Real-time Processor Interconnection Network for FPGA-based Multiprocessor System-on-Chip (MPSoC)

Stefan Aust, Harald Richter

Department of Computer Science
Clausthal University of Technology
Julius-Albert-Str. 4
38678 Clausthal-Zellerfeld, Germany
e-mail: stefan.aust|harald.richter@tu-clausthal.de

Abstract—**This paper introduces a new approach for a network on chip (NOC) design which is based on a NlogN interconnect topology. The intended application area for the NOC is the real-time communication of multiprocessors that are hosted by a single Field Programmable Gate Array (FPGA). The proposed NOC is an on-chip multistage interconnection network for which an upper limit can be guaranteed that is at most needed for the latency while delivering data between sending and receiving processors. The reason for the deterministic interprocessor communication is the constant path length from input to any output port of the NOC. In contrast to contemporary NOCs, no intermediate routers exist. Thus, no overloaded router with hot spot problems can occur, and the proposed NOC can be used for real-time applications. Example NoCs of size 4x4 and 8x8 were implemented in VDHL, together with their softcore processors on Spartan3 and Virtex-4 and -5 FPGAs from Xilinx.**

Keywords–network on chip; multistage interconnection network; softcore processor; real-time multiprocessor; FPGA-based multiprocessor

I. INTRODUCTION

The increasing quantity of logic cells that can be integrated into a single FPGA allows novel solutions by using the system on chip (SoC) paradigm. Just recently, multiprocessor system on chip (MPSoC) applications have become feasible that are hosted by a single FPGA [1,2]. In such MPSoCs, each processor exists only as Verilog or VDHL [3] description that can be extended or modified as needed, and that is afterwards synthesized for a target FPGA such as Spartan3 or Virtex-4/-5/-6 from Xilinx, for example. Because of the adaptability of the processor architecture to the demands of the real-time system, such computing devices are called soft-core or soft processors.

MPSoCs with soft processors exhibit both, the high performance of parallel computers and the flexibility of reconfigurable hardware [4]. In real-time systems, data- and computing-intensive applications can make use of this technological progress. For instance, driver assistant systems in cars require to service more sensors and actuators than ever. Such applications demand higher computing power and less electrical power at the same time, while the system size has to be minimized. To match such demands, the proposed network on chip (NoC) design can be used in MPSoCs. In the future, we believe that MPSoCs will replace in part conventional electronic controller units in automobiles as well as in complex machinery [5].

The majority of embedded systems are located in real-time applications. Amongst others, the real-time performance of multiprocessor computers relies on the predictability of the interprocessor communication. For an MPSoC, deterministic behaviour of the interconnection network has to be guaranteed. This requirement is hardly to implement with conventional packet routing that takes place in direct, i.e. static networks. In static networks, adaptive multi-hop routing together with packet prioritization induces an undesirable indeterminism to network latency. The formation of hotspots due to excessive data traffic in router nodes excludes predictability also. We therefore propose, a new paradigm for MPSoCs, which makes use of multistage interconnection networks (MINs) as a network on chip.

This paper is organized as follows: in section 2, the state of the art in NoCs is given. Section 3 makes a recap of MINs. Their utility and their problems in on-chip usage are investigated in section 4. In section 5, the topology of the proposed NoC is presented, and in section 6 its chosen implementation is described, together with the MPSoC for which it was developed. The paper ends with conclusion and literature reference in section 7.

II. STATE-OF-THE ART IN NETWORKS ON CHIP

Interprocessor communication in MPSoCs with tens of cores or more is no longer feasible by using shared buses due to their low intrinsic scalability in bandwidth and latency [6,7]. Also crossbar structures are no longer practicable due to their $O(N^2)$ complexity if N becomes large. To overcome the von Neumann bottleneck and the $O(N^2)$ increase in hardware, alternative architectures have been introduced by NoCs as the new paradigm in SoC design [8]. Since then considerable number of NoC designs have been proposed which provide diverse communication types and network topologies [7,9]. NoCs with direct (static) networks have been proposed by [10,11,12,13] such as mesh, tree, torus or hypercube. Some examples of these static topologies are given in Fig. 1. The basic principle of direct network topologies is that each processor is connected directly to a smaller number of neighbour processors where each processor acts in addition as a switch or router node for

 ISBN: 978-1-61208-101-4

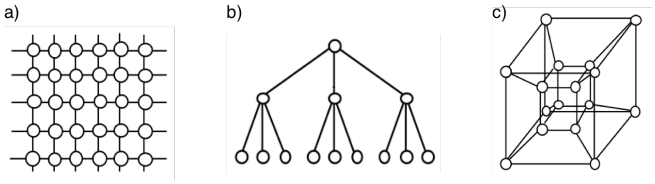

Figure 1. Toplogies of direct networks: a) mesh; b) tree; c) hypercube.

frames or packets, respectively. Routing is performed either statically or dynamically via the source and target information contained in the frame/packet headers. The communication channels between the processor nodes are operating on layer 3 of the ISO 7-layer model. Finally, some nodes provide additional communication channels for the necessary I/O.

A desirable property of NoCs for MPSoCs is scalability, which means that for small and large processor numbers as well, the same basic interconnect structure, can be used in principle. Another property which is mandatory for real-time systems is that the achievable bandwidth and the maximum latency for data transfer is deterministic i.e. predictable. This means that an upper limit for the latency must be guaranteed that is at most needed for data transfer, as well as a lower limit for the bandwidth. This is required to build and program systems that can react in due time.

However, all direct networks have the potential hazard of hot spots that are overloaded router nodes. Hot spots result in unpredictable bandwidth and latency, which in turn is not tolerable for real-time systems. Furthermore, scalability is also not possible if hot spots occur.

This is why we propose an alternative to direct networks that can be used for NoCs in MPSoCs and that is based on indirect networks. In indirect networks, computing nodes are connected via a cascaded set of switches. Because of the switch arrangement, each path from source to target is of the same length, and every switch has to serve only a fixed number of traffic streams. Thus, hot spots cannot occur, if the rearrangable non-blocking subtype of indirect networks is used. Indirect networks are described in the next section.

III. MULTISTAGE INTERCONNECTION NETWORK (MIN)

By origin, MINs were proposed for telephone exchange systems, and later for parallel computers. Vector supercomputers, multiprocessors and multicomputers with processors on individual silicon chips were introduced two decades ago for high-performance computing. MINs have been designed to match their bandwidth and latency constraints and to support effective execution of parallelized algorithms. Therefore, MINs are known from parallel computers, and we have adopted these structures to provide for deterministic on-chip interprocessor communication.

MINs connect computing nodes through a set of elementary switches that are organized in 1, 3 or logN stages, where N is the number of the ports the network features. The mathematical patterns between the switch stages are permutation functions, such as perfect shuffle, butterfly or bit reversal [14]. The Omega network, for example, which was introduced by Lawrie [15], consists of shuffle and exchange permutations in logN stages and can be defined by

$$\Omega_n = (\sigma_n \circ E)^n \quad (1)$$

where n is the total number of stages, σ is the shuffle permutation over n bits, and E is an exchange stage [16]. An example for an Omega network of size 8x8 is given in Fig. 2.

By using the smallest possible switch size of 2x2, the construction of a MIN needs $(N/2)\log_2 N$ switches only, which is the minimum number possible at all. For comparison, a crossbar network requires N^2 switches. Typical representatives for logN-MINs are Omega, Baseline and Butterfly networks [16,17,18,19]. They belong to the so-called delta subclass of MINs which means that routing through the logN-MIN is easily accomplished by using bit after bit of the target port address in order to set each switch so that it routes data either „=“ (parallel) or „x“ (cross). Because of the constant number of switches that data has to pass from the network input port to output port, a constant routing time or at least an upper limit for the routing canbe guaranteed. MINs are therefore beneficial for interprocessor communication with respect to latency, which is important for real-time applications. However, constant routing time cannot be guaranteed for transfers that take place at all input ports simultaneously. The reason is that each output can be reached from every input in principle, but there are permutations of inputs to outputs that can not be realized which is why MINs are called fully reachable but blocking. This is the main disadvantage of logN-MINs.

There are two other categories of MINs, which are called Clos and Benes networks that do not belong to the logN type and that are non-blocking [19]. Unfortunately, the Clos network has a switch complexity of $(3/2)N\sqrt{N}$, and the Benes network has $N\log_2 N$ complexity which is both not the minimum logN-MINs have. This means for the applicability of Clos and Benes MINs as on-chip networks that they consume more chip area as needed, and that they need more electrical power as logN-MINs do. Both are disadvantages for VLSI integration. Furthermore, Clos and Benes networks are non-blocking only for the price of rearranging already existing internal paths through the network, which is a problem for ongoing real-time transmissions. During path rearrangement, no data transfer can take place. Finally, path

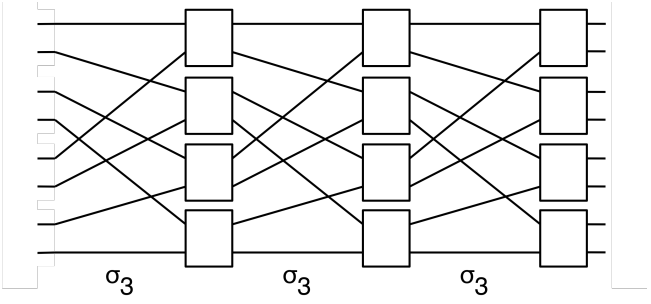

Figure 2. Omega topology of size 8x8.

rearrangements require a central control instance that executes the rearranging algorithm. However, a central control unit prevents from scalability, and the rearranging algorithm is so complex that it consumes a processor by its own, together with considerable computing time.

IV. IDEAL ON-CHIP MIN

In general, we can state that there is no ideal on-chip interconnection network with good scalability, deterministic routing latency for real-time capability, minimum chip area and minimum power consumption at the same time. However, with the introduction of FIFO buffers, each disadvantage of the above networks could be solved, at least for many practical applications. In the case of logN-MINs for example, FIFOs at all switch stages are needed as temporary storage for incoming data to hold them until the blocking situation is managed. Blocking management has to be made every time when the MPSoC requests a forbidden permutation from input to output ports. Delaying and serializing the needed transfers via FIFO accomplish this. Data is then delayed until blocking is over, and afterwards data is read out from the FIFO one after the other. A positive aspect is that blocking management can be achieved in a fully decentralized manner.

In the case of Clos and Benes MINs, FIFOs are required only at the input ports to store incoming data until the internal path rearrangements have been accomplished. If a permutation from input to output needs rearrangement, then the input FIFOs are filled while the network is drained. When all network-internal paths are empty, rearrangement can take place by setting switches newly. After that, data is let again into the network.

In both cases, the FIFO solution is not perfect because it introduces indeterministic delays in the MPSoC interprocessor communication. Depending on the filling state of a FIFO and depending on the needed transfers per time unit, more or less data frames or packets have to be temporarily stored in the FIFOs. Only by means of a fixed FIFO depth, an upper limit for the maximum latency can be stated for data delivery. However, this is sufficent in practice for many real-time applications. FIFO overflow can occur, but it is considered as a programming fault of the MPSoC. It has to be mentioned here also that is not the fault of the network but of the programmer if two input ports want to deliver data to the same output port at the same time. This is comparable to writing the same variable in a shared memory from two processors at the same time.

To summarize, the state of the art in on-chip networks is that logN-MINs are the best option because of their O(NlogN) scalability and their minimum chip and power consumption compared to busses, crossbars, Clos and Benes networks. Therefore, we propose a logN-MIN as the preferred NoC for MPSoC. In the next section we will explain which type of logN-MIN is best suited, and what we did to improve its real-time behaviour.

V. TOPOLOGY OF THE PROPOSED MINOC

The network topology we decided for is known as Baseline network [20]. In Fig. 3, a Baseline network of size 16x16 is presented, together with two routing examples. This topology has been introduced in 1980 by C. Wu and T. Feng to proof equivalence among logN-MINs. The stages in the Baseline network are connected via an unshuffle wiring. The topology of the Baseline network is mathematically isomorphic to other networks of the $\log_2 N$ class but the network has technological advantages compared to other logN-MINs. The production of the Baseline network is characterized by a recursive construction. Each stage is of 1,2,4,... sub-networks of the same type. From the view-point of the first stage, the Baseline consists of one switch block

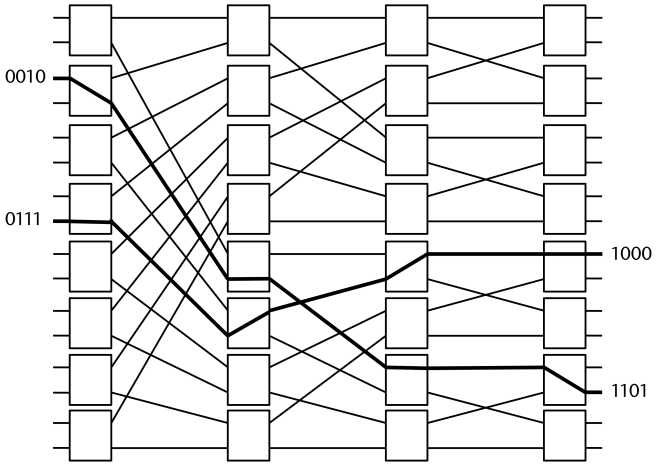

Figure 3. Baseline topology of size 16x16.

of size NxN. The second stage contains two switch blocks of size (N/2) x (N/2) and so forth. That iterates down to the smallest blocks of size 2x2 as atomic elements. In addition, each stage has the minimum possible number of crossing wires, and the wires have minimum lengths [14]. Both features, recursive construction and minimum wiring, are advantages for implementation in VLSI or FPGA that are not found in other logN-MINs. In Baseline networks, the routing algorithm evaluates the most significant bit of the n=logN bits of the target address first [16]. With every bit evaluation, the interval of possible output ports is halved. After n steps, the target output port is exactly specified. This routing algorithm is a good example of the divide-and-conquer principle known from theoretical informatics. Finally, the recursive construction of the Baseline network eases its definition in VHDL. The VHDL code of a 2x2 switch is for example:

```
signal A, B, C, D: std_logic_vector(0 to 31);
shared variable S: boolean;

C <= A when S = false -- parallel connection
else B;               -- cross connection
D <= A when S = true  -- parallel connection
else B;               -- cross connection
```

where A and B are inputs, C and D are outputs and S is the switch state. Not shown are FIFO buffers, but as we have learned from practice, the FPGA synthesis of the switch controller is much more complex than the switches and the FIFO. With our preferred MINoC, we yield a MPSoC of the symmetric multiprocessor type that is depicted in Fig. 4. Its architecture includes softcore processors P, local memory M_P and shared memory M_S.

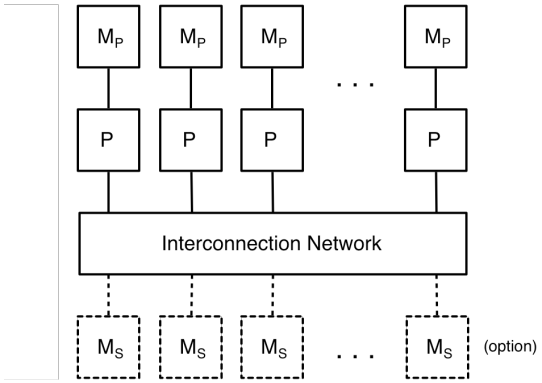

Figure 4. Block diagram of the resulting MPSoC Architecture.

With this architecture, both programming paradigms of message passing and shared memory are supported simultaneously.

VI. IMPLEMENTATION OF THE MINOC

The MINoC is implemented by adding a custom VHDL core to the existing description of a Xilinx Microblaze soft processor [21]. The block diagram of the so enhanced MicroBlaze is shown in Fig. 5.

Our custom IP core realizes the MINoC for the MPSoC. It consists of three components: 1.) switches 2.) network interface, and 3.) network controller. The first component (switches) implements the described Baseline topology. The second component is the network interface. It connects the MicroBlaze via its proprietary FSL bus [22,23] to the MINoC input ports and output ports. The third component is the network controller, which we have introduced to improve real-time behaviour. The network controller allows for interprocessor communication only in fixed points in time. This can guarantee a better upper limit for latency in data delivery.

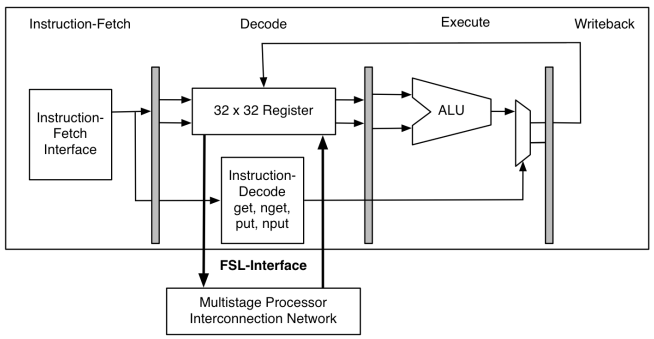

Figure 5. Block diagram of the soft processor enhanced by a MINoC.

A. Switches and Wiring

In the following sections of this paper we refer to message-based interprocessor communication. However, with the network coupling of shared memory (M_S) interprocessor communication via shared variables becomes feasible as well.

As seen before, the switches feature two states for direct and crossed connection paths, but for parallel computing mechanisms for task synchronization are needed also. These can be implemented by two additional switch states called upper and lower broadcast (Fig. 6). Direct and crossed connection paths are used in point-to-point communication

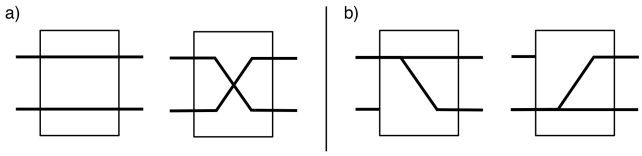

Figure 6. States of the switch: a) straight and cross b) upper and lower broadcast.

between sender/receiver pairs via message passing. Broadcast communciation is needed for synchronous task start and stop and for distributing input data to processors. Both types of communication are required in the intended application domain of real-time embedded systems.

The wire patterns between the stages are implemented via bidirectional communication channels that are defined as signals in VHDL. With bidirectional channels, handshake protocols between sender and receiver are implemented.

B. Network Interface

The soft processors are connected to the interconnection network via the Fast Simplex Link (FSL) from Xilinx [23]. Each FSL interface provides an uni-directional point-to-point communication channel that includes a FIFO buffer. The FIFO buffer decouples the processor clock from the network but it introduces indeterminism as described that cannot be avoided. However, the network controller reduces the jitter in message latency during data transfer. Since the FSL interface is an internal part of the soft processor, communication takes 2 processor cycles only for transferring a 32-bit word from sender to receiver if the FIFO buffers are free. Therefore, the FSL enables a high-speed interprocessor communication. When the FSL interface is added to the soft processor, the MicroBlaze instruction set is augmented with four additional instructions:

- Blocking Read (get)
- Non-blocking Read (nget)
- Blocking Write (put)
- Non-blocking Write (nput)

These instructions are for reading from and writing data to the FIFO of the FSL interfaces. As soon as data are written to FIFO, the put instruction terminates and the NoC moves all data from source to destination FIFO. Finally, a get instruction reads the data out of the receive FIFO.

 ISBN: 978-1-61208-101-4

C. Network Controller

The MIN operates not by packet- or message switching but circuit switching. This provides a direct connection between sender and receiver and delivers maximum speed for interprocessor communication since no frame or packet header is required that would be overhead only. Thus all data are received in sent order.

Furthermore, a time-slot that is returning periodically is granted to each processor according to a scheduling policy where messages can be sent. The scheduling policy has been implemented in the network controller in hardware by means of VHDL. When the network controller schedules a data transfer, a communication time-slot opens, and the network controller establishes a physical path between a pair of processors. As long as the time-slot stays open, data can be sent directly from sender to receiver with full speed of the processor interfaces. After a time-slot has closed, the next path will be opened round-robin. In our tests we have used preemptive scheduling with a fixed slot time. For this purpose, a central network controller was implemented and tested. This network controller serves all connection requests from the processors. Preemption is made if a processor whose time slot has arrived does not want a data transfer through the network. The desribed scheduling policy is identical to task scheduling in real-time operating systems. The usage of a scheduling algorithm for data transfer results in better predictability to the network latency which suffers from the indeterminism of the FIFOs. Furthermore, several scheduling algorithms are possible, such as priority scheduling or earliest-deadline-first which are known from real-time operating systems.

D. Overall Architecture

The entire MPSoC including the MINoC has been implemented and tested with evaluation boards carrying FPGAs from Xilinx of the Spartan-3, Virtex-4 and Virtex-5 types. As boards have been used the ML 505 from Xilinx with a Virtex-5 FPGA, the XpressFX100 from PLDA with a Virtex-4 and the Spartan-3 starter kit board from Digilent with a Spartan-3 chip. With Spartan-3, a 4x4 network was implemented together with 4 MicroBlazes on the same

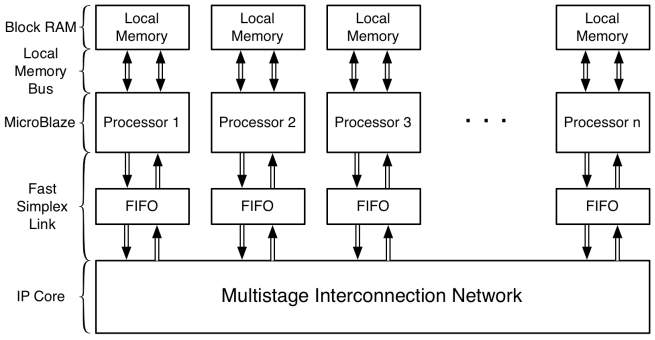

Figure 7. Overall architecture of the MPSoC.

FPGA. On the Virtex-4 and the Virtex-5 board, an MPSoC with a MINSoC of size 8x8 has been implemented. All processors are Xilinx MicroBlaze softcores that emulate a 32-bit RISC processor. Each soft processor features private local memory with Block RAM for instructions and data. Multiple Block RAMs are linked to processors via the Local Memory Bus (LMB) [21]. All MicroBlazes in turn are coupled to the interconnection network via the FSL-MIN interface as shown in Fig. 5. An overview of the system architecture as it was implemented is given in Fig. 7.

With using the FSL processor interface we can measure that it takes two clock counts of the processor to write or read a single 32-bit data to or from the network interface. Once the connection from sender to receiver has been established no additional latency of the circuit-switched network exists, so that a network clock rate of 100 MHz induces a real data transfer rate of 1.6 Gbit/s per connection. Depending on the topology of the network multiple connections between sender/receiver pairs can be established at the same time without additional latencies. Higher network clock rates are feasible because the FSL can be operated in asynchronous mode. The maximum frequency depends on the FPGA chip that is used for implementation.

Our design studies have shown, that the network topology can be implemented in FPGA hardware without problems. The challenge is the implementation of the network controller with its routing algorithm in an efficient way. But, once the network is fully implemented in FPGA hardware, the MINoC provides a deterministic high-performance network for interprocessor communication in MPSoCs.

VII. Conclusion and Future Work

This paper proposes an on-chip multistage interconnection network with the least possible number of hardware, the minimum amount of wiring between stages and the minimum wire lengths. It can be used for high-performance interprocessor communication in real-time applications. Although logN-MINs have been already researched and used in parallel super computers, they can be adapted also for network-on-chips as well. High bandwidth and low latency are combined with a deterministic behavior of interprocessor communication in the proposed NoC. The objective is to use MPSoCs in high-performance embedded systems with hard real-time constraints that can be found in electronic control units for cars or for production machinery.

In future work we want to discuss the pros and cons of blocking and non-blocking MINs for real-time computing and their implementation in FPGA hardware. Blocking networks such as the Baseline network minimize the costs in hardware but they require a suitable scheduling strategy because not all permutations of sender/receiver pairs can be realized. Therefore further scheduling algorithms such as priority scheduling or earliest-deadline-first have to be considered for hardware implementation. In comparison with blocking networks, non-blocking networks as the Benes network can be operated without complex scheduling strategies since messages can be sent to each free receiver port at any time. On the other hand non-blocking networks exhibit extra costs in hardware plus they require complex routing algorithms due to the rearrangement of alternative connection paths.

 ISBN: 978-1-61208-101-4

REFERENCES

[1] A. Jerraya and W. Wolf, "Multiprocessor Systems-on-Chip," San Francisco, CA: Morgan Kaufmann, 2005.

[2] M. Grant, "Overview of the MPSoC design challenge," in Proceedings of the 43rd annual Design Automation Conference, San Francisco, CA, July 2006, pp. 274-279.

[3] U. Heinkel, M. Padeffke, W. Haas, and T. Buerner, "The VHDL Reference," Cichester, England: John Wiley & Sons, 2000.

[4] C. Bobda, T. Haller, and F. Muehlbauer, "Design of Adaptive Multiprocessor on Chip Systems," in SBCCI 2007, Rio de Janeiro, Sept. 2007, pp. 177-183.

[5] S. Aust and H. Richter, "Space Division of Processing Power For Feed Forward and Feed Back Control in Complex Production and Packaging Machinery," in WAC 2010, Kobe, Japan, Sept. 2010

[6] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet switched interconnections," in DATE2000, March 2000, pp. 250-256.

[7] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," in ACM Transactions on Design Automation of Electronic Systems, Vol. 12, No. 3, Article 23, August 2007.

[8] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," in IEEE Computer magazine, vol. 35, no. 1, Jan. 2002, pp. 70-78.

[9] D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Benini, and G. De Micheli, "Network-on-Chip design and synthesis outlook," in Integration, the VLSI Journal, 41(2), Feb. 2008.

[10] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani, "A Network on Chip Architecture and Design Methodology," in Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2002, pp. 105–112.

[11] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," in Integrated Circuits and Systems Design (SBCCI) 2003, pp. 169-174.

[12] C. A. Zeferino and A. A. Susin, "SoCIN: A Parametric and Scalable Network-on-Chip," in ACM Computing Surveys (CSUR), vol. 38, issue 1, 2006.

[13] L. Benini and G. De Micheli, "Networks on Chips: Technology and Tools," San Francisco, CA: Morgan Kaufmann, 2006.

[14] H. Richter, US-Patent 5,175,539 "Interconnection Network".

[15] D. H. Lawrie, "Access and Alignment of Data in an Array Processor," in IEEE Transactions on Computers, vol. C-24, no. 12, December 1975, pp. 1145-1155.

[16] H. Richter, "Verbindungsnetzwerke für parallele und verteilte Systeme," in German, Heidelberg: Spektrum, 1997.

[17] J. L. Hennessy and D. A. Patterson, "Computer Architecture. A Quantitve Approach," 4th edition, San Francisco, CA: Morgan Kaufmann, 2007.

[18] J. Duato, S. Yalamanchili, and L. Ni, "Interconnection Networks. An Engineering Approach," San Francisco, CA: Morgan Kaufmann, 2003.

[19] W. Dally and B. Towles, "Principles and Practices of Interconnection Networks," San Francisco, CA: Morgan Kaufmann Publishers Inc., 2003.

[20] C.-L. Wu and T.-Y. Feng, "On a Class of Multistage Interconnection Networks," in IEEE Transactions on Computers, Vol. C-29, No. 8, August 1980, pp. 694-702.

[21] Xilinx Inc., "MicroBlaze Processor Reference Guide," Product Specification: UG081, v9.0, Jan. 2008

[22] Xilinx Inc., "Fast Simplex Link (FSL) Bus (v2.11b)," Product Specification: DS449, June 2009

[23] H. P. Rosinger, "Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel," Xilinx Application Note: XAPP529, v1.3, May 2004

 ISBN: 978-1-61208-101-4