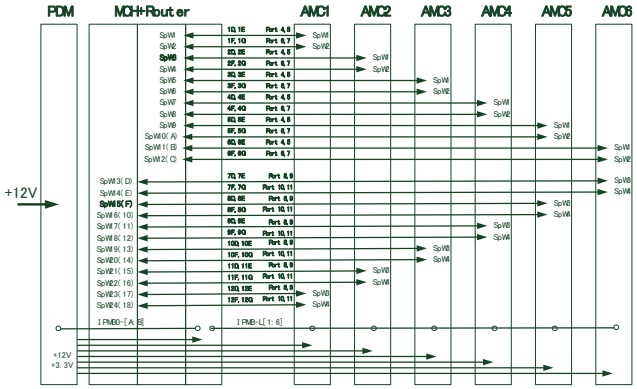

Fig.7. 6 slot backplane topology

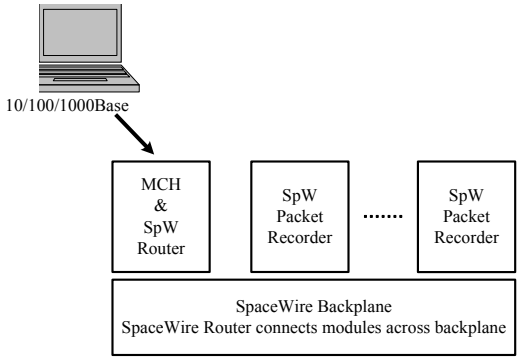

Fig.8. board configuration

TABLE2 back plane specification

chassis	micro-TCA
PC interface	Ether (10/100/1000BASE) x 1Port
Number of SpaceWire monitor channel	The SpaceWire Traffic Generator 6 slot : 1-24Ports 12 slot : 1-48Port The SpaceWire Packet Recorder 6 slot : 1-12 Ports 12 slot : 1-24Port (Link Speed: 200MBps (MAX))
Power	AC100/200V

Fig.8. The SpaceWire backplane6 slots shell Front View

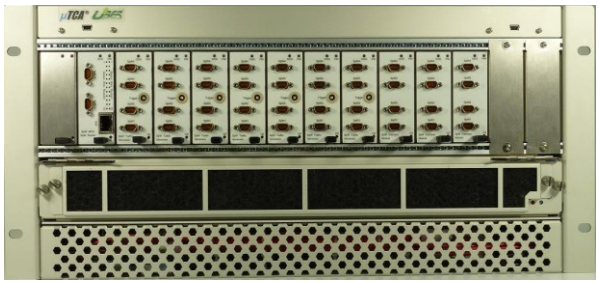

Fig.9. The SpaceWire backplane 12 slots shell Front View

MCH/ SpaceWire Router

All SpaceWire links are connected to MCH which installed SpaceWire router. We have developed original MCH with 28ports SpaceWire router.

Fig.10.Mch/SpaceWire Router

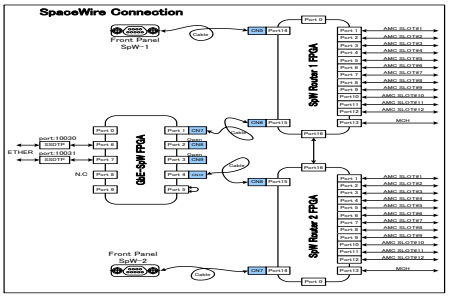

Fig.11.Mch/SpaceWire Router configuration

SpW PTraffic Generator

The SpaceWire Traffic Generator transfer generated SpaceWire packets simultaneously, sequentially or continually. The generated SpW packet(s) will stored in RAM from backplane SpW ports, and send it out when received transmit signal. This SpaceWire Traffic Generator is using same hardware as The SpaceWire Packet Recorder but FPGA IP.

Fig.12.SpW Traffic Generator front view

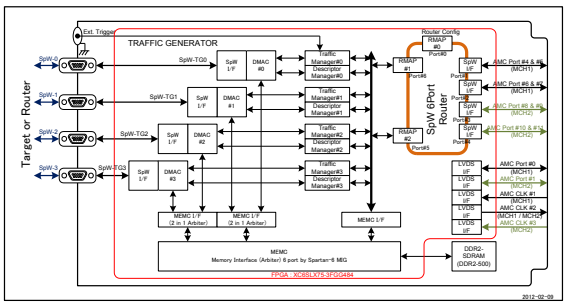

Fig.13.SpW Traffic Generator block diagram

SpW Packet Recorder

The SpaceWire Packet Recorder monitor the SpaceWire link interface. It store Spacewire packet in RAM via MDM connector at the front panel on board with timestamp and attribute information in accordance with the conditions which set by PC. A SpaceWire Packet Recorder board could record 2 channel (4 ports) of SpaceWire links, and the buffer size is assigned 2 M bytes per port.

When the monitoring SpaceWire packet meet the trigger condition, it send notice to host. The method of monitoring, configure the trigger condition, monitor data on memory and time stamp can read or write to/from PC via MCH/SpaceWire Router.

Fig.14.DATA format is describe data format to be stored in the memory and Fig.15.SpW Packet Recorder is front View of the SpaceWire Packet Recorder.

- Format 1

32bit data

Time stamp	Attribute	data
16bit	8bit	8bit

- Format 2 (once / 64K bit)

64bit data

Time stamp	Attribute	data
16bit	8bit	8bit
Time stamp		
32bit		

Fig.14.DATA format

Fig.15.SpW Packet Recorder front view

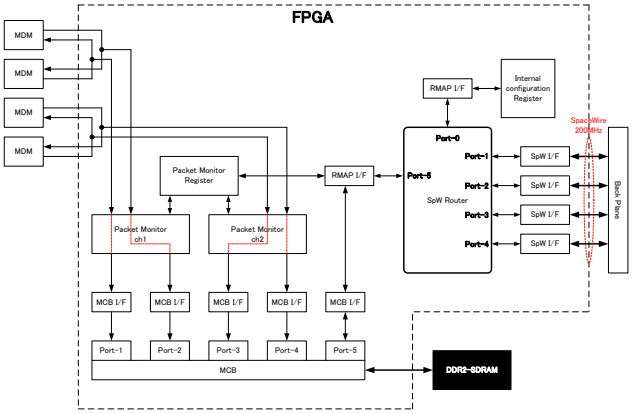

Fig.15. SpW Packet Recorder Block Diagram

Packet Recorder function

1) Trigger much stop mode

It can set various trigger on SpaceWire level and/or RMAP level. When much the data and trigger, stop monitor by configure setting on PC (start trigger, center trigger and end trigger). Fig.11. Example of sampling screen. Below diagram shows the screen with the data and trigger.

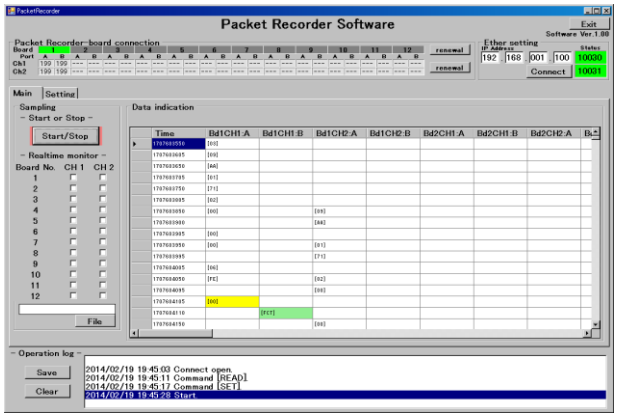

Fig.11. Example of sampling screen

2) Long term continues recording

This mode can record SpaceWire packets on PC HDD until receiving stops command from PC. It could set to record or not the NULL, FCT, EOP/EEP, and data.

Table 3 describes the relation of transfer rate and number of ports to record packets without losing the packets.

Table 3 Number of record port and maximum transfer rate

Number of Port	Rate [bit/sec]	Rate [Byte/sec]
2	66 Mbps	16.5 MB/s
4	33 Mbps	8.3 MB/s
6	22 Mbps	5.5 MB/s
8	16 Mbps	4.0 MB/s
16	8 Mbps	2.0 MB/s
24	5.5 Mbps	1.4 MB/s
32	4 Mbps	1.0 MB/s
40	3.3 Mbps	0.8 MB/s
48	2.7 Mbps	0.4 MB/s

Note

1 : The maximum transfer rate might be different by PC specification

2 : The data size of SpaceWire packet is quarter on this table, the size on the memory includes some information such as time stamp etc.

REFERENCES

[1] Masaharu Nomachi, Shuhei Ajimura, “SpaceWire backplane for ground equipment”, Osaka University, International SpaceWire Conference, Gothenburg, June 2013.

[2] Tadayuki Yuasa, Tadayuki Takahashi, “SpaceWire Traffic Generator: a highly-scalable packet generation device”, JAXA, International SpaceWire Conference, Gothenburg, June 2013.

Using SpaceWire with LabVIEW

SpaceWire Test and Verification, Short Paper

Alex Mason, Steve Parkes
STAR-Dundee Ltd.
Dundee, Scotland, UK
alex.mason@star-dundee.com, steve.parkes@star-dundee.com

***Abstract*— To support customers using the National Instruments LabVIEW software development environment, STAR-Dundee Ltd. have developed LabVIEW libraries and drivers allowing for the rapid integration of STAR-Dundee SpaceWire interface devices into EGSE or test and verification applications. Customers familiar with STAR-Dundee's STAR-System API suite can use a wrapper library to control and configure any supported SpaceWire interface device under the Windows operating system. Using a native LabVIEW NI-VISA driver, users can interface to STAR-Dundee SpaceWire PCI and cPCI, boards on any platform supported by LabVIEW, including National Instruments real-time targets.**

In this paper, the LabVIEW solutions provided by STAR-Dundee are described, including an overview of the APIs, and example usage demonstrating solutions to common tasks.

***Index Terms*— SpaceWire, LabVIEW, NI-VISA, VISA**

I. Introduction

The design of SpaceWire electronic check-out and ground support equipment can be both costly and time consuming. To help alleviate this problem, STAR-Dundee supplies a number of test and development devices that can be used to transmit and receive SpaceWire traffic and configure and monitor devices on a network. Users can write their own custom applications using a provided powerful API.

National Instruments LabVIEW can be used to rapidly develop test and measurement systems with custom graphical user interfaces.

Combining STAR-Dundee equipment with LabVIEW provides a means of rapidly developing SpaceWire test applications.

II. LabVIEW

LabVIEW is a software development environment provided by National Instruments Corporation [1]. The environment provides a visual dataflow programming language in which functions are laid out in a flow chart style, with 'wires' connecting the output of one node to the input of another. Data is operated on at each node immediately as it becomes available, and the compiler identifies segments of code that can run in parallel and automatically splits the application into multiple threads.

LabVIEW offers the ability to work at a higher layer of abstraction than typical text based programming languages like C. For example, no manual memory allocation is required by the user, there are many included libraries hiding the implementation of File I/O and network connectivity, and graphical user interfaces are created in a drag and drop manner.

As an example of the way LabVIEW allows rapid development, compare the volume of C code required to select a desired SpaceWire device and configure its link speed to the code required to perform the same operation with the STAR-Dundee VISA Driver: (Fig. 1)

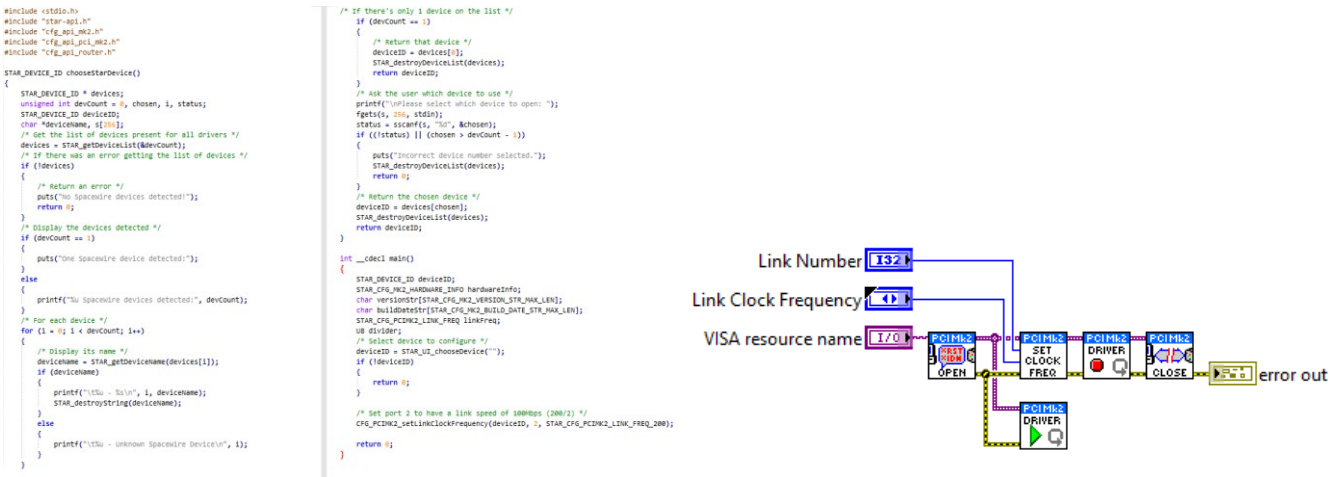

Fig. 1. LabVIEW source compared to text based code.

III. STAR-Dundee LabVIEW Solutions

STAR-Dundee provides two separate LabVIEW solutions: a LabVIEW wrapper around the existing STAR-System libraries (currently provided only for Windows based hosts), and a native LabVIEW NI-VISA driver that can be used on all targets supported by LabVIEW.

A. STAR-System Wrapper

STAR-System is the driver and API system provided with all new and future STAR-Dundee interface and router devices [2]. STAR-System provides high bandwidth and low latency packet transmission and reception, and a consistent API interface to numerous device types. Supported devices include the SpaceWire USB Brick Mk2 and Router Mk2s, and the PCI Mk2 and PCIe boards.

The STAR-System LabVIEW wrapper library [3] provides access to every function exported by the STAR-System C API, and includes a number of example VIs (Virtual Instruments) that provide implementations of commonly performed SpaceWire tasks, such as setting up routing tables, sending and receiving time-codes and configuring link speed and status. Also provided are the RMAP packet library and example implementations of an RMAP Target and Initiator (shown in Fig. 2).

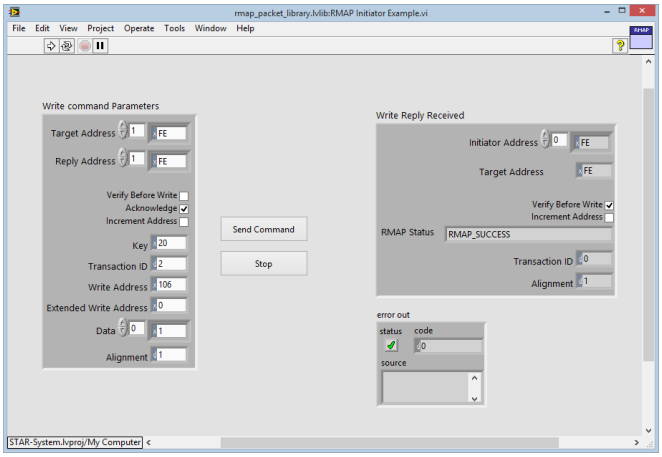

Fig. 2. RMAP Initiator example front panel.

Using the STAR-System wrapper allows LabVIEW applications to share data with other STAR-System processes running on the host. For example, device names set up using the STAR-System Device Configuration GUI can be viewed or modified with changes propagated across all running processes. This can help a user quickly identify and select a desired device to work with without looking up serial numbers.

The complexity of the C API has been abstracted away where possible. No manual memory allocation is required to transmit and receive packets; this is handled by the wrapper with packet data buffers provided as LabVIEW byte arrays. LabVIEW events are used to implement device listeners and transfer completion events.

Performance of the LabVIEW wrapper compares favorably with that of unwrapped STAR-System performance [4] with performance figures roughly the same when transmitting and receiving packets of length above around 60 bytes (Fig. 3). These figures are for a 200Mbit/s link speed, and show performance is close to the maximum theoretical data rate (160 Mbits/s).

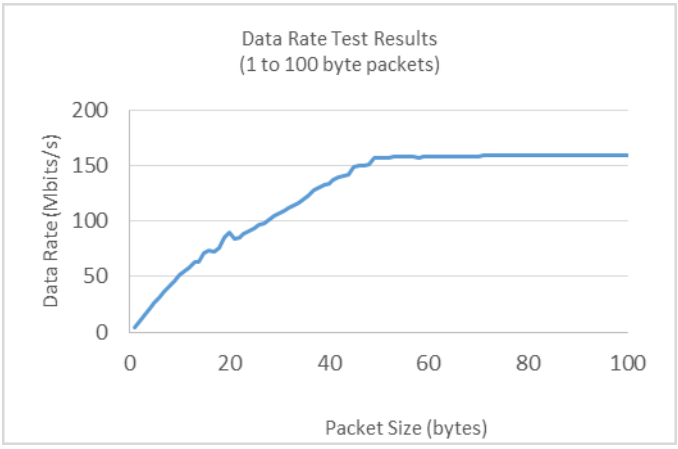

Fig.3. STAR-System wrapper loopback performance.

B. NI-VISA Driver

National Instruments VISA (NI-VISA) provides a standard programming interface between hardware and development environments such as LabVIEW [5]. NI-VISA is supported across the National Instruments product line.

The STAR-Dundee SpaceWire NI-VISA driver has been implemented as a native LabVIEW driver, providing support for the STAR-Dundee PCI family of devices. Software written to control these devices may be deployed on any hardware platform that supports cPCI/PCI and NI-VISA, including both Windows based hosts and LabVIEW Real-Time targets, without requiring modifications to source code. The software is provided as LabVIEW source with password protected block diagrams, allowing users to compile for any target.

The driver allows STAR-Dundee SpaceWire PCI cards to be detected with and controlled by National Instruments' MAX (Measurement and automation explorer) tool (Fig. 4).

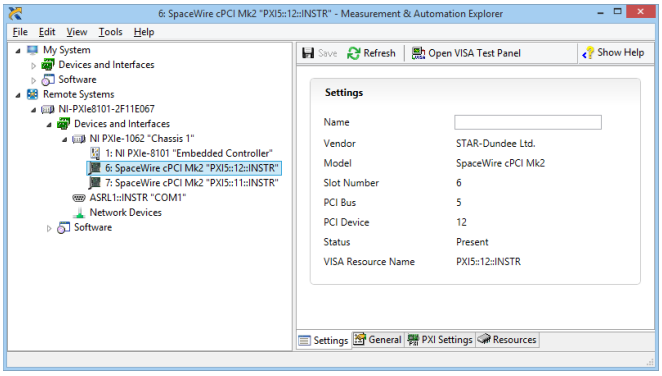

Fig. 4. MAX displaying chassis with cPCI cards

The driver has been designed to be intuitive to work with for LabVIEW users. For example, device access follows the familiar "Open, Perform Action, Close" architecture, with LabVIEW arrays used to pass SpaceWire data to transmit and receive functions. Figure 5 demonstrates the ease of use of this API. This example implements a software loopback device: packets are received on one port of the device, and are then looped back out of another. One could easily extend this example into a useful tool by inspecting the received traffic and permuting it in some way, perhaps by inserting or removing time codes, or injecting errors, before re-transmitting out the other port.

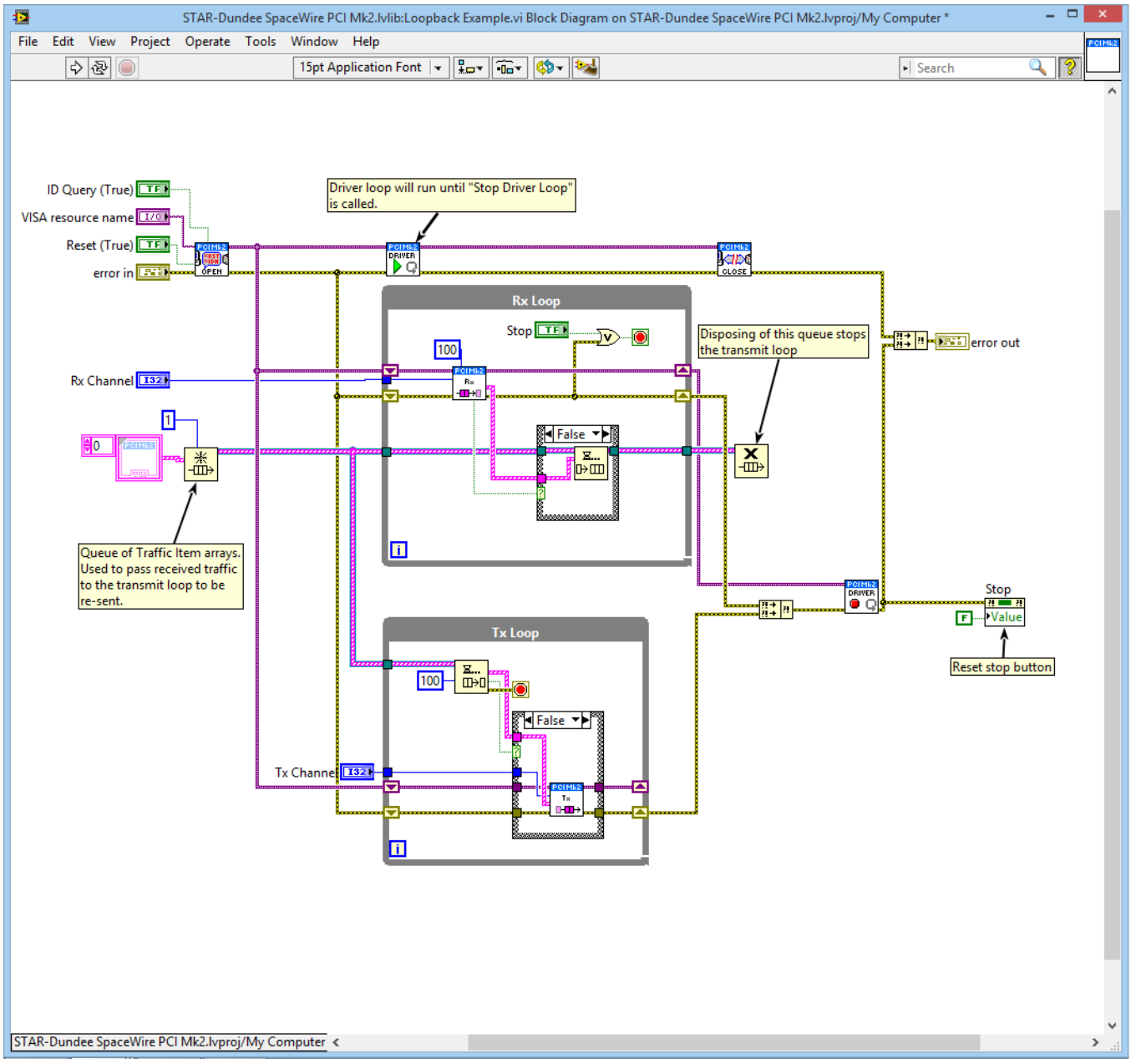

Fig. 5. LabVIEW source code example showing a software loopback application.

IV. Future work

The STAR-System wrapper for LabVIEW supports all functionality provided by the current STAR-System libraries. This wrapper will be continuously upgraded to support any new functionality and released at the same time as new STAR-System releases

The NI-VISA driver is currently capable of transmitting and receiving SpaceWire packets, and configuring SpaceWire links. The RMAP packet library (already provided with STAR-System) will be ported to native LabVIEW code allowing it to be used with the NI-VISA driver on LabVIEW RT targets. Error injection support will also be added, allowing a user to inject, for example, a parity error on a given byte in a data stream, along with all the device configuration operations offered by the STAR-System API. Currently only the cPCI/PCI Mk2 cards are supported by this driver, but a USB driver could be quickly developed by re-using the existing top level API.

V. Conclusion

LabVIEW is a software development platform that allows for rapid development of test and measurement applications. Users of STAR-Dundee SpaceWire equipment can leverage the features of LabVIEW by using ready-built SpaceWire wrapper libraries and drivers in order to reduce the time and cost of developing test and verification tools.

References

[1] National Instruments, http://www.ni.com/labview/ LabVIEW, National Instruments Website.

[2] STAR-Dundee, http://star-dundee.com/sites/default/files/STAR-System.pdf, STAR-System Datasheet, STAR-Dundee Website.

[3] STAR-Dundee, http://www.star-dundee.com/products/star-system-labview, STAR-System for LabVIEW, STAR-Dundee Website.

[4] STAR-Dundee, http://star-dundee.com/knowledge-base/star-system-performance, STAR-System Performance, STAR-Dundee Website.

[5] National Instruments, http://www.ni.com/visa/ National Instruments VISA, National Instruments Website.

Standardisation (Short)

SpaceWire 2: Needs and Evaluation Metrics

NOT PERMITTED TO PUBLISH PAPER

Manchester Coding Option for SpaceWire: *providing choices for system level design*

SpaceWire Standardization, Short Paper

Glenn Rakow
NASA Goddard Space Flight Center
Greenbelt, MD, USA
Glenn.P.Rakow@nasa.gov

Alexander Kisin
NASA Goddard Space Flight Center/AS&D
Greenbelt, MD, USA
Alexander.B.Kisin@nasa.gov

I. *Abstract*— This paper proposes an optional coding scheme for SpaceWire in lieu of the current Data Strobe scheme for three reasons. Firstly, to provide a straightforward method for electrical isolation of the interface; secondly, to provide ability to reduce the mass and bend radius of the SpaceWire cable; and thirdly, to provide a means for a common physical layer over which multiple spacecraft onboard data link protocols could operate for a wide range of data rates. The intent is to accomplish these goals without significant change to existing SpaceWire design investments.

The ability to optionally use Manchester coding in place of the current Data Strobe coding provides the ability to DC balance signal transitions, unlike the SpaceWire Data Strobe coding; and therefore the ability to electrically isolate the interface without additional concerns.

Additionally, because the Manchester coding scheme encodes the clock and data on the same signal, the number of wires in the existing SpaceWire cable could be reduced by 50%. This reduction could be an important consideration for many users of SpaceWire as indicated by the effort currently underway by the SpaceWire working group to reduce the cable mass and bend radius by elimination of shields. Reducing the signal count by half would provide even greater gains.

It is proposed to restrict the data rate for the optional Manchester coding to a fixed data rate of 10 Megabits per second (Mbps) in order to simplify the necessary changes and still able to operate in existing radiation tolerant Field Programmable Gate Arrays (FPGAs). Even with this constraint, 10 Mbps will satisfy many applications where SpaceWire is used. These include command and control applications and instrumentation applications with moderate data rate requirements.

For most NASA flight implementations, SpaceWire designs are implemented using rad-tolerant FPGAs and the desire to preserve the heritage design investment is important for cost and risk considerations. The Manchester coding option can be accommodated in existing designs with only changes to the FPGA.

II. *Index Terms*— SpaceWire, Signal level, Line encoding, Manchester encoding

III. Introduction

Developers of spacecraft using SpaceWire have expressed concern with the inability to electrically isolate the physical interface without possibility for voltage build-up of the signal, resulting in failure of the interface [1]. This is because the SpaceWire Data Strobe (DS) line coding does not have an equal distribution of ones and zeroes over time; i.e., it is not a Direct Current (DC) balanced signal.

Another concern expressed for potential users of SpaceWire is the bend radius and the mass of the cable specified in the original SpaceWire standard [2], ECSS-E-50-12A. Both of these concerns are being addressed by efforts by the SpaceWire working group, but with solutions that are very different than the original SpaceWire standard, which would impede incremental improvements to existing SpaceWire designs, necessary to preserve the cost of the investment.

A simple solution to address these concerns for many applications under 10 Megabit per second (Mbps) would be to modify the line coding portion of SpaceWire design to encode both the clock and data on the same signal. This would halve the number of wires for the interface and provide for a DC balanced line encoding so that electrical isolation could be achieved. The resulting physical interface consisting of a differential pair in both directions may also be used with other DC balanced line coding schemes, such as 8b/10b, so that the interface may be shared with multi-gigabit per second (Gbps) applications or SpaceWire-Real Time (SpaceWire-RT), a new SpaceWire specification that uses 8b/10b line code. For implementations that use a Field Programmable Gate Array (FPGA), this allows hardware to be independent of the data link protocol used.

IV. Original SpaceWire Coding Scheme

The original SpaceWire encoding scheme is Data Strobe (DS), which has several advantages over other encoding schemes because it is simple to implement and provides a variable data rate without negotiation between transmitter and receiver.

A major advantage is that the DS decoding circuit is a trivial asynchronous implementation. Because of the

asynchronous recovery of the DS received clock, the NASA SpaceWire implementation can decode a bit stream that is two and a half times faster than the decoder's local oscillator. This has been an important consideration for flight applications where an asymmetrical link is used, i.e., where data is received faster than it is transmitted. Another scenario is where the SpaceWire implementation is in a one-time programmable FPGA that does not contain clock multiplier circuitry necessary for oversampling and clock resynchronization for high rate data.

Another advantage of the DS encoding is that the frequency components of the two signals (Data and Strobe) are half the frequency of the transmitted bit rate, which results in lower Electro-Magnetic Interference (EMI) emissions when compared to traditional clock and data schemes.

Lastly, unlike the traditional clock and data transmission schemes, DS encoding has a whole bit period of clock to data skew margin versus a half-bit period for the traditional clock and data scheme because the clock is recovered at the receiver by an exclusive OR (XOR) operation.

These advantages have been important and will remain important considerations for spacecraft onboard network designs. However, there are other considerations and applications (described later) that require a trade-off analysis at the system level, and these include electrical isolation of the interface, cable mass, and bend radius for applications where 10 Mbps is sufficient bandwidth. When applicable, it would be beneficial for system engineers to be able to make decisions on a link-by-link basis depending upon what considerations are important for the particular function. This would be possible, if an optional minimal encoding scheme like Manchester is utilized.

V. Manchester Encoding Option for SpaceWire

This paper proposes that the SpaceWire working group consider the standardization of an optional Manchester line encoding scheme for SpaceWire for the reasons stated previously.

The Manchester scheme encodes the clock and data over the same signal and therefore reduces the number of wires by half when compared to the original SpaceWire DS encoding scheme. It also is a DC balanced signal so the interface may be easily isolated with either a transformer or in-line capacitors.

Inherent to Manchester codes, it always performs a mid-bit period signal level transition to indicate the logic value. The logic value is encoded by the direction of the level transition, either high-to-low or low-to-high transition, to encode either a logic one or zero depending upon the particular Manchester code.

Manchester codes also have a level transition at the beginning of the bit period if the previous logic value (bit) is the same as the current logic value. However, if the current logic value is different than the previous logic value, there is no signal level transition at the beginning of the current bit period.

The trick is to determine which transition is the beginning of a bit period or a mid-bit transition. The Manchester decoding is trivial to accomplish for a moderate fixed data rate (10 Mbps) application in a typical rad-tolerant FPGA without clock multiplier circuitry.

Popular protocols that use Manchester codes are MIL-STD 1553, which uses Manchester II Bi-Phase L coding at a low data rate of 1MHz and 10Base-T Ethernet (802.3), which uses a Manchester code at 10MHz. MIL-STD-1553 and 10Base-T Ethernet use Manchester codes that have opposite voltage levels.

The key advantage with SpaceWire using a Manchester option have over MIL-STD-1553 and 10Base-T Ethernet is that is has 10 times the bandwidth of MIL-STD-1553, and it is a simpler and less complex protocol with a smaller packet header compared with 10Base-T Ethernet.

Additionally, the Manchester encoded SpaceWire option will provide for a single network protocol to unify the other existing SpaceWire options, which include, SpaceWire-RT and SpaceFiber (multi-Gbps protocol) that may be run over copper (instead of fiber) as well as the original SpaceWire.

Either Manchester coding options could be adopted by the SpaceWire working group.

VI. Use Cases

The primary use cases for the SpaceWire Manchester code option would be for those applications that only need 10 Mbps of bandwidth and require an electrically isolated interface. In addition, Manchester coding would be suitable for applications where the routing of the electrical harness is challenging because of space constraints and a need for a thinner cable that provides a tighter bend radius (assuming the bandwidth requirement of less than 10 Mbps is acceptable).

The electrical isolation may be required to prevent the destruction of a Low Voltage Differential Signal (LVDS) transceiver. This could occur by the propagation of a failure through an intermediate shared cross-strapped connection for a critical function because of a power supply failure in another unit.

Other possibilities are to prevent a latent electrical failure due to an Electrostatic Discharge (ESD) event, or to increase the margin for common mode voltage range of the transceiver, or to reduce the signal noise back to the connecting system that is sensitive to conducted noise.

These applications are directly applicable for the implementation of the NASA SpaceAGE Bus electrical interface specification.

The SpaceAGE Bus is an electrical specification to connect board level components within an avionics box [3]. Unlike the traditional backplane, the SpaceAGE bus defines point-to-point electrical interfaces to integrate avionics board level functions by cabling together mechanical card frame enclosures that house electronics boards to form avionics box functions. The SpaceAGE Bus defines a complete set of physical interfaces that are independent of protocol, including communication, clock, analog, power, and more, that are typical for space avionics backplanes.

The rationale for SpaceAGE Bus is to reduce the non-recurring Engineering (NRE) development of avionic systems through the elimination of "glue" elements such as backplane,

low voltage power supply (LVPS) and mechanical chassis that change depending upon the number and arrangement of electronic board functions. Because the SpaceAGE Bus board level functions have electrical interfaces that are isolated, network attached with primary power input, they are like independent boxes themselves, and allow for new configurations to be easily connected together with greatly reduced NRE.

This is one reason why the standardization of a common physical layer, independent of protocol is important for NASA, because one set of interfaces can be used for a wide range of requirements.

For example, spacecraft onboard communications have numerous differing requirements from kilobit per second (kbps) data rates up to Gbps data rates depending upon their application. Examples of kbps applications include board functions that control power for heaters or solenoid position values for propellants, to low rate telemetry collection of temperature and other engineering data, etc. Examples of Gbps applications include memory operations between a processor and a high data rate instrument or a Solid State Recorder (SSR); or from SSR to a Digital Signal Processor; or high-rate down-link from SSR to a downlink function, etc.

VII. Implementation Considerations

The data rate required for an application's SpaceWire link will determine the technologies necessary to implement the SpaceWire protocol. The Manchester encoder function is a straightforward implementation that only involves the Exclusive-Or (XOR) Boolean function of the clock and the data represented as non-return-to-zero (NRZ).

The decoder for the Manchester code, however, requires oversampling of the encoded waveform and comparison to known synchronization value (SpaceWire NULL character) to acquire the bit period boundaries and the mid-period transition used to reconstruct the NRZ data.

Because of the fixed 10 Mbps data rate for the SpaceWire Manchester coding option, the implementation is straight-forward for clock frequencies typical for a rad tolerant FPGA without clock multiplier circuitry. This is significant because low complexity and design heritage are key considerations for many electronic board functions for spacecraft command and control electronics, which perform actuator functionality and low rate housekeeping data telemetry collection. These types of functions are typically redundant and the isolation of the electrical interface is an important consideration for cross-strapped redundancy. Additionally, many instrument functions require less than 10 Mbps bandwidth, and this reduces the complexity for their data link protocol implementation as well.

Since the SpaceWire protocol requires the link to start-up at 10 MHz, there is no change of frequency for the Manchester encoding option, which also simplifies the implementation.

There are many publications for how to decode Manchester encoded data. The focus here is using radiation tolerant FPGAs that NASA typically uses. This would necessitate performing the Manchester decoding without clock multiplier circuitry.

One Manchester decoder method that requires very few flip-flops and logic gates uses a decoder local clock that is asynchronous to the received Manchester waveform. This implementation requires a nominal eight times (8x) clock of the received data rate, but the receiver local clock may be as low as five times (5x) clock, but no more than twelve times clock (12x)[4]. This method also filters out edges after a valid transition is detected, minimizing the effects of noise on the signal. Since 10 MHz is selected, an 80 MHz oscillator is well within the margin of a radiation tolerant FPGA without clock synchronization logic [5].

There are additional Manchester decoding options, including one that utilizes additional logic but uses the same decoder clock (10 MHz) as the receive data rate to create four phases with which to sample the received Manchester data[6][7]. Even though it implements a lower clock frequency and it has a good tolerance toward input jitter and receiver/transmitter frequency mismatch caused by oscillator tolerance differences, it is significantly more complex than the previously described circuit and requires more logic.

The SpaceWire specification allows the 10 Mbps receive data to be +/- 1 Mbps (or from 9 Mbps to 11 Mbps), both of these previously described decode methods support this difference in frequency but the eight times (8x) implementation maintains lock easier and is the method simulated for the NASA application.

Still, there are numerous other methods for decoding Manchester waveforms with and without clock synchronization logic and the implementation details described above were provided as a cursory survey of options.

VIII. Preservation of SpaceWire Investment

The decoding schemes referenced previously are sufficient to decode SpaceWire Manchester encoding at a fixed 10MHz without the use of a clock multiplier. Regardless of how the Manchester decoder is implemented, the important part is that the heritage of existing SpaceWire designs can be preserved. Because the changes required implementing the Manchester encoding only involve the signal layer of the SpaceWire specification (where the encoding is specified), the remainder of the SpaceWire design may stay the same.

For many users, like NASA, this is an important consideration as millions of dollars of NRE have been expended across multiple missions to develop, debug, and refine the SpaceWire design, including verification environments and test equipment. For example, the NASA SpaceWire design heritage spans over a decade with the missions of Swift, JWST, LRO, LCROSS, GOES-R, MMS, and GPM. Additionally, the NASA SpaceWire design has been provided to well over 100 companies, and much feedback has been received concerning problems which have been fixed throughout this time, adding additional value to the design. This makes it compelling and difficult to completely abandon the existing SpaceWire design for new solutions that are not incremental in nature.

IX. SpaceWire Corrective Efforts

The SpaceWire working group has also been exploring solutions to fix SpaceWire, especially in the Quality of Service (QoS) realm to prevent blocking on the network. The solution has the side effect of providing a DC balanced line code, which could be used to electrically isolate the SpaceWire interface. This new protocol called SpaceWire-RT [8], uses an 8b/10b line coding that is used by most multi-Gbps protocols. However, SpaceWire-RT is intended for SpaceWire data rates of 2 to 200 Mbps. The problem with this approach is that it discards the design investments accumulated with the original SpaceWire design and, in its place, proposes a more complex and larger design solution within a typical rad tolerant FPGA when compared the Manchester option. It is therefore not viewed as an incremental approach in the near term. It does however; provide a means to define a common interface for a wider range of data rates, i.e., interfaces that both the multi-Gbps SpaceFiber and SpaceWire-RT can utilize. SpaceWire-RT is seen by the authors as a long term solution, and one where additional complexity can be accommodated and where new design investment is acceptable.

Independently, the SpaceWire working group have also been working on defining a lower mass SpaceWire cable, which will also reduce the bend radius by the elimination of some shields [2].

These are important efforts. It is the position of the authors that an incremental approach to change that maintains as much backward compatibility to the original SpaceWire to be a more practical solution, especially given how difficult it is to insert new technologies into missions because of the risk adverse posture of space mission projects.

X. Summary

This paper presented an incremental design approach option to improve SpaceWire, yet leverages most of existing FPGA based SpaceWire designs for moderate data rate applications that require electrical isolation. It also describes an additional way to further reduce the mass and bend radius of the SpaceWire cable for applications that are tight on space. Additionally, it provides a means to specify a common physical layer and which could work with any protocol that uses a DC balanced line code, such as 8b/10b (used for multi-Gbps protocols). Overall, this approach provides options for system engineers to optimize system level designs.

Acknowledgment

The authors would like to thank J. Fraction for his help in writing, simulation and testing efforts, as well as E. Gorman for mechanical development of the SpaceAGE Bus specification.

References

[1] M. Suess, J. Ilstad, and W. Gasti, “Galvanic Isolation of SpaceWire Links: Requirements, Design Options and Limitations”, 13th SpaceWire Working Meeting, 14-15 September 2009

[2] J. Ilstad, “Low Mass SpaceWire Cable”, 16th SpaceWire Working Meeting; 22 March 2011

[3] A. Kisin, G. Rakow, E. Gorman, “SpaceAGE Bus: New Avionics Building Block Concept” , 4th International SpaceWire Conference 2011

[4] Xilinx Magazine: Manchester Decoder in 3 CLBs: http://www.xilinx.com/publications/archives/xcell/Xcell17.pdf

[5] Actel RTAX-S/SL FPGA Datasheet: http://www.actel.com/documents/RTAXS_DS.pdf

[6] Xilinx AppNote XAPP224: Data Recovery: http://www.xilinx.com/support/documentation/application_notes/xapp224.pdf

[7] Xilinx AppNote XAPP225: Data to Clock Phase Alignment: http://www.xilinx.com/support/documentation/application_notes/xapp225.pdf

[8] S. Parkes, A. Ferrer,Y. Sheynin, SpaceWire-RT Project and Baseline Concepts, 17th SpaceWire Working Meeting, 14 December 2011

SpaceWire Control Codes in SpaceWire, GigaSpaceWire and SpaceFibre Networks

Standardisation, Short Paper

Elena Suvorova, Ludmila Koblyakova, Evgeny Yablokov
Institute of High-Performance Computer and Network Technologies
St. Petersburg State University of Aerospace Instrumentation
SUAI
St. Petersburg, Russian Federation
suvorova@aanet.ru, liudmila.koblyakova@guap.ru, evgeny.yablokov@guap.ru

Abstract—important tasks for any on-board communication network are time synchronization (devices work synchronization) and transmission of low-frequent hard real-time signals, alarm and critical commands. In the SpaceWire standard there are time-codes and distributed interrupts propagation mechanisms for these purposes. The SpaceFibre is a very high-speed serial link for on-board communication; it carries SpaceWire packets over virtual channels. For broadcast propagation of control information SpaceFibre provides the broadcast messages service which may be considered as similar to SpaceWire time-codes for every virtual channel. The GigaSpaceWire has been developed to enhance link characteristics for SpaceWire networks. In the paper we consider transmission of SpaceWire control codes through the communication onboard network where SpaceWire, GigaSpaceWire and SpaceFibre technology are used together; comparison of SpaceWire control codes (distributed interrupts and time-codes with SpaceFibre broadcast messages.

One of the main problem in above-stated standards is the problem of transmission of SpaceWire C-codes. In GigaSpaceWire links these codes are implemented by the pair of symbols — descriptive 8b/10b K-code + C-code itself. In SpaceFibre this code could be implemented by a new 4 byte symbol, which does not go to the Retry level. So at the network level in a routing switch the control-code propagation will be similar to SpaceWire standard but with another time characteristics.

Index Terms—real-time signalling, Distributed Interrupts, Time-codes, standardisation.

I. INTRODUCTION

The important tasks for on-board distributed communication network are time synchronization (device work synchronization) and also a task of informing devices about of certain single events in a system in hard real time, for example failure of some devices or readiness for some action, and required signals with and without acknowledges, [1, 2]. For this purpose a hard real time signals are required.

In modern on-board distributed communication networks the technology is used in which the transmission of data and control traffic are union and transmit over the same links. The basic modern on-board communication standards are SpaceWire—/GigaSpaceWire [3, 5] and SpaceFibre [4]. Transmission of control codes in hard real time in on-board system based on these three technologies, is important task. In this paper we consider and compare mechanisms of control code transmission in these standards.

In SpaceWire and GigaSpaceWire for hard real-time signal transmission the distributed interrupt and time-codes mechanism are used. In SpaceFibre for control code transmission the broadcast messages mechanism is used. In practice it can be required to build networks, where all three technology are used, so the important task is providing opportunities for hard real-time signal transmission through such networks.

II. HARD REAL TIME SIGNALS

A. Hard real time signals classification

In general hard real time signals can be separated in two classes:

- Synchronous signals are signals which value depends on a previous signal value and form a sequence of logically dependent events with a certain period. Their general purpose is synchronization of devices' work and providing a common time in all system, etc.
- Asynchronous signals are single signals which do not depend on previous signals. Such signals are necessary for informing devices about single critically important events in real time.

Synchronous signals also can be periodic and aperiodic, asynchronous signals can be with or without acknowledgement.

Consider the control codes in SpaceWire, GigaSpaceWire and SpaceFibre, from the point of this classification.

B. Time-codes propagation mechanism in SpaceWire and GigaSpaceWire

Time-codes distribution mechanism relates to transmission of synchronous signals.

At the symbol level under the time-codes there are allocated special symbols. These symbols have the highest priority, (higher than other control codes and data characters

have), that allows to transmit time-codes fast in condition of high network load by data transmission, and provide time-code transmission without delays through the loaded or blocked by data paths. For time codes six bits are allocated that allows to encode 64 subsequent codes.

A time-code source is a node (the source node), and all network devices which receive the code can handle it. The time-codes are distributed by broadcasting. When any device has received the time-code, it compare the code with the stored in the device value and if the new value is one greater than previous (it corresponds to correct value), than the devise stores it and sends the code to all ports, the incoming port excluding. If the time code is incorrect than the device overwrites its value but does not transmit further. This comparison determines the dependence of every subsequent code from successful/unsuccessful transmission of the previous code. In Fig.1 the example of time-code propagation is shown. The digit inside is the current time-code value.

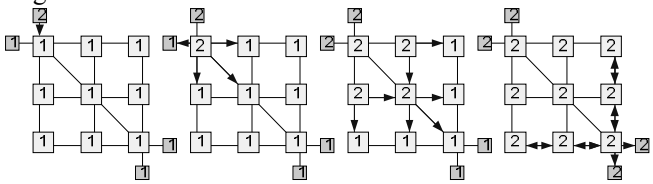

Fig. 1 Example of time-code propagation

C. Distributed interrupt mechanism in SpaceWire and GigaSpaceWire

The distributed interrupt mechanism allows to transmit asynchronous signals with and without acknowledgement.

At the symbol level of the SpaceWire and GigaSpaceWire protocol stack for distributed interrupt and acknowledge codes special symbols are allocated – Interrupt-codes and Acknowledge-codes. These symbols have the higher priority than data symbols have; it allows fast transmission of distributed interrupt in case of strong network load by data symbols and provides Interrupt-codes transmission without delays through the loaded or blocked by data paths. The higher priority has only time-codes but the network load of time-codes is low and has limited influence on the Interrupt-code propagation time. For the Interrupt-code 5 bits are allocated that allow to encode 32 distributed interrupts. There are two possible mode of distributed interrupt mechanism: mode with acknowledge and mode without acknowledge.

In general case the sources and handlers of the Interrupt-codes are terminal nodes. Interrupt-codes and Acknowledge-codes are broadcasted to all network nodes. For protection from retransmission in a network with cycles every node and router has a 32-bit ISR register, the bit *i* of which corresponds to the Interrupt-code type with number *i*. When a certain event has happened in a node, the Interrupt-code with corresponding to this event 5-th bit code is formed. Then the ISR checks and if the corresponding bit is equal to zero, it is set to one and Interrupt-code is sent to the network. If the bit is already set to one it means that Interrupt-code with the same identifier has been already sent to the network and acknowledgement has not been received yet, so the Interrupt-code is not sent again. When the Acknowledge-code is received the corresponding bit of ISR is set to zero.

When a router receives the Interrupt-code it also checks the corresponding bit in ISR. If it is equal to zero, it sets it to one and the Interrupt-code is sent to all output ports excluding the incoming one. If the bit in ISR is equal to one, then the Interrupt-code is ignored and is not sent further; it is necessary for protection from endless time-code retransmission in networks with cycles.

In case of using mode without acknowledge for clearing the ISR bits after sending of Interrupt-code and in case of Interrupt-code has been lost, the timeout T_{Reset} for every bit of ISR is used. It allows automatically cleaning the bit of ISR if the acknowledge has not been received in time, and thus recover registers for the next Interrupt-code transmission. Also for protection of crossing Interrupt-codes and Acknowledge-codes waves the timeout $T_{ISRChange}$ for every ISR bit is used, which does not allow to change the ISR bits earlier than certain time has elapsed.

In Fig. 2 the example of Interrupt-code propagation of one type is shown. The digit inside is a value of the correspondent ISR bit.

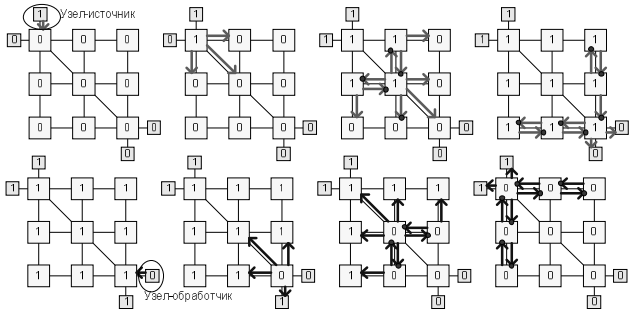

Fig. 2 The example of Interrupt-code and Acknowledge-code propagation

D. The of Broadcast messages mechanism in SpaceFibre

The SpaceFibre standard together with the data packet transmission service provides the service for messages broadcast (analog of control codes) transmission that is responsible for broadcast propagation of short messages (8 bytes) to all network nodes. These messages can transmit time and signals of synchronization and can be used for indication about different events in a network.

The interface of the SpaceFibre broadcast channel codec consist of registers set for writing broadcast message parameters, and the same set for reading parameters of incoming broadcast messages.

Broadcast message has the following parameters:

- Broadcast channel number,
- Sequence number B_SEQ,
- type,
- data,
- late flag.

Sources of broadcast messages are nodes. Recipients are all other nodes and routers. When in a user application it is necessary to send a broadcast code, their parameters are defined and request is transmitted to the SpaceFibre port

interface TX_BROADCAST.request (Broadcast Channel, Broadcast Sequence Number, Broadcast Type, Late, Message), which initiate transmission of the broadcast message through the SpaceFibre port. When the broadcast message is sent the broadcast sequence number is incremented by one.

When a router receives a broadcast code it checks its broadcast number B_SEC with the current value in the device for given broadcast channel and determines if the received code is correct or not. Incoming code is correct if its sequence number is one more than the current value. When the correct code is received the current sequence number is incremented by one and the code is transmitted to all output ports excluding the incoming port. If the sequence number is incorrect then the code is not transmitted further. So in a network with cycles a repeatedly incoming code is not transmitted further. Also it means that broadcast messages are synchronous messages because the transmission of the next code depends on the previous code.

A broadcast frame format is shown in Fig.3.

0 7	8 15	16 23	24 31
COMMA	SBF	BC	B_SEQ#/B_TYPE
DATA 1 LS	DATA 1	DATA 1	DATA 1 MS
DATA 2 LS	DATA 2	DATA 2	DATA 2 MS
EBF	RSVD/LATE	SEQ_NUM	CRC

Fig. 3 Broadcast frame format

A Broadcast frame starts with the control SBF (Start Broadcast Frame) word and finishes by the EBF (End Broadcast Frame) word. The BC (Broadcast Channel) field identifies broadcast channel of transmitted message. The B_SEQ#/B_TYPE field contain two subfields: 3 bits for sequence number B_SEQ (7:5) and 5 bits for message type B_TYPE (4:0). The broadcast sequence number field contains incrementing value which is specified for the broadcast channel. Every broadcast channel has its own broadcast sequence number which is used for broadcast frame propagation through the SpaceFibre network. The type field defined broadcast message type and the semantic of the following 8 data bytes.

At the end of the broadcast frame there is a RSVD/LATE field, which contain 7 reserved bits and 1 bit is a flag LATE, which is set to one if the code was resent at retry level. It is used for informing a receiver node that given broadcast frame has been delayed as a result of one or several retransmission. If the broadcast message contains the time for synchronization than user application can decide to ignore it because of late delivery, or the broadcast message may contain information about some event, which still can be useful for the application despite delay.

The sequence number SEQ_NUM at the end of frame is used for supporting retransmission at the Retry level. 8-bit CRC cover fields from SBF to EBF.

For monitoring and limitation used by the broadcast message amount of link bandwidth with the broadcast mechanism should be associated one Broadcast Bandwidth Credit Counter for all broadcast channels. It should monitor and control the aggregate bandwidth of all broadcast channels. The control parameter, which is called the Expected Broadcast Bandwidth Percentage, should define a portion of the link bandwidth, which is reserved for a broadcast message including the overhead of the broadcast frame delimiters. If the allocated percentage of bandwidth is already used, the broadcast messages will not pass.

Broadcast messages are synchronous signals within the same broadcast channel because for every broadcast channel there is own counter for the broadcast sequence number, which increments for every new message in the broadcast channel; the message is correct if its number is one more than the previous one. The type defines the data semantics.

III. Mechanism of broadcast messages in SpaceFibre and Control Codes in SpaceWire and GigaSpaceWire

E. Mechanism of broadcast messages in SpaceFibre and Time-codes in SpaceWire and GigaSpaceWire

Their mechanisms use synchronous messages without acknowledge.

In SpaceWire there is only one channel and it uses very compact six-bit sequence number; there are no additional fields for type and data. Because of it little jitter of control code propagation in a network is provided, that is very important characteristics for synchronization.

In SpaceFibre the size of broadcast message is substantially greater, so the propagation time of broadcast message over the link will be greater than in GigaSpaceWire and SpaceWire. Due to the large number of broadcast channel, and respectively the greater number of propagating broadcast codes at the same time in a network, the waiting time in a broadcast code transmission queue can be large.

Also in SpaceFibre there is the Retry level and retransmission, so control code can be delayed in a retry buffer for indefinite time, that makes broadcast propagation time much less predictable, whilst it is a critical characteristic for hard real-time signals. On the other hand, even delayed code arrival can allow to receive correctly the next code. However if the code delay was quite big and the correct code reaches the destination by another path in network cycles, and the next correct code has already sent, then the appearance of the old code can spoil the broadcast sequence number and thus the next correct code can be erroneously dropped. These situations require a separate investigation.

So the time-code mechanism in SpaceWire and GigaSpaceWire is much more fast and simple but with limited functionality (only one channel and six bit for sequence number, without type and data field); it corresponds to hard real-time requirements. The broadcast messages mechanism in SpaceFibre is more complex, with great features, but slower. It is good to have both mechanisms, they complement each other. The time-code mechanism is for more accurate synchronization (as a main synchronization in a system), for situations, where jitter and code delivery time are critically important, for using it for hard real-time. For other cases, where hard real-time is not required, the SpaceFibre broadcast messages because it

more flexible and give much more features (due to type/data fields for every message).

F. Transmission of asynchronous signals in SpaceFibre and distributed interrupt mechanism in SpaceWire and GigaSpaceWire

The main aim of the distributed interrupt mechanism is transmission of different asynchronous signals set with or without acknowledge in a hard real-time mode.

The broadcast messages mechanism by the message transmission type is a synchronous and without acknowledge, it has type and data field, which allow transmitting big amount of different messages. To use the broadcast messages mechanism "as is" for asynchronous signal transmission is not possible.

The broadcast messages mechanism is a synchronous within the same broadcast channel and there could be 256 channels , so it is possible to send 256 independent from each other massages. It is possible to consider such messages, in general, as asynchronous ones, because they are independent from each other. Every channel can correspond to one message type (Interrupt_Identifier), and for implementing of acknowledges it is possible to use type and data field of broadcast message. The reliability and time characteristics of such asynchronous messages transmission by using broadcast messages will be significantly worse than distributed interrupt mechanism's characteristics in SpaceWire and GigaSpaceWire and will not correspond to hard real-time requirements:

- Big overheads in comparison with distributed interrupts.
- Error recovery time will be longer because timeouts which allow to distributed interrupt mechanism recover the initial register values after errors, have to be implemented in software over broadcast messages mechanism.
- Dependence from the previous errors. For example, the transmission of the current code can be indicated as a fault because the previous code of the same type was lost and the sequence number is not incrementing in the part of network; for their recovery can be required to send several codes (the number of codes depends on network topology and the place of error). For example if the network has the tree structure (the worst case), then for the sequence number recovery there are needed as many code sending, how many levels there are in the tree (the shortest path length). The correct codes will not reach the destination only because of the previous error. The existence of retry level broke the main principle of asynchronous signal transmission (by its dependence from the previous codes). It makes impossible using of broadcast message mechanism for asynchronous signals transmission in hard real-time, because everything will be good only if there is no errors. And also there is a dependence on the network structure.
- The Retry level makes unpredictable the message delivery time. In the distributed interrupt mechanism in the mode with acknowledge the all timeouts and parameters values depend on estimation of maximally possible code propagation time in the worst case. This time should be estimated taking in account possible retransmissions at the Retry level. If there are many cycles, then the retransmission may severely degrade the mechanism work. For example if the code has been delayed in some device's buffer due to disconnections, but reaches the all other nodes by other paths and the acknowledge has been already sent, and after that from the retry level the old code has been sent, it only damages the sequence number and the next code will not reach all network nodes.
- The bandwidth limitation for broadcast message propagation (broadcast percentage parameter) also can cause the control code delay or loss.
- It will be difficult in administration.

Thus it is clear that it is possible to use the broadcast messages mechanism only in not hard real-time mode.

To enable in SpaceFibre the asynchronous messages transmission in hard real-time mode, the non-standard implementation of distributed interrupt mechanism is done.

IV. NON-STANDARD IMPLEMENTATION OF THE DISTRIBUTED INTERRUPTS AND TIME CODES MECHANISMS IN SPACEFIBRE

In devices, which have been implemented in collaboration with the "ELVEES" company, the non-standard implementation of time-codes and distributed interrupt mechanisms were added, similar to SpaceWire and GigaSpaceWire.

At the level before the Retry level for time-code and interrupt/acknowledge code the special symbols are allocated, which perform similar to time-codes and Interrupt-codes and Acknowledge codes in SpaceWire. It allows to solve several tasks:

- asynchronous signals transmission in SpaceFibre;
- time-codes mechanism with smaller jitter;
- supporting the time-codes and distributed interrupt mechanism in a networks, where at the same time the SpaceWire, SpaceFibre and GigaSpaceWire are used.

In Fig.4 the new, additional Control code layer is shown.

The new Layer can send and receive the CCode of the SpaceWire network. The main difference from broadcast layer – the messages on the Control code layer are flowing much quicker through the SpaceFibre network, thus making the CCodes of Spacewire reasonable. The CCodes of SpaceWire network are inserted between the Retry and the Lane layers of SpaceFibre. The CCodes are lower than the Retry layer, so no error is detected on the Retry layer.

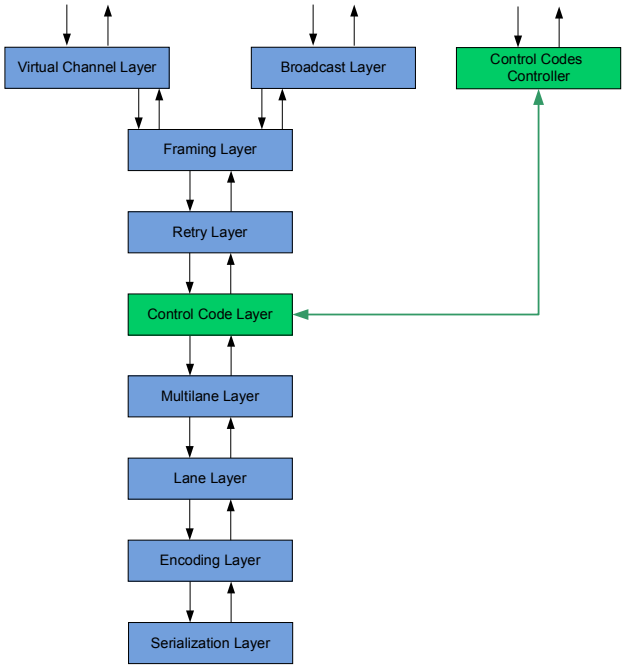

Fig. 4 New Control Code Layer

V. The Time Characteristics of Control Codes in SpaceWire, GigaSpaceWire and SpaceFibre

Let's estimate the minimum possible control code transmission time in the network.

We assume that the local frequency of router's work are the same and equal to 125 MHz. The transmission speed in SpaceWire network is 400Mbit/s, transmission speed in GigaSpaceWire and SpaceFibre is 1250 Mbit/s. For estimations of distributed interrupts and acknowledge codes, which have the priority less than the time-codes priority, we assume that they are transmitted at the moments when the time-codes are not transmitted in a network.

Dependence of the minimal transmission time from the number of routers in a network is shown in the Fig 5.

As can be seen from these grapfhics the time-codes and distributed interrupt propagation time for all network types are realy close to each other. The distributed interrupt codes propagation time is more than time-code propagation time at 7-8% bacause of their handling in a router requires more number of actions then for time-codes.

The minimal broadcast code propagation time on the overage at 1,7 times greater then time-codes and distributed interrupt propagation time. That is because of broadcast codes have bigger length and at every data link the CRC is checked.

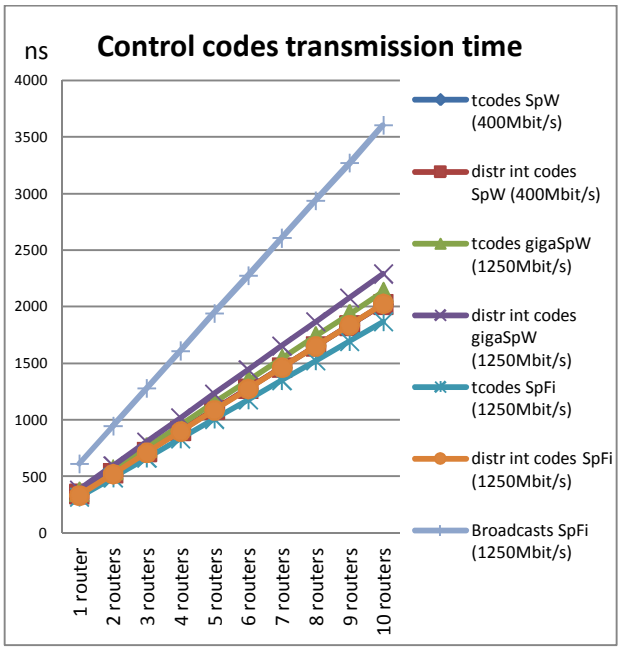

Fig.5. Graphic of minimal transmission time dependence from the number of routers in a SpaceWire, SpaceFibre and GigaSpaceWire network

VI. Conclusion

In the paper we consider the main space standards for onboard communication networks - SpaceWire, GigaSpaceWire and SpaceFibre. There is given a classification of real time signals, which are required for control code transmission – distributed interrupt and time-codes mechanisms and broadcast messages. It is considered the implementation of the control-code in main standards and their short comparison is made. Overview of the non-standard control code implementation in SpaceFibre is given.

References

[1] S, Parkes D 1.1 Consolidated set of Requirements for SpaceWire-RT, SpaceWire-RT Consortium

[2] S, Parkes, "D2.1 SpaceWire-RT Outline Specification," SpaceWire-RT Consortium, September 2012.

[3] ECSS-E-50-12C. SpaceWire - Links, nodes, routers and networks. - European Cooperation for Space Standardization (ECSS), 31 July 2008

[4] S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, "SpaceFibre Standard Draft F3", University of Dundee, 10th September 2013

[5] Evgeny Yablokov, Yuriy Sheynin, Elena Suvorova, Alexander Stepanov, Tatiana Solokhina, Yaroslav Petrichcovitch, Alexander, Glushkov, Ilia Alekseev, "GigaSpaceWire – Gigabit Links for SpaceWire". Networks Proceedings of the 5th International SpaceWire Conference. Gothnburg 2013.

Missions & Applications (Long)

Flight equipment Validation with iSAFT: The EUCLID Fine Guidance Sensor case

Missions & Applications, Long Paper

Vangelis Kollias, Nikos Pogkas, Antonis Tavoularis, Michalis Tsagkaropoulos
Teletel S.A.
Athens, Greece
{V.Kollias, N.Pogkas, A.Tavoularis, M.Tsagkaropoulos}@TELETEL.eu

***Abstract*— iSAFT is an integrated powerful HW/SW environment for the simulation, validation & monitoring of satellite/spacecraft on-board data networks supporting simultaneously a wide range of protocols. This paper presents a study on how iSAFT modules can be used for the validation of demanding spacecraft subsystems such as the EUCLID Fine Guidance Sensor (FGS). Validation of the EUCLID FGS includes the accurate injection of static sky images through SpaceWire and acquisition of the units' response through another SpaceWire and MIL-STD-1553 channels and, synchronization with the AOCS SCOE which provides quaternion and angular rate for dynamic simulations and remote control of other testbed elements (e.g. OGSE). The paper presents the proposed EGSE HW and SW architectures, their configurations and the performance characteristics which pose very strict requirements on the design of the EGSEs.**

***Index Terms*— SpaceWire, FGS, iSAFT, validation, 1553, FMEA, IRIG.**

I. INTRODUCTION

EUCLID is an ESA mission which aims to map the geometry and nature of the dark Universe by investigating the distance-redshift relationship and the evolution of cosmic structures. To meet the high precision imaging requirements, one of the most crucial components of the satellite is the Euclid Attitude and Orbit Control System (AOCS). AOCS is a high precision control unit that is used for the provision of stable pointing for visual exposure.

One of the main components of the AOCS is the Fine Guidance Sensor (FGS) required to satisfy the mission's pointing requirements. The FGS consists of three electronic modules: a detector which acquires raw sky images, a module that process them and a unit which uses the images and a star catalogue to calculate accurate pointing information.

In this paper, the validation requirements for the EUCLID FGS are presented and analysed as an example of how TELETEL's iSAFT integrated environment can be used to address the validation needs of complex flight equipment.

iSAFT is an integrated powerful HW/SW environment for the simulation, validation & monitoring of satellite/spacecraft on-board data networks supporting simultaneously a wide range of protocols (RMAP, PTP, CCSDS Space Packet, TM/TC, CANopen, etc.) and network interfaces (SpaceWire, ECSS MIL-STD-1553, ECSS CAN). It is based on over 20 years of experience in the area of protocol validation in the telecommunications and aeronautical sectors, and it has been fully re-engineered in cooperation with ESA & space Primes, to comply with space on-board industrial validation requirements (ECSS, EGSE, AIT, AIV, etc.). iSAFT is also highly modular and expandable to support new network interfaces & protocols (Fig. 1).

The iSAFT environment consists of COTS and in-house made hardware subsystems (such as communication interfaces like SpaceWire, MIL-STD-1553, CAN boards, power subsystems, specific I/O subsystems, etc.) plus the lower and higher layer software.

The iSAFT Software tool chain is composed of the following general parts:

- The iSAFT Console which is based on a state of art windowing graphical user interface, which provides to the operator easy configuration, control and monitoring capabilities, plus additional tools for traffic logs display and management as well as a test management and execution environment (iSAFT TestRunner).
- The iSAFT Runtime Environment (RTE) containing modules that perform simulation, monitoring and data processing using the underlying physical interfaces. iSAFT RTE provides service interfaces for all containing modules providing a scalable and fully distributed framework that can be deployed in multiple iSAFT stations and provides LAN remote control.
- General Application Management modules including the configuration management, common logging functionality, self-tests and diagnostic functions as well as internal Database management.
- Protocol Modules for command and control with external EGSEs, SCOEs or the CCS. The Hardware Abstraction Layer which provides an abstraction to the underlying Driver APIs being able to change the Boards with different ones while keeping higher layer software independent.
- The Drivers and API libraries of the Interface Cards, the DAQ system, the Power supply equipment and the DC Electronic Load equipment.

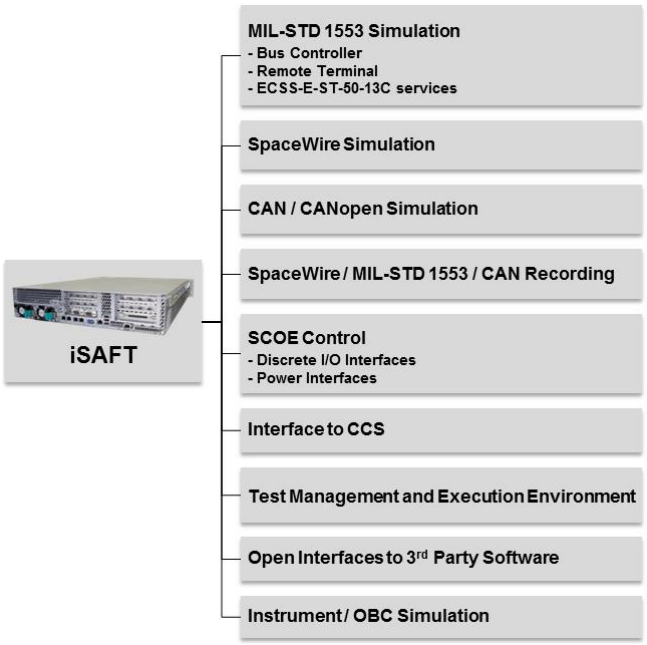

Fig. 1. iSAFT features

The following sections present the study on how you can build a complete EGSE based on the iSAFT simulation, time-stamping and synchronization capabilities in order to validate the EUCLID FGS subsystem. Validation can be performed at each FGS unit individually and at the entire FGS integrated system. The main concept of the validation is the injection of static sky images through SpaceWire and acquisition of the units' response through another SpaceWire/MIL-STD-1553 channel, as well as synchronization with the AOCS SCOE which provides quaternion and angular rate for dynamic simulations and remote control of other testbed elements (e.g. OGSE).

II. EUCLID FGS EGSEs Design Approach

A. Overview

The FGS is a sensitive camera (star sensor) that provides dedicated, mission-critical support for the EUCLID's AOCS system by providing the AOCS with the high accuracy attitude measurement required to meet the demanding pointing performance during science observation. The proposed EGSE solution is based on the existing iSAFT Protocol Validation System (PVS) product instances, and on specific extensions in order to meet the EUCLID FGS EGSEs requirements [4,5,6,7].

The FGS is composed of a Focal Plane Assembly (detectors and detectors support structure) and the Proximity Electronics Module (PEMs), installed on the Euclid Payload Module (PLM), as well as the Electronic Unit (EU) mounted on the Euclid Service and Module (SVM). The EUCLID FGS EGSEs (Fig. 2) should be able to stimulate electrically the EUCLID FGS PEMs and EU by simulating the behavior of the detection chain (detector + PEM read out electronics) and the behavior of the complete detection chain (PEM output: after data processing) to carry out closed loop tests and avionic open loop test in real time with hardware in the loop.

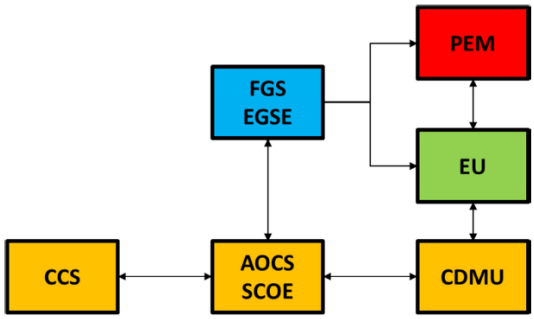

Fig. 2. FGS EGSE overview

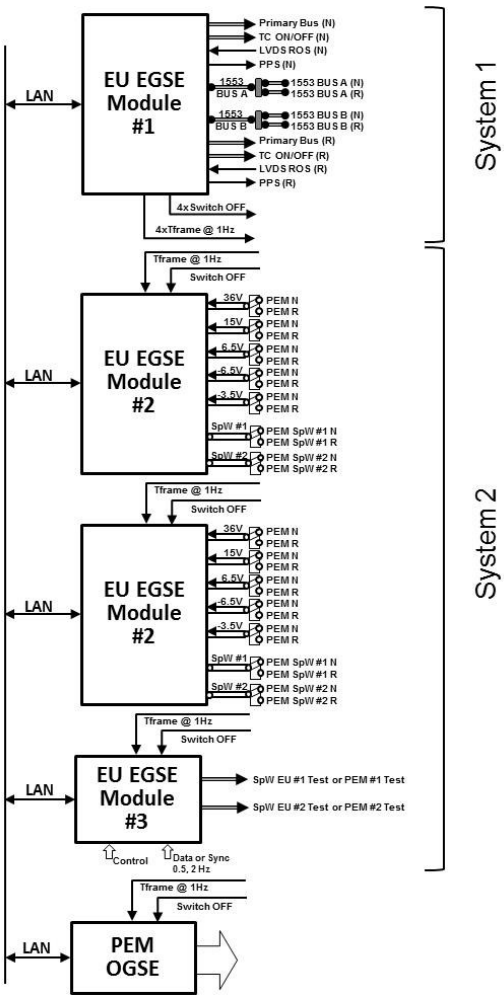

Fig. 3. EU + FGS EGSE requirements

In details, the EU+FGS EGSE should be provided for the verification of the EU, PEM and integrated FGS. EU+FGS EGSE should be provided for EU Assembly, Integration and Testing (AIT) activities and FGS AIT activities. This EGSE shall be used to test the EU unit and to test the FGS subsystem (EU + PEM +Detectors). This EGSE shall be able to test the EU, the EU + PEM's and the EU+ PEM's + Detectors.

The FGS communicates internally through SpaceWire and externally through MIL-STD-1553 with the CDMU. Test connectors are provided for the injection of emulated sky images through SpaceWire.

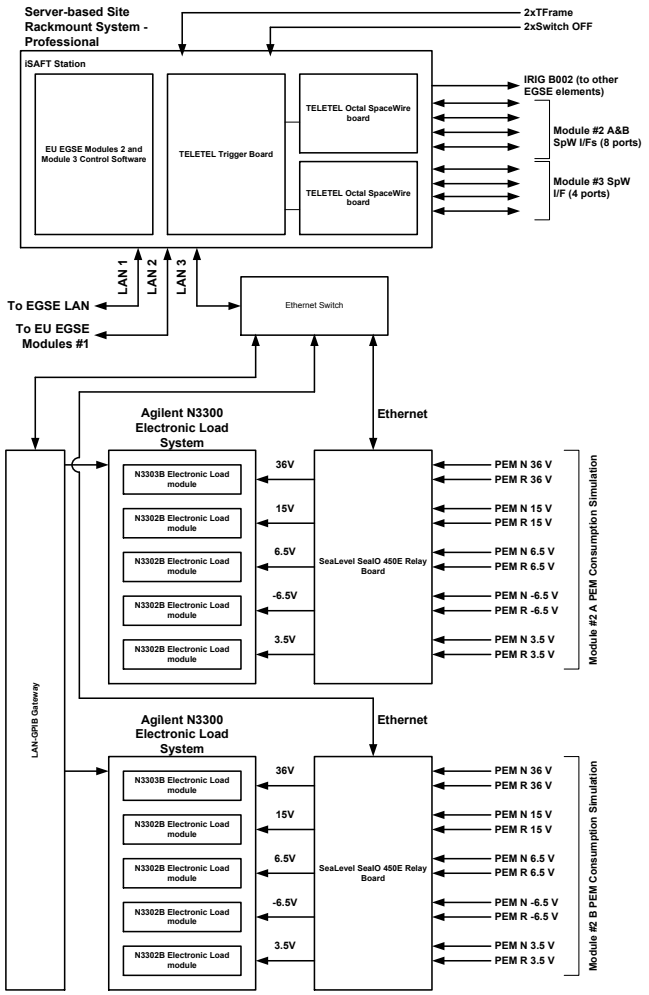

Fig. 4. iSAFT based EU EGSE module #2 & #3 hardware architecture

Finally, the unit shall be controlled through ECSS-14C discrete interfaces for ON/OFF commanding and synchronization with the CDMU, whereas power shall be provided through a Latching Current Limiter. The design approach for the main components is presented in the following section.

B. EU+FGS EGSE

An overview of the structure for EU+FGS EGSE is shown in Fig. 3. It includes three EU EGSE modules (#1, #2 and #3) and the PEM OGSE (Optical Ground Support Equipment). The main functionalities of the EU EGSE module #1 module include EU/FGS FULL Power Supply, Discrete & Synchronization Signals Generation, EU 1553 Protocol simulation and Modules Control.

For each of the EU EGSE Module #2 (A or B) two main functionalities are defined including PEM Power Consumption Simulation and PEM SpaceWire Simulation.

The EU EGSE Module #3 is used for the AIT testing activities and its main functionality is to stimulate simultaneously 2 EU channels or 2 PEM channels.

The components modules that compose the EU EGSE Module #2 & #3 are shown in Table I and their overall HW and SW architectures are presented in Fig. 4 and Fig. 5 respectively.

As derived from the HW architecture of the EU EGSE module #2 (Fig. 4), 4 SpW channels are needed per module #2. These can be supported for both A and B modules through a single TELETEL Octal SpaceWire board that provides SpW Simulation over 8 ports and IRIG time-stamping.

The board supports per port independent transmission trigger conditions and actions and can be configured to transmit upon the assertion of the Tframe signal and to disable the SpW ports upon the assertion of the Switch OFF signal. In addition the level of the Switch OFF can be readable by the SW in order to allow disabling all other elements of Modules #2 A and B.

TELETEL's 16 trigger channels board can be used only to perform electrical adaptation of the Tframe signal and feed it to TELETEL's SpW board at appropriate levels.

1) PEM SpaceWire simulation

The iSAFT control software can simulate PEM SpaceWire links being able to reply to all the TM/TC commands sent by the EU. Additionally, it can send preconfigured file data, which correspond to the data generated by the PEM unit when the detectors are connected.

PEM SpaceWire simulation can be supported by using the SpW Simulation engine, as shown in Fig. 6, and a dedicated module (i.e. PEM Simulation module) that implements the SpaceWire protocol used at the EU-PEM communication and can be controlled by the MMI (i.e. PEM Simulation control window).

The PEM Simulation control can support the configuration of the PEM simulation options, the selection of preconfigured data to be transmitted as well as selection and transmission of specific TM commands during the simulation. The PEM SpaceWire simulation configuration and control can also be performed through user defined Test Cases.

TABLE I. EU EGSE MODULE #2 & #3 COMPONENTS

Function	Subsystem
Processing unit	N/A
Modules #2 PEM Power Consumption Simulation	Electronic Load system
	Electronic load modules for the 36V network
	Electronic load modules for the 15V, 6.5V, -6.5V and 3.5V networks
	Nominal Redundant EU Power Supply switching unit
	LAN-GPIB gateway
	Ethernet Switch for control of Electronic loads and relay switches
Module #2 (A and B) SpW Simulation	Software
	SpW interface
	Tframe triggers
Module #3 SpW Simulation	SpW interface
Interface to other EGSE elements	Switch OFF signals receiver

During PEM simulation the SpW Monitoring engine can simultaneously monitor all SpW links and log and archive all packets with a resolution down to 8 nsecs with an external IRIG source.

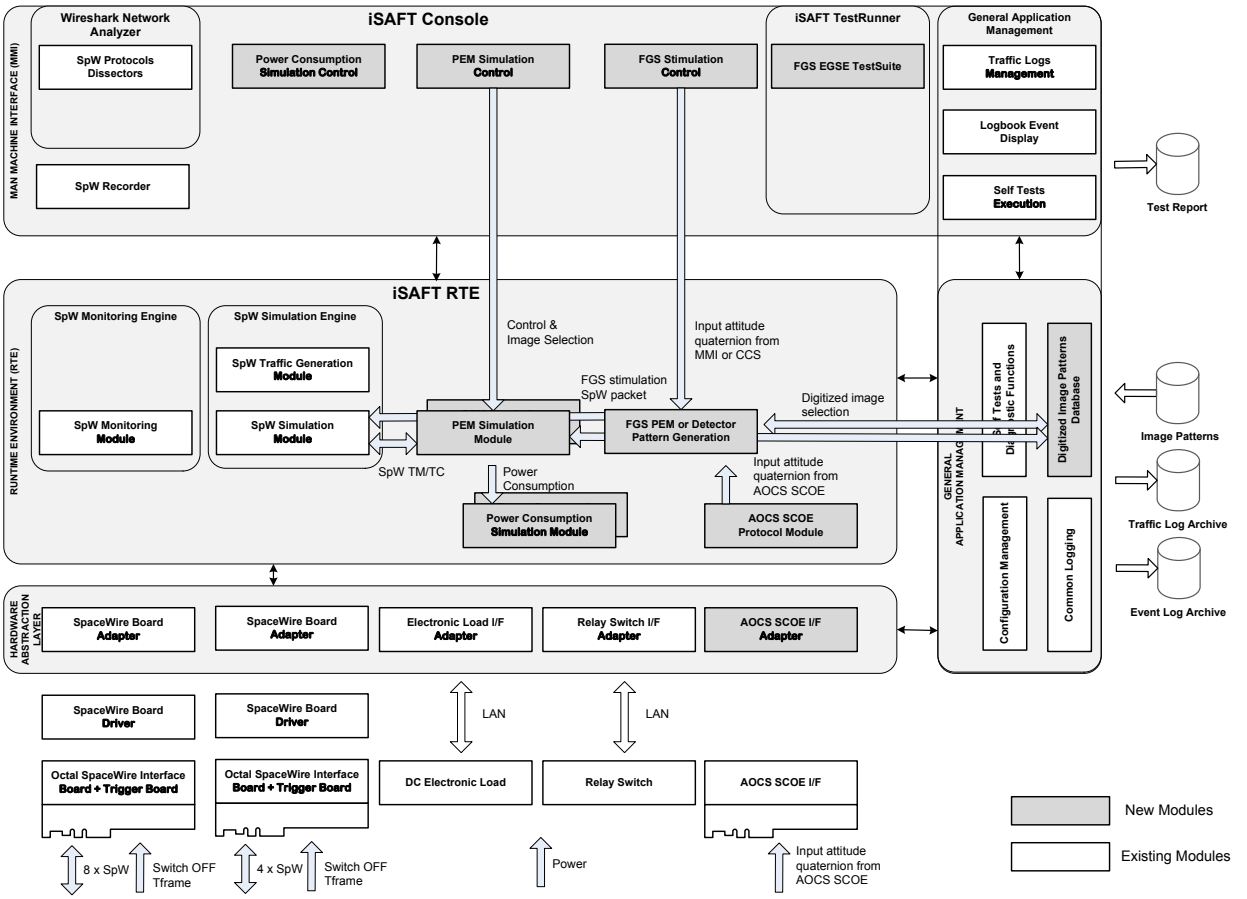

Fig. 5. iSAFT based EU EGSE module #2 & #3 software architecture

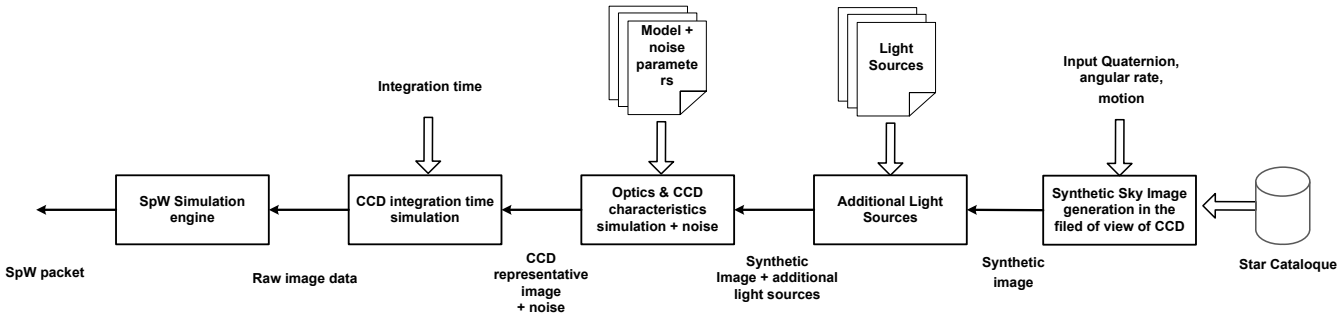

Fig. 6. PEM SpaceWire Simulation

2) FGS Stimulation

The iSAFT Control software can support the stimulation of the FGS PEM or EU Test interfaces through a dedicated module (i.e. the FGS PEM or Detector Pattern Generation module) and the use of the SpW Simulation engine.

For the PEM channels the electrical stimulation signal can be defined as the actual digitized image which would have been delivered by the detector acquisition / readout stage, in consistency with the FGS operational mode, and the satellite dynamics state.

For the EU Channels: the electrical stimulation signal can be defined as the actual digitized image which would have been delivered by the PEM pre-processing in consistency with the FGS operational mode, and the satellite dynamics state. This option will be used with the EU configuration.

The Detector Pattern Generation module implements an algorithm that processes the input (quaternion, angular rate,

etc.) and based on the input selects, generates and processes an image pattern from a data source or data base (i.e. the Digitized Image Pattern Database). The resulting processed data can be packetized and transmitted to the SpW PEM or EU Test links. The data source will contain digitized image patterns, the star catalogue, additional light sources and characteristics of the CCD optics already available.

The FGS stimulation can provide the electrical stimulation signal (SpW packets transmission) at [0.5, 2] Hz according to the inputs delivered by the AOCS SCOE that include:

- Attitude quaternion from the Inertial Reference Frame to the Boresight Reference Frame,
- Satellite's angular rate up to 0.001 deg/s and
- Linear motion.

More specifically, for the PEM channels stimulation the following algorithm can be used:

1. Generate the image of the sky with those stars that are located in the field of view that is observed taken into account the AOCS SCOE input data (attitude quaternion, angular rate, linear motion) and an initial light offset.
    a. This dynamic pattern (two new ones [4000x4000]16bits pixels, each one to be sent for each independent SpW link) can be generated taken into account the AOCS SCOE input data described above (attitude quaternion, angular rate, linear motion) and an initial light offset.
    Note: A [4000x4000] 16bit pixels shall be generated, but two different transmissions to PEM are foreseen:
    1. In case of being needed to transmit the full frame image, this will require a transmission time of 4 secs in a 100Mbps SpW link, in this case the performance requirement for 1 sec delay between consecutive patterns cannot apply.
    2. In windows mode, before sending to PEM any information, a windowing readout shall be simulated on the [4000x4000]16bits pattern. The maximum information that will be sent to PEM is 15windows of 65x65pixels 16bits, which is compatible with 100Mbps and 1s delay.
    b. This dynamic pattern will be generated after correlating the AOCS SCOE input data with the star catalogue info that will be stored in a 1Mbyte memory bank. Only 1Mbyte will be used for storing the star catalogue info that is useful for the real time patterns generation of the current test.
    Note: The produced image will be a synthetic simulated image based on the star objects and their lighting attributes selected from the available star catalogue in the field of view of CCD.
    c. 100Mbyte – 1Gbyte can be used for storing the rest of the star catalogue information that is not being used for the current test.
2. Take into account additional light sources defined by the user, such as, false stars, non-continuous light sources, straylight patterns.
3. Simulate the optics characteristics, such as, focal length, distortions, aberrations, etc. and light noise and faulty pixels.
4. Simulate the integration time of the detector (based on user defined input values).

Tasks 2, 3 and 4 are real time processes that are performed on the selected pattern output of task 1 before being sent to PEM via the SpW link. During FGS stimulation the SpW Monitoring engine should simultaneously monitor all SpW links and log and archive all packets with a resolution of 8 nsec.

Additional open loop simulation can be available in which the user can program one sequence of input parameters to be used instead of the data received from the AOCS SCOE external interface.

This FGS PEM or Detector Pattern Generation module can manage dedicated interfaces (from iSAFT MMI, the CCS LAN, user Test Cases or the AOCS SCOE interface) in order at least to:

- Send Start/Stop Simulation commands,
- Send the required data during closed loop execution, i.e. quaternion, S/C rate, S/C attitude, …
- Send a wrong attitude pattern for special tests.

Finally, the FGS stimulation will be able to stimulate the PEM with a "wrong" attitude patterns commanded by CCS LAN.

III. Conclusions

The validation of flight equipment can be a very demanding process, especially for critical mission equipment as in the case of EUCLID FGS. In this paper, the possible use of the iSAFT integrated HW/SW environment for the validation of flight equipment and more specifically in the case of the EUCLID FGS has been presented. The technical approach presented includes the hardware and software architectures based on the existing iSAFT Protocol Validation System (PVS) for the verification of the EU, PEM and integrated FGS, according to the EUCLID design/technical specification and general requirements for EGSEs.

The analysis showed that iSAFT can support the different requirements of FGS like flight devices by providing a modular and expandable architecture with several features that can be applied for the accurate simulation, validation & monitoring of such a device's behavior. Due to the variable configurations supported by iSAFT, it is possible to cover different performance/reliability/cost requirements for demanding scientific missions like the EUCLID Fine Guidance Sensor case.

REFERENCES

[1] Euclid Mission, http://sci.esa.int/euclid/

[2] TELETEL iSAFT PVS platform, http://teletel.eu/isaft-protocol-validation-platform/

[3] TELETEL iSAFT SpaceWire / MIL-STD-1553 / CAN Recorder, http://teletel.eu/isaft-SpaceWire-mil-std-1553-can-recorder/

[4] EU+FGS EGSE Requirements DRAFT (Ref. EUCL-TAS-RS-2-005)

[5] EGSE PEM+DETECTORS Requirements DRAFT (Ref. EICL-TASE-RS-2-006)

[6] General Design and Interfaces Requirements Specification (GDIR) (Ref. EUCL-TAST-RS-1-003).

[7] EUCLID Fine Guidance Sensor Requirements Specification (Ref. EUCL-TAST-RS-2-016).

Papers Indexed by Author

Author Surname A - K

Author Surname L - R

Author Surname S - Z

Papers Indexed by Session

Tuesday 23 September

Networks & Protocols 1 (Long Papers)

Components (Short Papers)

Missions & Applications (Short Papers)

Standardisation (Long Papers)

Test & Verification (Long Paper)

Wednesday 24 September

Networks & Protocols (Short Papers)

Components (Long Papers)

SpaceFibre (Long Papers)

Poster Presentations

Thursday 25 September

Networks & Protocols 2 (Long Papers)

SpaceFibre (Short Papers)

Onboard Equipment & Software (Short Papers)

Test & Verification (Short Papers)

Standardisation (Short Papers)

Missions & Applications (Long Papers)

Exhibitors

4LINKS LTD

4Links, designs, manufactures and supplies an extensive range of SpaceWire test and simulation equipment and IP products. The company was founded in 2000 by personnel who contributed to the European Space Agency SpaceWire standard, a spacecraft on-board network technology now used internationally on more than 100 satellites. Today the 4Links product range is renowned as being the most comprehensive and reliable on the market. 4Links is based on the Science and Innovation Centre on Bletchley Park, the World War II code-breaking centre, in Buckinghamshire, UK.

AEROFLEX

Aeroflex Microelectronic Solution - HiRel divisions supply integrated circuits such as standard products for HiRel applications including FPGAs, LEON 3FT Microprocessors, Logic, MIL-STD-1553 Databus/Transceivers, Clocks, Voltage Regulators and Supervisors, MUXes, Diodes, MOSFETS, LVDS and Memory families and our SpaceWire products - Transceivers, Protocol IP, Routers.

Our RadHard-by-Design Digital and Mixed-Signal ASICs handle design complexities up to 3,000,000 usable gates. We also offer Radiation Testing and Circuit Card Assembly Services.

Aeroflex Gaisler, based in Goteborg, Sweden, is a provider of SoC solutions and IP-cores for exceptionally competitive markets such as Aerospace, Military and Commercial applications. The Aeroflex Gaisler's IPcores consist of user-customizable 32-bit SPARC V8 processor and floating-point-unit cores, SpaceWire cores, peripheral IP-cores and associated software and development tools. The new GR712 LEON Microprocessor is in production. Aeroflex Gaisler solutions help companies develop application-specific SoCs that are highly competitive for customer specific applications. Gaisler Research's personnel have extended design experience, and have been involved in establishing standards for ASIC and FPGA development.

ATMEL

In Europe, ATMEL has 2 main Business Units:

- MCU: MicroController Business Unit:
  This BU develops Standard products and Custom products based on AVR8, AVR32 and ARM core. ATMEL is becoming the first supplier of 8bit controllers thanks to its success with many applications and especially the MaxTouch family.

- Automotive, Memory and Aerospace Business Unit:
  This BU develops products for dedicated Markets and applications.

Aerospace developments within ATMEL are all located in Europe, mainly in France (Nantes and Rousset) but also with technical centres supporting ASIC and FPGA business locally (France, Italy, Germany, UK).

There is no involvement of any USA Atmel employees and Aerospace products are guaranteed not to be restricted by ITAR and EAR rules.

ATMEL Nantes site has been developing Integrated Circuit for space application since 1985. The development team installed now in Nantes and Rousset has a very large experience of radiation hardened circuits design and fabrication constraints.

ATMEL circuits are available in rad-hard versions that meet the harsh environment (cumulated dose, latch-up and transient phenomena) of space applications. Design and manufacturing facilities reach international quality standards recognition and are QML-V certified and ESCC QML certified.

High-reliability radiation-hardened products provided by Atmel mean:

- Full military operating temperature range (-55 to + 125°C)
- 100K - 300Krd range, Latch-Up, SEE, SEFI hardened

ATMEL also proposes some Rad Tolerant products for Space applications like launchers, manned space flight and LEO satellites. Those devices are also targeting civil and military avionic critical applications where single events need to be minimized.

Rad Tolerant products mean Latch up immunity, 20 to 50Krd range TID and higher SEU LET versus COTS devices.

Qualification flow is also adapted to the targeted market.

Atmel portfolio contents advanced technical and competitive solutions for space market for the following products range:

- Processors & microcontrollers (32-bit SPARC, ARM & 8 bits AVR)
- Memories (SRAM & EEPROM)
- Communication ICs
- SRAM-based Reprogrammable FPGAs
- ASICs (up to 30M gates)

Atmel is committed for the long term to support the aerospace industry.

AXON' CABLE

The Axon' group designs and manufactures wire, cable, connectors and cable assemblies for advanced technology applications in the principal fields of space, aeronautics, medical electronics, automotive and scientific research. Headquartered in France (100 Km east of Paris) the Group employs some 1700 staff in 14 subsidiaries across Europe, America and Asia, with an annual turnover of €115 million euro.

Axon' Cable has been involved in many space projects, including the International Space Station, various LEO and GEO satellites and rocket launchers including Ariane 5, and can boast flight heritage dating back to 1997.

The group offers various types of products for space applications:
- ESCC approved wires, cables and connectors,
- lightweight aluminium round cables and braids,
- aluminium bus bars for satellite power distribution,
- MIL-STD-1553 databus looms for digital transmission systems,
- high data rate links for Voice-Data-Image transmission including SpaceWire, IEEE1394, Ethernet and Fibre Channel,
- solutions suitable for the forthcoming multi-gigabit protocol, SpaceFibre,
- and custom-designed products for specific applications.

Additionally, Axon' has been involved either as prime or subcontractor on a number of ESA EMITS tenders including the development of high temperature thruster cables, the development of low mass SpaceWire, the evaluation of shielding techniques for Spacecraft harnesses, the evaluation of Nano-D for Space, the development of Combo Micro-D's and the provision of cables for the SpaceFibre Demonstrator.

MOSCOW INSTITUTE OF PHYSICS AND TECHNOLOGY

Moscow Institute of Physics and Technology (MIPT) is one of the leading Russian universities in the areas of physics, mathematics, and informatics.

MIPT was founded in 1946 by the Leading Soviet scientists on Special Decision of the Soviet Government as an advanced educational and research Institution for the preparing of the specialists in advanced fields of Science (with primary concentration in Physics) and Industry.

For the time of MIPT functioning 8 Nobel Laureats were its Professors. MIPT graduates Andrey Geim and Konstantin Novoselov were awarded jointly by the Nobel Prize in Physics 2010.

In the 60 years of its history, MIPT has trained over 30,000 high-level specialists in various fields of Science, Technology, Economics, and Business. Over 17,000 MIPT graduates have become Ph. Doctors of Science; over 6,000 have gained degree of Habilitated Doctor of Science. More than 150 alumni have been elected Full and Corresponding Members of the Russian Academy of Sciences.

Our mission is to provide training of highly employable graduates for cutting-edge science and technology fields.

From the outset, MIPT has used a unique system for training specialists, known as the *Phystech System*, which combines fundamental science, engineering disciplines and student research. Students and graduates of MIPT are representatives of an elite circle who, thanks to their interdisciplinary scientific surroundings, are able to fully realize their potential.

Prospectus with detailed information: http://mipt.ru/education/abitur/MIPT_overview_en.pdf

SHIMAFUJI ELECTRIC

Since 1990, Shimafuji Electric has been developing microcomputer boards including transmission, graphics and other complex peripheral functions and also producing small amount of products for some OEMs.

Shimafuji have joined the Japan SpaceWire user Group since early days. We developed the SpaceWire compliant cubic computer, Space Cube with JAXA, and we have some SpaceWire function boards, like Sampling ADC, Digital I/O, and ETC since 2005. Then, our one of latest model is the 4 port Space Wire to Gigabit Ether Unit and we are developing the 24-link SpaceWire Packet Recorder and 48-port SpaceWire Packet Generator based on the 12-slots microTCA SpaceWire Backplane system.

STAR-DUNDEE LTD

STAR-Dundee specialises in supporting users and developers of SpaceWire and SpaceFibre; data networking standards for on-board satellites and spacecraft.

SpaceWire is established as one of the main data-handling networks used on many ESA, NASA and JAXA spacecraft and by research organisations and space industry across the world. SpaceWire's speed, simplicity, flexibility and interoperability have contributed to its continuing adoption and popularity.

STAR-Dundee has a comprehensive product line of SpaceWire test and development equipment that can test across all levels of SpaceWire standard. The product portfolio encompasses equipment to enable the design, development, integration and testing of SpaceWire networks and devices, along with industry-leading flight IP cores, chip designs, design services, consultancy and training.

SpaceFibre is an emerging ESA standard networking technology that provides a very high-speed serial data-link for high data-rate payloads. SpaceFibre aims to complement the capabilities of the widely used SpaceWire standard: achieving initial data rates of 2 Gbits/s improving to 5 Gbits/s long-term, capable of operating over fibre-optic and copper cable, reducing cable mass by a factor of four, adding integrated QoS including bandwidth reservation, priority and scheduling, enhancing robustness with FDIR features at all protocol levels, providing galvanic isolation, and multi-laning improves the data-rate further to well over 20 Gbits/s.

SpaceFibre is being developed by the University of Dundee for ESA and STAR-Dundee can now provide SpaceFibre IP Cores and chip designs, SpaceFibre interfaces, SpaceWire to SpaceFibre Bridge, and SpaceFibre link analysis tools; everything needed for the early adoption of this new technology.

The STAR-Dundee team has leading expertise in all areas of SpaceWire and SpaceFibre technology and is committed to helping our customers adopt these technologies, providing continued support through the full development life-cycle.

TELETEL

TELETEL, founded in 1995, is a private Greek software and hardware, design and development company, having a long history of providing development services and products in the space, defense and aeronautics sectors. TELETEL works very closely with the European industry having provided software and hardware solutions to DASSAULT, SAGEM, THALES, MBDA, EADS, AIRBUS, ALCATEL-LUCENT, MOTOROLA and many other customers. Since Greece's membership to ESA, TELETEL invests in space technologies at an accelerating pace, being today one of the most successful Greek organizations in the space market.

TELETEL's main competence is the provision of system, SW & HW solutions mainly for communication systems with special emphasis on **test, validation and data interfaces simulation (Spacewire, MIL-STD-1553, CAN)**. Since 2007, various activities in the validation of Space related components (SpW, SpW-T, SpW-D, IMA TSP, SCOC3, N-Mass, etc.) have been successfully handled, internal infrastructure (i.e. representative testbeds for on-board network architecture) has gradually been built, and TELETEL developed its own platform/product (i.e. iSAFT PVS) to address the needs of various mission EGSE, SCOE or DFE configurations.

The iSAFT PVS product line includes today the following space products:

- iSAFT Protocol Validation Platform for on-boards data networks (http://teletel.eu/isaft-protocol-validation-platform)
- iSAFT SpaceWire/MIL-STD-1553/CAN Recorder (http://teletel.eu/isaft-spacewire-mil-std-1553-can-recorder)
- iSAFT SpaceWire/MIL-STD-1553 Simulator (http://teletel.eu/isaft-spacewire-mil-std-1553-simulator/)

TELETEL is fully certified according to the ISO 9001:2008 Quality Standard and can handle graded material according to NATO C-M (55) 15 FINAL Security System. The company strictly follows development practices and standards such as ECSS, DO-178B, etc. TELETEL is also involved in various R&D programs funded by EU, ESA, EDA, NATO and industrial consortia. Further information about TELETEL can be found at www.teletel.eu

esa NASA ROSCOSMOS JAXA teletel UNIVERSITY OF DUNDEE