Design and Implementation of NoC architectures based on the SpaceWire protocol

Sami HACHED
Dept. of Electrical Engineering
University of Carthage, INSAT,
LECAP Laboratory (EPT/INSAT)
TUNISIA
sami.hached@gmail.com

Mohamed GRAJA
Dept. of Electrical Engineering
University of Carthage, INSAT,
LECAP Laboratory (EPT/INSAT)
TUNISIA
epyon_grj@yahoo.fr

Slim BEN SAOUD
Dept. of Electrical Engineering
University of Carthage,INSAT,
LECAP Laboratory (EPT/INSAT)
TUNISIA
slim.bensaoud@gmail.com

***Abstract*- The SpaceWire is a standard for high-speed links and networks used onboard spacecrafts, designed by the ESA, and widely used on many space missions by multiple space agencies. SpaceWire has shown a great flexibility by giving the space missions a wide range of possible configurations and topologies. Nevertheless, each topology presents its own set of tradeoffs such as hardware limitations, speed performance, power consumption, costs of implementation, etc. In order to compensate these drawbacks and increase the efficiency of the SpaceWire networks, many solutions are being considered. One of these solutions is the Network on Chip or NoC configuration which resolves many of these drawbacks by reducing the complexity of designing with a regular architecture improving speed, power, and reliability and guaranteeing a controlled structure. This paper presents the main steps for building Network on Chip based on the SpaceWire protocol. It describes the internal structure, the functioning and the communication mechanism of the Network's Nodes. It also exposes the software development and the validation strategy, and discusses the tests and results conducted on the adopted NoC topologies.**

***Keywords*- NoC architectures; SpaceWire protocol; Codec IP; ,FPGA.**

I. INTRODUCTION

The SpaceWire standard was developed to meet the electromagnetic compatibility (EMC) specifications of typical spacecraft. It is also easy to implement, has a great flexibility, and supports fault-tolerance by providing redundancy to networks [1,2]. In order to interpret and exploit the received data coming from space equipments correctly, the reliability and the protection from errors of these data are essential. To ensure reliability without jeopardizing the spacecraft operations neither its components, the Spacewire was quickly adopted in many space missions, not only in the ESA but also among the largest agencies world's space like NASA, JAXA, or RKA.

The financial constraint of these missions eventually led to many researches to elaborate new strategies of integration to reduce weight and improve configurability within a distributed satellite system (DSS) network. Among the solutions explored, the Network on Chip (NoC) is a recent solution paradigm adopted to increase the performance of multi-core designs [3, 4]. The key idea is to interconnect various computation modules (IP cores) in a network fashion and transport packets simultaneously across them, thereby gaining performance [5,6]. In addition to improving performance by having multiple packets in flight, NoCs also present others advantages including scalability, power efficiency, and component re-use through modular design. The concept of Network-on-Chip or NoC is a new approach to design mechanisms for internal communications of a System On-a-Chip. The NoC-based systems can accommodate multiple asynchronous clock signals that use recent complex SoC [7, 8, 9].

The desired architecture built using NoC allows interconnecting different systems' types via a communications network, all integrated on a chip. This configuration gives the benefits of large scale integration, which leads to a reduction of power consumption and a significant decrease in wiring and noise. On the other hand, it allows the system to take advantage of techniques and theories of development network as the routing of data, parallelism and communication modularity.

In this context many works were conducted such as the SpaceWire based System-on-Chip (SoCWire) communication network. Designed to give enhanced dynamic reconfigurable processing module, the SoCWire provides a mean to reconfigure various parts of the FPGA during flight safely and dynamically. The SoCWire has a dynamic partial reconfigurable architecture with host system including SoCWire CODEC and PRMs (Partial Reconfigurable Modules) with SoCWire CODEC as well as an additional module for control and data generation [10]. Besides, Configurable System-on-Chip (SoC) solutions based on state-of-the art FPGA have successfully demonstrated flexibility and reliability for scientific space applications. Furthermore, in-flight reconfigurability and dynamic reconfigurable modules enhances the system with maintenance potential and at runtime adaptive functionality. To achieve these advanced design goals a flexible Network-on-Chip (NoC) is proposed which supports in-flight reconfigurability and dynamic reconfigurable modules [11, 12, 13, 14].

The INTA (Instituto Nacional de Técnica Aeroespacial) has made several successful missions and focuses on small satellites with NANOSAT and INTAµSAT. In addition, the INTA has recently made a data architecture based on Spacewire protocol for the INTAµSAT. This design includes several subsystems including an On Board Data Handling (OBDH). Due to its importance as it ensures the best performances for data exchange on small satellites, the OBDH subsystem also comprises dedicated Remote Terminal Units (RTU) and a Mass Memory Unit (MMU). The ladder consists of a set of SDRAM memory banks and a Payload Processor as part of a System-on-Chip design. Together with the processor, the SoC will provide a Spacewire router, a low data rate interface with the payload and a CAN interface [15].

Moreover, recent researches discuss other possibilities and solutions to use the SoC technology. In fact, the similarities between the IEEE802.11 and the Spacewire standards were exploited in order to design a wireless link for Spacewire networks. The idea was to create a bridge to connect SpaceWire routers with an IEEE802.11 transceiver. This bridge was also capable of converting data from one standard format to the other and controlling information flow and had a mechanism to provide memory access to SpaceWire routers. The bridge was incorporated into a high performance SoC, which provides a communication platform enabling spacecraft with SpaceWire networks to communicate via inter-satellite links based on the IEEE 802.11 wireless network standard [16].

Additionally, other works present a design of a SpaceWire Router-network using CSP (Communication Sequential Processes) where an IEEE1355 network router is integrated as an Intellectual Property (IP) core [17]. The router design has been evaluated, refined and verified from the point of view of robustness and security using CSP method, one of the formal design methods [18]. The router was implemented in a Network on Chip (NoC) formed with several TPCOREs [19] an IP (Intellectual Property) core for the T425 Transputer where the same machine instructions as the transputer are executable in this IP core.

In this paper, we investigate the protocol SPW which is widely used to provide communications between devices in satellites and we propose to explore different implementation solutions in the form of NoC. After a brief presentation of the NoC solutions advantages of, we introduce the SpaceWire IP Codec we developed and its internal components. Then we discuss the adopted design strategy and present the structure of our SpaceWire node which is the unitary element of the discussed NoC solutions. Finally we expose the main studied NoC architectures, their performance/capability and the obtained testing results.

II. Advantage of NOC Architectures

The NoC solution can combine theories and methods used in networks communication mechanisms on chip. Compared to the communication bus conventional, they greatly improve the scalability of systems on chip, the exchange data and power consumption in complex systems. A network on chip is established from multiple type connections point-to-point interconnected by routers. The messages or frames can circulate throughout the network and reach all machines. The design of a NOC is similar to a telecommunication network-based modems and uses the switching digital bits or packets through links multiplexed [20].

Thanks to this architecture, the connections are shared by several signals that achieve a high level of parallelism since all links may convey different packets of data simultaneously. And more complex integrated networks increases, the size and performance NoC are improved over previous architectures communications [21].

Networks-on-chip (NoCs) are designed to provide high bandwidth and parallel communication, while minimizing the number of employed interconnect resources. However, NoC performance can significantly degrade in absence of an effective flow control mechanism. The flow control is responsible for preventing resource starvation and congestion inside the network. This goal is achieved by managing the flow of the packets that compete for resources, such as links and buffers [22, 23].

Due to the reduction in the construction’s scale and the circuit’s optimization, the NoCs tend to reduce design complexity and routing connections in a SoC, while ensuring low power consumption, minimal noise and high reliability. NoCs support modularity and allow distribution of data processing and communication. They are very suitable for test platforms and increased productivity.

III. The Spacewire Node Design and Validation

A. *The Node Hardware Design*

A SpW node is composed mainly of two modules, namely (1) the host based on the processor on which the user application will be implemented and (2) the SpW CODEC implementing the protocol interface “Fig. 1”. The Host represents a processing unit that will execute tasks and eventually communicate with other nodes. While the SpW interface plays the role of the modem and enables interfacing with the SpW network “Fig. 2”.

The designed SpW CODEC IP in “Fig. 1”, is composed of the following main modules:

- Transmitter module: responsible for emissions. Its role is to synchronize the reception of data arriving from the host, build frames Spacewire (FCTS, NULLS, N-CHARS and Times Codes) and send coded "DATA-STROBE”.
- Receiver module: responsible for receiving data. Unlike the transmitter module, it decodes the incoming frames, classifies them by type, and synchronizes the delivery of N-CHARS and time codes with the host.
- FSM module: is the state machine responsible for connection and communications management. This is the heart of the interface that controls the operation of all other modules.
- Timer module: has the task of ensuring the necessary synchronization for initializing a connection.

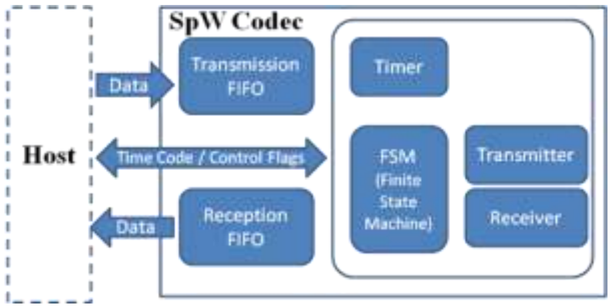

Figure 1. Overview of the SpaceWire CODEC architecture

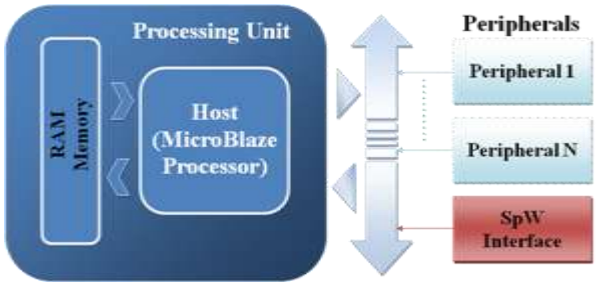

Figure 2. Overview of the SpaceWire node architecture

For the processing unit, a Xilinx MicroBlaze microprocessor (soft-core clocked at 100 MHz) is used and connected to:

- A RAM Memory bloc of 64 Kbits with its controllers.
- An OPB bus controller for OPB peripherals.
- An UART to control, monitor and debug the functioning of the node with a computer.
- An interrupt controller for the management of priority signals.

The HOST communicates with its devices on the OPB Bus. Data exchange is made via formatted requests and responses through the bus and in its native form. The SpaceWire Interface's IP is only usable with low-level binary signals. It is far from being equipped to be directly plugged into the OPB peripherals bus. To overcome this inability, an OPB IPIF (Intellectual Property InterFace) was added; so we can transform our "raw IP" into an OPB interface device that can communicate not only with our processor but can also serve in any other project involving this bus.

The concept of the IPIF is to regroup the different I/O into registers interposed between the OPB bus and the IP. The exploitation of the device (formerly binary) is achievable through loading and unloading register's driver-functions targeted by the address of the device in question. So in order to command or read any I/O of the interface, we have just to load the bits of the register covering that I/O while identifying the interface on which the changes will be brought.

B. The Node Software Development

The programming of host was made in C language with EDK environment. We wrote a function for each elementary operation of the interface: initialization, sending and receiving data, sending and receiving time codes and the display on the output stream. This facilitates the programming, avoids cluttering our source code and facilitates the errors localization when debugging.

With these functions we created a library dedicated to our device. So the host's main program will be less loaded and the update or the summoning of a function in several source codes does not require its "full" presence / reissue in each code. That was very handy in a NOC programming or any multiprocessor configuration.

C. Tests and validation

In order validate the functioning of the designed node; we have established a checklist that includes the main criteria for compliance with the standard. We made an experimental plan that runs through scenarios putting in trial : The proper connection, the proper sending and receiving of data (Chars, control codes, time codes) and the proper functioning of the FiFos (Indicators, limitations and good management).

The tests were conducted on a self-looped Node and on a host equipped with two interconnected interfaces.

IV. The Spacewire NoC designed Architectures

A. The double Nodes Network:

We first began with a double node network-on-chip that consists of two nodes connected to each others. We have doubled the material of a single node, so each Node has its own processor, working memory and peripherals. The only exclusive device to one of the nodes is the serial interface that does not support multi-drive and is used to debug the program and the functioning of the network.

We could use an OPB Bridge to interconnect the processor's OPB buses on and provide simultaneous access to the UART, but it goes against our main objective which is to route two separate nodes connected only by Data-Strobe signals for SpaceWire communication. The node equipped with UART will perform the monitoring of the network and inform us of its status. If need dictates, it is possible to supervise each node individually by connecting it to the UART, but with each change, a new synthesis and a new routing is required.

Concerning the software layer, the two processors have their own individual programming source, headers and libraries (dual-processor configuration) and exploit the SpaceWire library we created.

To test this setup, we used a simple source code, exchanging data. The test's scenario consists on sending X data and time / control codes from the first node to the second and the second node sends fully the received data to the first one as shown in the flowchart diagrams of "Fig. 3" and "Fig. 4". The tests were positive: Whether in data, control codes or time codes exchange, the dual node architecture meets the standards of Spacewire protocol. Using this architecture, a parallel processing could be made with an exchange of data at high speed and very low error rate.

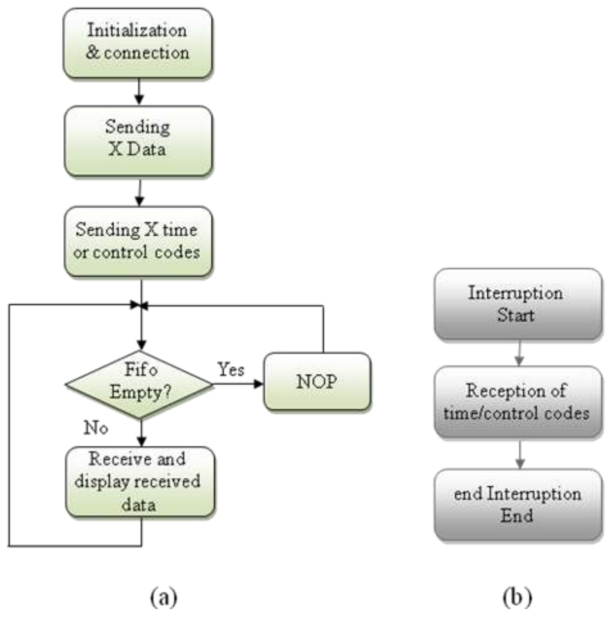
Initialization & connection
Sending X Data
Sending X time or control codes
Fifo Empty?
Yes
NOP
No
Receive and display received data
Interruption Start
Reception of time/control codes
end Interruption End
(a)
(b)

Figure 3. Node A flowcharts: (a) Main Program; (b) Interruption

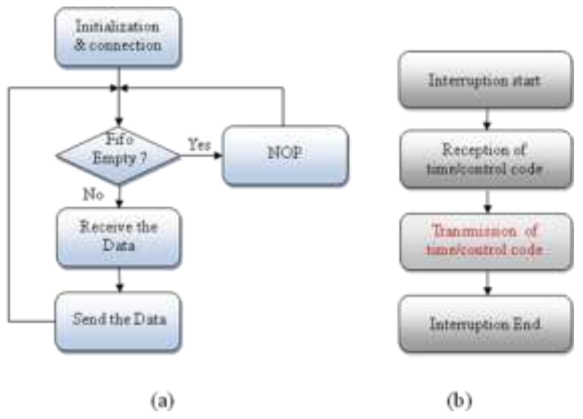
Initialization & connection
Fifo Empty ?
Yes
NOP
No
Receive the Data
Send the Data
Interruption start
Reception of time/control code
Transmission of time/control code
Interruption End
(a)
(b)

Figure 4. Node B flowcharts: (a) Main Program; (b) Interruption

B. The triple Nodes Network

In the fully meshed networks, the nodes are linked together by direct connections (without any intermediary node). When a node needs to communicate with another it refers directly to it (no data carrying or intermediate machine). This direct communication provides the mesh network the highest possible performance in terms of communications: It minimizes the loss of data, errors and data transfer time. Unfortunately fully meshed networks suffer from a major handicap which is the high number of links and communications interfaces necessary for their establishment. Indeed, the number of interface is double the number of connections which is N × (N-1) / 2. That is costly in terms of hardware (special cables, shielding, interfaces ...), size and weight.

At our level of integration, the problems related to the scale are resolved; the only persisting limitation is the capacity of the FPGA. The space probe intended to explore Venus has received a Virtex 4 with a Spacewire network and feedback over this mission are very good! That's why we decided to explore this kind of Networking Method on Chip.

In the same way as the dual node architecture, we tripled the number of components. The only difference compared to the dual node architecture lies in the number of interfaces in each node. We provided every Spacewire node with two interfaces: One for each channel. So when Node A needs to transfer data to Node B or C, it sends them through the interface linking it with the Node in question.

The network architecture is as follows:

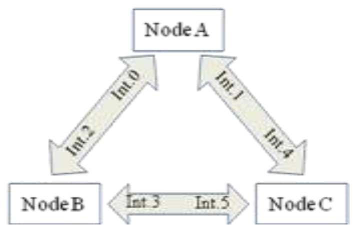

Figure 5. Triple nodes Architecture

As for the dual network node, each of our three nodes must be programmed separately. The testing scenario we executed consists on sending data through the network and make it returns back to its issuers: We send data and time / control codes from the first node to the second, the second node sends them to the 3rd which will forward them back to the first (that supervises the network and displays the data movement on the serial connection).

According to the original node and the destination node, the data are sent threw the interfaces linking to the two concerned nodes.

The programs of each node are illustrated by the flowcharts diagrams shown on "Fig. 6" and "Fig. 7":

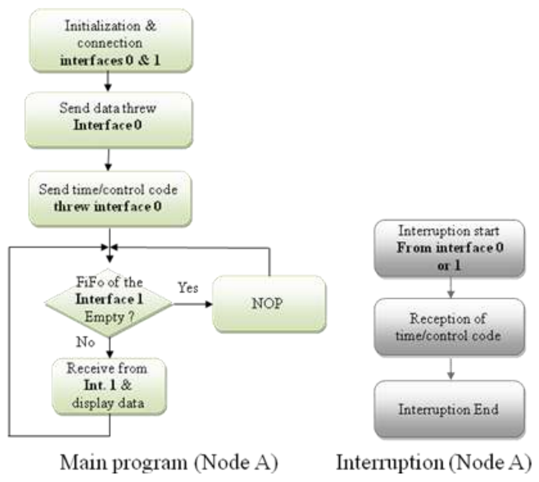
Initialization & connection interfaces 0 & 1
Send data threw Interface 0
Send time/control code threw interface 0
FiFo of the Interface 1 Empty ?
Yes
NOP
No
Receive from Int. 1 & display data
Interruption start From interface 0 or 1
Reception of time/control code
Interruption End
Main program (Node A)
Interruption (Node A)

Figure 6. The Node A distributed program

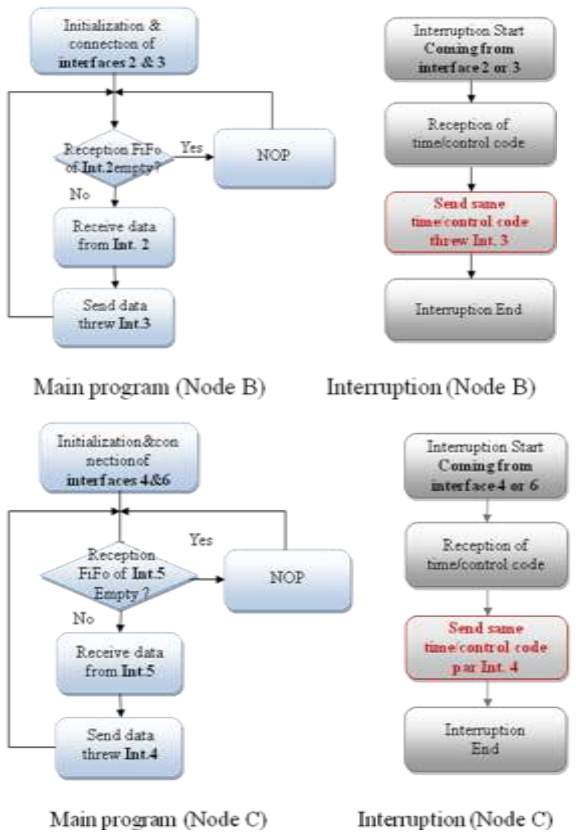
Initialization & connection of interfaces 2 & 3
Reception FiFo of Int.2empty?
Yes
No
NOP
Receive data from Int. 2
Send data threw Int.3
Interruption Start Coming from interface 2 or 3
Reception of time/control code
Send same time/control code threw Int. 3
Interruption End
Main program (Node B)
Interruption (Node B)
Initialization&connection of interfaces 4&6
Reception FiFo of Int.5 Empty ?
Yes
No
NOP
Receive data from Int.5
Send data threw Int.4
Interruption Start Coming from interface4 or 6
Reception of time/control code
Send same time/control code par Int. 4
Interruption End
Main program (Node C)
Interruption (Node C)

Figure 7. The Node B and C distributed programs

As for the dual nodes network, the transmission and reception of data takes place properly. The nodes communicate correctly. The transmitted data does a complete turn and returns back to its issuer. The Network is well operational.

V. Tests and Validation of Designed Systems

In this work, we have developed NoC architectures based on the SpW communication protocol which is widely used as a proved reliable interface standard on-board spacecrafts. Our work focused on implementing a SpW CODEC and its integration into point to point networks architectures.

The designed IP has several features such as:

- It operates at a frequency of 100 MHz while maintaining the standard (flow and exchange time of silence);
- It is synthesizable and routable;
- It does not load its FiFo Randomly (total synchronization with the host);
- It connects and operates normally when it is associated to a host.

In order to test the SpW CODEC IP, we have programmed various codes that emulate a HOST connection with a specific behaving scenario. After each execution of a test, the temporal evolution of signals has been visualized. So we could conclude on the smooth running of the script and compliance behavior of the interface.

In addition, this IP has been associated with a host and tested with different configurations of NoCs. The obtained results showed the efficiency and flexibility of the proposed solution for systems based on FPGA circuits.

The digital clock manager DCM of the used FPGA (Virtex II Pro) is limited to 100 MHz, which limits our communication rate to 50 Mbits/s. The use of more recent FPGA circuits such as the Virtex 6 would increase the number of nodes and the transmission frequencies.

VI. Conclusion

In this paper, we study various NoC architectures based on the SpW protocol. These solutions integrate the SpW IP codec developed as part of this work and are achieved by the introduction of point to point interconnections between different nodes. Each node consists of a host processor associated with one or more SpW interfaces (IP codec).

Compared to other related works like SoCwire Or MARC architecture, we built simple but versatile and intelligent nodes that can be used to develop more complex applications. In fact, the main benefits of the proposed architectures are modularity, flexibility and usability. Since the Nodes are all connected directly to each other through the dedicated IP codec, (without any intermediate units) communication speed is increased, routing errors are reduced, and in case of an error or a disconnection between two nodes occurs, alternate way to transmit data through other nodes is always possible.

The number of necessary links and communications interfaces may appear significant In case of a large Network establishment but in reality it is not a consequent problem in front of the performance of the new generations of FPGA platform like Virtex 5 or Virtex 6.

Another advantage of our design is the SpacWire Codec peripheral wich is an OPB Perphiral. It can be easily integrated in any design or mechanism that includes that bus and establish a SpaceWire Comunication.

Once the initial tests done, it is possible to carry out more optimizations and improvements of the designed NoC architectures. In addition, some bits of the frame are reserved for future extension: they may be used for example to implement additional techniques to manage messages priorities and or to generate distributed interruptions between nodes.

References

[1] ECSS Secretariat ECSS-E-ST-50-12C, "SpaceWire - Links, Nodes, Routers and Networks", ESA Requirements and Standards Division, 31 July 2008

[2] B. M. Cook, C. P.H. Walker, "SpaceWire NetworkTopologies", International SpaceWire Conference,September 2007

[3] C.P. Bridges and T. Vladimirova, "Dual Core System-on-a-Chip Design to Support Inter-Satellite Communications",NASA/ESA Conference on Adaptive Hardware and Systems June, 2008.

[4] C.F. Kwadrat, W.D. Horne, B.L. Edwards, "Inter-Satellite Communications considerations and requirements for distributed spacecraft and formation flying systems", NASA/SpaceOps, 2002

[5] A. Senior, W. Gasti, O. Emam, T. Jorden, R. Knowelden, S. Fowell, “Modular Architecture for Robust Computation”, International SpaceWire Conference 2008.

[6] A. Senior, P. Ireland, S. Fowell, R. Ward, O. Emam, B. Green, “ModularArchitecture for Robust Computing (MARC)” , International SpaceWireConference 2007

[7] S. S. Mehra, R. Kalsen and R. Sharma, "FPGA based Network-on-Chip Designing Aspects", National Conference on Advanced Computing and Communication Technology | ACCT-10, India, 2010

[8] A. Janarthanan, “Networks-on-Chip based High Performance Communication Architectures for FPGAs”, M.S. Thesis, University of Cincinnati 2008

[9] R. Gindin, I.Cidon, I.Keidar, "NoC-Based FPGA: Architecture and Routing", First International Symposium on Networks-on-Chip-Cover, mai 2007

[10] F. Bubenhagen, B. Fiethe, H. Michalik, B. Osterloh, P. Norridge, W. Sullivan, C. Topping and J. Ilstad, "Enhanced Dynamic Reconfigurable Processing Module For Future Space Applications", Long Paper, International SpaceWire Conference, House of Scientists, St. Petersburg, Russia, June 2010

[11] B. Osterloh, H. Michalik and B. Fiethe, "Socwire: A Spacewire Inspired Fault Tolerant Network-On-Chip For Reconfigurable System-On-Chip Designs In Space Applications", Long Paper, International SpaceWire Conference, Nara Prefectural New Public Hall, Nara, Japan, November 2008

[12] B. Osterloh, H. Michalik and B. Fiethe, "SoCWire: A Robust and Fault Tolerant Network-on-Chip Approach for a Dynamic Reconfigurable System-on-Chip in FPGAs", ARCHITECTURE OF COMPUTING SYSTEMS, Lecture Notes in Computer Science, 2009, Volume 5455/2009

[13] B. Osterloh, H. Michalik, B. Fiethe, K. Kotarowski, "SoCWire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications", NASA/ESA Conference on Adaptive Hardware and Systems, June 2008

[14] B. Fiethe, H. Michalik, C. Dierker, B. Osterloh and G. Zhou "Reconfigurable System-on-Chip Data Processing Units for Space Imaging Instruments",Design, Automation & Test in Europe Conference & Exhibition, April 2007

[15] D. Guzmán, M. Angulo, L. Seoane, S. Sánchez, M. Prieto and D. Meziat, "Overview Of The Intaμsat’s Data Architecture Based On Spacewire", Short Paper, International SpaceWire Conference, House of Scientists, St. Petersburg, Russia, June 2010

[16] J.R.Paul and T. Vladimirova, "Design Of A Wireless Link For Spacewire Networks", Short Paper, International SpaceWire Conference, House of Scientists, St. Petersburg, Russia, June 2010

[17] K. Tanaka, S. Iwanami, T. Yamakawa, C. Fukunaga, K. Matsui and T. Yoshida, "The Design and Performance of SpaceWire Router-network using CSP", Short Paper, International SpaceWire Conference, Nara Prefectural New Public Hall, Nara, Japan, November 2008

[18] K. Tanaka, S. Iwanami, T. Yamakawa, C. Fukunaga, K. Matsui and T. Yoshida, "Proposal of CSP based Network Design and Construction", Short Paper, International SpaceWire Conference, Nara Prefectural New Public Hall, Nara, Japan, November 2008.

[19] M. Tanaka, N. Fukuchi, Y. Ooki and C. Fukunaga, "Design of a Transputer Core and its implementation in an FPGA", Proceedings of Communication Process Architecture 2004 (CPA2004) held in Oxford, England, pp.361-372, IOS press.

[20] D. Atienza, F. Angiolinic, S. Muralia, A. Pullinid, L. Beninic, G. De Michelia, "Network-on-Chip design and synthesis outlook", INTEGRATION, the VLSI journal 41 (2008) , 340–359.

[21] Agarwal, C. Iskander Survey of Network on Chip (NoC) Architectures & Contributions, "Journal of Engineering, computing and architecture", Volume 3, Issue 1, 2009.

[22] R. Dobkin, "Credit-based Communication in NoCs", Introduction to Networks on Chips: VLSI aspects, Technion, Winter 2007

[23] E. Salminen, A. Kulmala, and T.D. Hämäläinen, "On network-on-chip comparison", Digital System Design Architectures Methods and Tools, 2007.

AUTHORS PROFILE

S. HACHED received the Engineer degree (2009) and the MSc degree (2010) in Automation and Industrial Computing from the National Institute of Applied Sciences and Technology of Tunis. Currently, he is a PhD student in Electrical Engineering at the Ecole Polytechnique, Montreal Univeristy.

M. GRAJA received the Engineer degree (2008) and the MSc degree (2010) in Automation and Industrial Computing from the National Institute of Applied Sciences and Technology of Tunis. Currently, he is a development Engineer at the ASIC Company.

S. BEN SAOUD (1969) received the electrical engineer degree from the High National School of Electrical Engineering of Toulouse/France (ENSEEIHT) in 1993 and the PhD degree from the National Polytechnic Institute of Toulouse (INPT) in 1996. He joined the department of Electrical Engineering at the National Institute of Applied Sciences and Technology of Tunis (INSAT) in 1997 as an Assistant Professor. He is now Professor and the Leader of the “Embedded Systems Design Group” at INSAT. His research interests include Embedded Systems Architectures, real-time solutions and applications to the Co-Design of digital control systems and SpaceWire modules.