SoCWire: A Network-on-Chip approach for reconfigurable System-on-Chip Designs in Space Applications

B. Osterloh, H. Michalik, B. Fiethe, K. Kotarowski
IDA TU Braunschweig, Hans-Sommer-Str.66, D-38106 Braunschweig, Germany
B.Osterloh@tu-bs.de

Abstract

Configurable System-on-Chip (SoC) solutions based on state-of-the art FPGA have successfully demonstrated flexibility and reliability for scientific space applications like the Venus Express mission. Future high end payload applications (e.g. Solar Orbiter) demand high-performance on-board processing because of the discrepancy between extreme high data volume and low downlink channel capacity. Furthermore, in-flight reconfiguration ability enhances the system with re-programmable hardware and thus a maintenance potential. To achieve these advanced design goals a Reconfigurable System-on-Chip (RSoC) architecture is proposed supported by a flexible communication architecture. The flexibility for on-chip communication is covered by a dedicated Network-on-Chip (NoC) approach. The configurable System-on-Chip solution is introduced and the advantages are outlined. Additionally we present our newly developed NoC approach, System-on-Chip Wire (SoCWire) and outline its performance and the applicability for in-flight reconfigurable systems.

1. Introduction

Configurable System-on-Chip solutions based on FPGAs have already been successfully demonstrated in Data Processing Units (DPUs) for scientific space applications like the Venus Express mission. This approach provides the capability of both flexibility and reliability for system design. For future space missions e.g. Solar Orbiter, the demand for high performance on-board processing has drastically increased. Classical ground processing steps like scientific parameter extraction and subsequent data evaluation need to be performed on-board because of the discrepancy between extreme high data volume and low downlink channel capacity. Additional a further enhancement for DPUs based on FPGAs would be to use their capabilities for in-flight reconfigurability, which allows both system maintenance and performance enhancement on-board. To achieve such an enhanced Reconfigurable System-on-Chip (RSoC) approach a flexible communication architecture is needed, which provides high data transfer rates and is suitable for reconfigurable modules as well.

In this paper we will focus on our newly developed Network-on-Chip (NoC) architecture approach *System-on-Chip Wire* (SoCWire). First we will introduce the configurable System-on-chip approach for the Venus Express Monitoring Camera (VMC), successfully demonstrated in space. We will then outline the essential for a NoC and introduce our SoCWire based architecture and outline its performance, which has been measured in a demonstration implementation

2. Configurable SoC approach

The availability of radiation tolerant high density FPGA enables the integration of special functions (e.g. data compression or formatting/coding) and processor system completely in a single or few high density FPGAs. This scalable on-chip architecture for data handling systems use common modules, standardized interfaces and modeling tools to customize the processing chain and adapt to new mission objectives. Such highly integrated systems have low resource requirements for both mass and power. The major advantages of such a system are flexibility, (re)programmability and module re-use, which can be easily adapted to specific requirements. With the Xilinx XQR Virtex-I and Virtex-II series, FPGAs of large gate counts with required qualification level and radiation tolerance are available for space applications. These FPGAs are SRAM-based and therefore their internal memory cells (Logic, BRAM) are sensitive to Single Event Upsets (SEU), but they enable the reconfigurability of the complete or partial system during development or even in-flight. Dedicated SEU mitigation techniques (configuration memory scrubbing, Triple Modular Redundancy, TMR) reduce the SEU rates to a negligible or at least tolerable value. Primary design goal e.g. for instrument DPUs is to achieve the optimum availability for given system constraints (cost, performance, mass, etc.). The Venus Express Monitoring Camera (VMC) is one example of such a flexible SoC approach. It is based on a “LEON-2” processor, which is provided as highly configurable VHDL model, achieving a computer power of 20 Million Instructions Per Second (MIPS) in our

978-0-7695-3166-3/08 $25.00 © 2008 IEEE
DOI 10.1109/AHS.2008.43

system. Additionally the FPGA includes all peripheral logic and interfaces to different sensors and communication units (e.g. SDRAM Controller including zero wait state error correction, spacecraft RTU interface logic,1355 SpaceWire interface controller)[1].

The calculated overall non-correctable SEU rate of the design is 3.9 error per year for Galactic Cosmic Ray (GCR) + 0.25 % Peak 5 min, which is tolerable for a scientific instrument with an operation time only some hours during a 24h orbit. VMC science operation has been started in the mid of May 2006. Since then, the VMC DPU was switched on for an accumulated time of 4600h, taking more than 81,000 images and is running well. So far only three SEUs in the Xilinx FPGAs have been observed, which is in the expected range.

3. Requirements for future Space applications

VMC demonstrated a successful and space suitable configurable SoC approach. For future space missions classical ground processing steps like scientific parameter extraction and subsequent data evaluation need to be performed on-board. The VMC camera for example consists of a detector of 1Mpixel with 14bit resolution capable to accumulate one image every second which leads to a maximum read out data rate of 1M·14bit/1sec ≈ 14 Mbps. With this readout data rate image processing (e.g. compression) can be carried out by software, if large enough intermediate mass memory is provided (1 Gbit for VMC). Future instruments like the Polarimetic and Helioseismic Imager (PHI) instrument implementation of the Visible-light Imager and Magnetograph (VIM) [2] on Solar Orbiter comprises a 4Mpixel detector with a maximum readout data rate of 4M·12bits/100msec ≈ 500Mbps. The average data rate to the S/C is only 20...60kbps. Therefore, the high speed image readout and summation with extensive processing (e.g. flat-field calibration, telescope PSF reconstruction, vector demodulation, and data compression) have to be done on-board by specific processing cores in hardware.

Additionally, a significant advantage would be an in-flight reconfigurability for maintenance purposes and the capability to update processing modules to improve functionality, e.g. a special data compression core can be replaced by a sophisticated core to calculate scientific parameters directly on-board. Both cores are loaded on demand during different parts of the mission.

4. In-flight reconfigurability architecture

In-flight reconfigurability allows both system maintenance and performance enhancement on-board. The cost is increased development effort e.g. for safe configuration data transfer, safe reprogrammable configuration memory and possibility for SEU mitigation techniques like scrubbing. Additionally, for an in-flight update of a processing module in the system the once on ground achieved qualification and reliability of the total system has to be maintained.

Our framework for in-flight reconfigurability is based on the VMC approach with the additional feature of partial reconfigurability. The FPGA is subdivided into static and partial reconfigurable areas which can be updated during flight. The static area remains unchanged during the whole mission and stores all mission critical interfaces (e.g. processor, interfaces to spacecraft). The partial reconfigurable area can be updated during flight. This offers the advantage that only the updated module has to be qualified in a delta-qualification step, not the whole system [3].

The architecture model we use for our instrument DPU designs is usually a macro-pipeline system with pre- and post-processing steps in a dedicated structure as depicted in Figure 1. This architecture covers the typical processing chain requirements in classical instruments up to complete payload data handling systems.

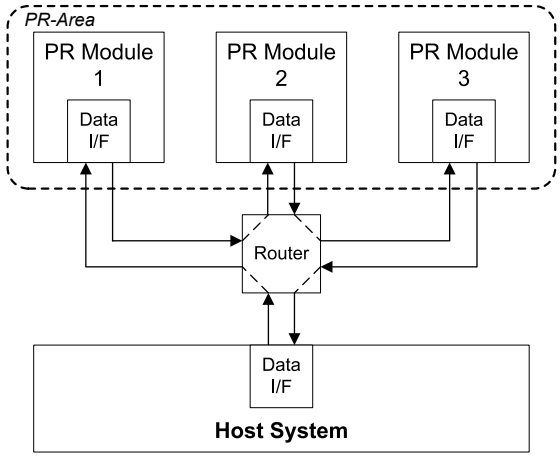

Figure 1. DPU architecture macro-pipeline system

5. Partial reconfiguration in Virtex-4

Xilinx announced the space suitable radiation hardened Virtex-4 QPro-V family. In contrast to earlier Virtex families e.g. Virtex-II the internal configuration of the hardware architecture has changed. In previously FPGAs the CLBs (Configurable Logic Block) were surrounded by a ring of IOBs (Input-Output Buffer). The IOBs are now organized in columns. Additionally the FPGA is divided in clock regions, each comprising 16 CLBs. These clock regions have significant influence on the configuration process of the FPGA. Xilinx FPGAs are customized by loading configuration data into the internal

configuration memory. The configuration memory is arranged in frames that are tiled about the device. These frames are the smallest addressable segment of the configuration memory space. One frame comprises 16 CLBs and therefore one clock region [4]. This architecture with clock regions and IOB structures has the advantage to overcome the limitation of partial reconfiguration in the Virtex-II architecture. Partial Reconfigurable Modules (PRMs) do not have to occupy the full height of the device and IOBs above the top edge and below the bottom edge of the module are not part of the module resources. Therefore the logic resources left, right, top and bottom of a PRM can be used for the static area. The Virtex-4 family provides now a 32Bit data word width configuration interface (SelectMap) running at 100Mhz which significantly decrease reconfiguration time by a factor of 8 compared to Virtex-II. For communication between modules (static and partial reconfigurable area) Xilinx provides new unidirectional Bus-Macros in the Virtex-4 family which can connect modules horizontal and vertically.

These Bus-Macros are suitable for hand shaking techniques and bus standards like AMBA or Wishbone. As mentioned before, classically the instrument DPU architecture is a macro-pipeline system with high data rate point-to-point communication.

To realize this architecture in a bus structure, multi master and bus arbitration are needed. Also bus structures are limited in the Xilinx hardware architecture. In a partial reconfigurable system a bus requires wires that distribute signals across the device. The Virtex family provides bidirectional vertical, horizontal long lines that span the full high and width of the device and 3-State buffer horizontal long lines that span the full width of the device. These long lines are limited in resources in the device, 24 bidirectional horizontal, vertical and 4 3-State buffer long lines per column CLB [5]. With these limitations a bus structure based system would encounter the following disadvantages:

- An SEU in the PRM bus interface logic could stop the system
- No dedicated Bus-Macros are provided by Xilinx to access long lines which lead to manual time consuming routing.
- Failure tolerant bus structure (high efforts) is necessary to guarantee data integrity
- Dynamic reconfiguration of a PRM could block the bus and stop the system
- Limited long lines resources restrict the bus structure in data word width

Further issues come from the physics of micro technology: long global wires and buses have unpredictable performance, high power consumption and noise phenomenon [6]. With the issues mentioned before we consider instead a networked architecture with a Network-on-Chip (NoC) approach providing:

- Reconfigurable point-to-point communication
- Support of adaptive macro-pipeline
- High speed data rate
- Hot-plug ability to support dynamic reconfigurable modules
- Easy implementation with standard Xilinx Bus-Macros

In order to achieve these requirements we have developed our own NoC architecture: *System-on-Chip Wire* (SoCWire).

6. System-on-Chip Wire (SoCWire)

Our approach for the NoC communication architecture, which we have named SoCWire, is based on the well known ESA SpaceWire interface standard [7]. SpaceWire is a well established standard, providing a layered protocol (physical, signal, character, exchange, packet, network) and proven interface for space applications. It is an asynchronous communication, serial link, bi-directional (full duplex) interface including flow control, error detection in hardware and hot-plug ability for an automatic reconnection after a link disconnection.

6.1. SpaceWire

SpaceWire uses Data Strobe (DS) encoding. DS consists of two signals: Data and Strobe. Data follows the data bit stream whereas Strobe changes state whenever the Data does not change from one bit to the next. The clock can therefore be recovered by a simple XOR function. The performance of the interface depends on skew, jitter (Figure 2) and the implemented technology. Data rates up to 400 Mb/s can be achieved.

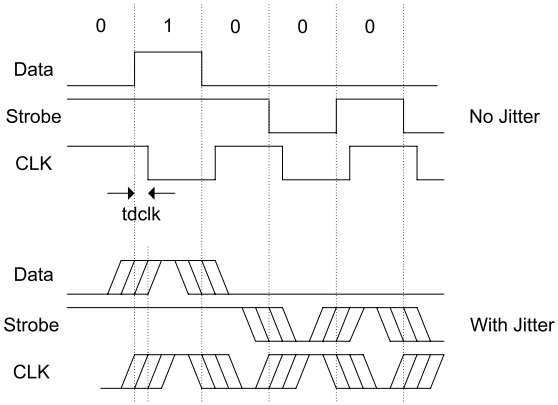

Figure 2. Data Strobe encoding

The SpaceWire character level protocol is based on the IEEE Standard 1355-1995 with additional Time-Code distribution. The character level protocol includes data character, control character and control codes. A data character (10bit length) is formed by 1 parity bit, 1 data-control flag and 8 data bits and includes data to be transmitted, as shown in Figure 3.

The data-control flag indicates, if the current character is a data (0) or control character (1). Control characters (4-bit length) are used for flow control: A flow control token (FCT), end of packet markers (EOP or EEP) and an escape character (ESC) are used to form higher level control codes (8-14bit length) e.g. NULL (ESC+FCT) and Time-Code (ESC + Data character).

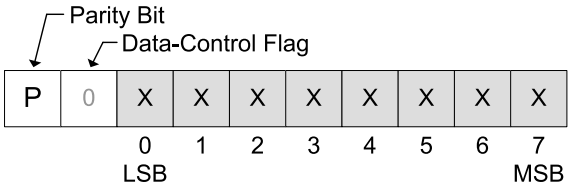

Figure 3. Data character

6.2. SoCWire CODEC

As mentioned before, SpaceWire is a serial link interface and the performance of the interface depends on skew, jitter and the implemented technology. For our NoC approach we are in a complete on-chip environment with up to 6 reconfigurable modules, which can be operated by one switch. The maximum character length in the SpaceWire standard without time code, which is not needed in our NoC, is 10bit (data character). Therefore we have modified the SpaceWire interface to a 10bit parallel data interface, as depicted in Figure 4.

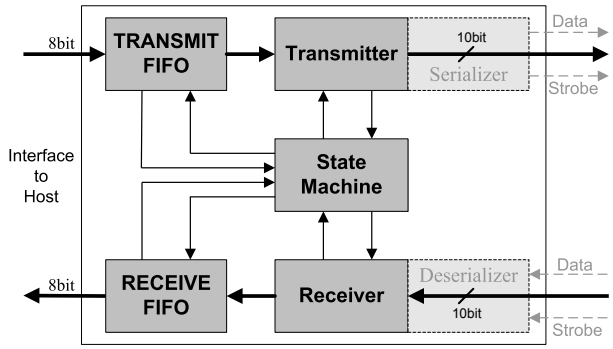

Figure 4. SoCWire CODEC derived from Space Wire, w/o Serializer/Deserializer

The advantage of this parallel data transfer interface is that we can achieve significantly higher data rates as compared to the SpaceWire standard. Additionally, we have implemented a scalable data word width (8-128bit) to support medium to very high data rates.

On the other hand we keep in our implementation the advantageous features of the SpaceWire standard including flow control and hot-plug ability. Also the error detection is still fully supported making it suitable for an SEU sensitive environment. Figure 5 shows a simulation of a full-duplex point-to-point (A and B) communication. It shows the data interface (dat_in, dat_out), control signals (dat_ful, dat_nwrite, dat_empty) and the signals of one link (TX, RX). Core A TX, RX(0) reflects the parity, Core A TX, RX(1) the data-control flag and Core A TX, RX(2-9) data character or control character

The initial data transfer phase shows 11 data characters until the first FCT occurs, which leads to the internal core data pipeline. After the initialization phase the flow control follows the SpaceWire standard where after every eight data characters one FCT is sent to signal that Core A is ready to receive data. The transfer ends with an EOP. The maximum data rate for a bi-directional (full-duplex) transfer can therefore be calculated by:

$$DRate_{Bi}\left[\frac{Mb}{s}\right] = f_{Core(MHz)} \times DWord\ Width - \frac{f_{Core(MHz)} \times DWord\ Width}{8}$$

For a unidirectional data transfer the flow control characters are processed in parallel and the maximum data rate can be calculated by:

$$DRate_{Uni}\left[\frac{Mb}{s}\right] = f_{Core(MHz)} \times DWord\ Width$$

Table 1 shows data rates for different data word width, unidirectional and bi-directional (full-duplex) data transfer.

The SoCWire CODEC has been implemented and tested in a Xilinx Virtex-4 LX60-10. Table 2 shows the occupied area and maximum clock period depend on the data word width

Table 1. SoCWire CODEC data rates for given core clock frequency

DWord Width	f_{Core} *(MHz)*	*DRate [Mb/s]*	
		Unidirect.	*Bi-direct.*
8	100	800	700
32	100	3200	2800

Table 2. SoCWire CODEC synthesis report

DWord Width	*Max.* f_{Core} *(MHz)*	*Area*	
		LUT	*FlipFlops*
8	200	297	140
32	180	447	260

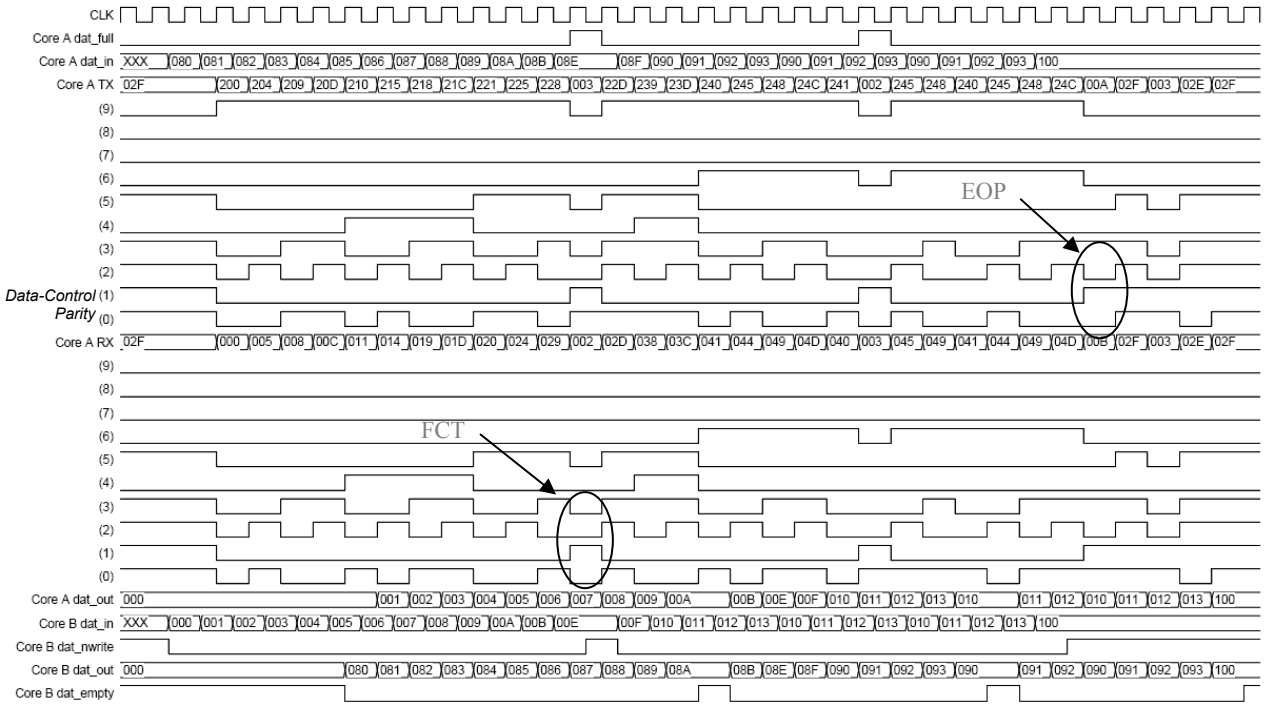

Figure 5. Simulation of a bi-directional (full-duplex) point-to-point communication

Table 1 and Table 2 show that high data rates are achieved at significantly small area cost occupied by the SoCWire CODEC. We ensured that no distributed RAM elements like SRL16 have been implemented in the design to enable configuration scrubbing also of these FPGA parts.

6.3 SoCWire Switch

To build up a network, a switch and a packet oriented protocol is needed. The switch enables the transfer of packets arriving at one link interface to another link interface on the switch, and then sending out from this link. The SoCWire Switch and its protocol are again based on the SpaceWire standard. The packet level comprises: destination address + cargo + end of packet. The destination address includes the destination identifier to support routing of packets. The cargo contains the data characters that need to be transferred from source to destination. The end of packet marker can be either EOP for normal end of packet or alternatively EEP for exceptional end of packet as an indication of an error in the packet. The SoCWire Switch determines from the destination address where the packet is to be routed to. As mentioned before, our NoC consists of up to 6 reconfigurable modules. Therefore the direct port addressing (packets with a port address are routed directly to one of the output ports) has been implemented. As soon as the destination port of a packet is determined and the port is free the packet is routed immediately to that output port. The port is marked as busy and can not be accessed until the end of the packet. This is also known as wormhole routing which reduces buffer space and latency. Our SoCWire Switch is a fully scalable design supporting data word width (8-128bit) and 2 to 32 ports. It is a totally symmetrical input and output interface with direct port addressing including header deletion. The SoCWire Switch has been implemented and tested in a Xilinx Virtex-4 LX60-10. Table 3 shows the occupied area and maximum clock period for a 4 port switch dependent on the data word width

Table 3. SoCWire Switch (4 Ports) synthesis report

DWord Width	*Max. f_{Core} (MHz)*	*Area*	
		LUT	*FlipFlops*
8	190	1736	668
32	170	2540	1169

The SoCWire Switch basically consists of a number of SoCWire CODECs according to the number of ports and additional fully pipelined control machines. The maximum data rate is therefore equivalent to the SoCWire CODEC. The design is also prepared for configuration scrubbing without distributed RAM elements.

7. Tests and Results

We have implemented four SoCWire CODECs, one in the Host system, three in the PRMs and one SoCWire Switch in a dynamic reconfigurable macro-pipeline system, see Figure 1. The Host system and SoCWire Switch where placed in the static area and the PRMs in the partial reconfigurable area. All SoCWire CODECs where configured with an 8 bit data word width. The implementation of the system with reconfigurable areas could be easily implemented with the standard unidirectional Xilinx Bus-Macros. Figure 6 shows a cut out of the placed and routed SoCWire macro-pipeline system: the PRMs (PRM1, PRM2 and PRM3) and Bus-Macros in a Virtex-4 LX 60. The static area is distributed over the FPGA.

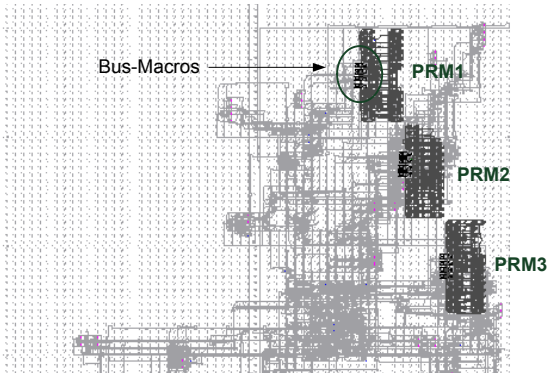

Figure 6. SoCWire Macro-Pipeline System

The PRMs were configured as packet forwarding modules. We have tested different configuration of packet forwarding e.g. between modules, through the whole macro-pipeline system, under the condition of parallel communication between nodes. The system runs at 100Hhz and the maximum data rates of the simulation could be validated to be 800 Mbps. We dynamically reconfigured one PRM in the system. During the reconfiguration process the communication between the PRM and SoCWire Switch was interrupted, the other PRMs connections were still established. After the reconfiguration process was completed the communication between the two nodes was built up automatically without any further external action (e.g. reset of node or switch). This makes the system ideal for even dynamic reconfigurable systems. The Partial Reconfiguration Time (PRT) can be calculated by:

$$PRT[s] = \frac{PRM[Bytes]}{CCLK[Hz] \times SelectMap_{DWord\ Width}[Bytes]}$$

The size of one PRM was 37912 Bytes (64 Bytes command + 37848 Bytes data) and therefore the PRT 758µs (SelectMap, 8Bit data word width at 50Mhz). For this test system the area for one PRM was set to utilize 0.6 % of the logic resources.

8. Conclusion

Configurable System-on-Chip Data Processing Units based on state-of-the art FPGAs are a proven solution for space applications. For future space applications the demands for high performance on-board processing increases enormously. Additionally, in-flight reconfigurability is a further enhancement to support update of hardware functions on-board. To meet these requirements an enhanced architecture with a NoC approach is needed. In this paper we presented our NoC approach SoCWire. SoCWire meets all requirements for a high speed reconfigurable architecture. High data rates are achieved with significantly small implementation efforts. Hardware error detection, hot-plug ability and support of adaptive macro-pipeline are provided. SoCWire can be easily implemented with standard Xilinx Bus-Macros in the Virtex-II (with limitation of the architecture) and Virtex-4 family. The high flexibility of the reference design allows the designer to adapt the SoCWire system quickly to any possible basis architecture. Validation and verification are simplified by a low resource use. Robustness and error mitigation are guaranteed by a clear and straight RTL design. Automatically reconnection after update of a module makes it even suitable for dynamic reconfigurable system.

9. References

[1] B. Fiethe, H. Michalik, C. Dierker, B. Osterloh, G. Zhou, "Reconfigurable System-on-Chip Data Processing Units for Miniaturized Space Imaging Instruments" *Proceedings of the conference on Design, automation and test in Europe (DATE)*, pp. 977-982, ACM, 2007, ISBN 978-3-9810801-2-4

[2] ESA-ESTEC, *Solar Orbiter Payload Definition Document*, ESA-ESTEC, October 2007, SOL-EST-SP-00705

[3] B. Osterloh, H. Michalik, B. Fiethe, F. Bubenhagen, "Enhancements of reconfigurable System-on-Chip Data Processing Units for Space Application", *AHS'07. pp. 258-262, Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, Edinburgh, August 2007

[4] Xilinx, *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Functional Description*, www.xilinx.com, September 2005

[5] Xilinx, *Virtex-4 Configuration Guide*, www.xilinx.com, October 2007

[6] A. Jantsch, H. Tenhunen, *Networks on Chip*, Kluwer Academic Publishers, USA, 2003, ISBN 1-4020-7392-5

[7] ECSS, *Space Engineering: SpaceWire–Links, nodes, routers, and networks*, ESA-ESTEC, Noordwijk Netherlands, January 2003, ECSS-E-50-12A