



Measuring the Efficiency of Routing Architecture for DyNOC Using Novel Method

Y.L.Ajay Kumar

Dr. D. Satyanarayana

Dr. D. VishnuVardhan

Research Scholar,

Professor & Controller of Examinations,

Assistant Professor

Department OF E.C.E.,

Department OF E.C.E.,

Department OF E.C.E.,

JNTUA, Ananthapuramu.

RGM CET, Nandyala.

JNTUA, Ananthapuramu.

ajaykumarphd@gmail.com

dsn2003@rediffmail.com

vishnu.ece@jntua.ac.in

Abstract:

SOC has a maximum amount of integration of technology in a less space for providing efficient communication. For efficient communication SOC's uses traditional bus based architecture. But this bus based architecture is helpless, when the huge traffic is arrived. To address the above problem Dynamic Network On Chip (DyNOC) came in to the existence. In which the above traditional bus architecture is replaced with PE's (Processing Elements) and Agents. Dynamic Network On Chip (DyNOC) is having loop back module, which is distinct feature of DyNOC, which helps to dynamically re-route the data packets. If the neighboring PE's becomes unavailable all of a sudden. The routing operation is governed with the help of agents. With these maximum efficient routing can be achieved in NOC.

Key Words: DyNOC, packet injection, data integrity, processing element, routing agents, routing efficiency, FPGA NOC, NOC.

Introduction:

VLSI (Very large scale integration) is the measure of introducing IC 's on a single chip as the thousands of combination of transistors. VLSI started this in 1970s where the period of developing complex semiconductor and communication technologies. One of the important devices related to the VLSI is micro processor. The system advances added more number of transistors and more independent functions were integrated at a time. To fabricate one or more logic gates on single chip a few devices held by first IC's. At one time, it is very difficult to name and calibrate of various levels

of large scale integration (LSI) which consists of thousands of logic gates. The terms which suggest greater than VLSI levels of integration are no longer in widespread use. In all microprocessors we use to say VLSI or better given the common assumptions.

Each chip in semiconductor consists of transistor. In current technology millions of gates and hundreds of millions of transistors on today's microprocessor. Above VLSI we can use ultra-large scale integration (ULSI) on common device has rendered such fine distinctions moot.

As in early 2008, there was availability of billion transistors commercially. For example, Intel's monteci to titanium chip. As semiconductor fabrication moves from current generation of 65nm to next 45nm generation expected as more common place. In new challenges the example is NVIDIA's 280 series GPU.

This microprocessor is unique that its 1.4 billion transistor count, capable of a teraflop of performance, it is entirely dedicated to logic (Titanium's transistor count is due to 24MB L3 cache large). To oppose the earliest devices, we use current designs automation and automated logic syntheses to layout the transistors, in result of logic function enable the higher levels of complexity. To ensure more efficiency we have high performance logic blocks like SRAM cell.

II Related Work:

Some work focuses on reduction of hop counts using some communication architectures. In this existed system we have hybrid scheme which share the common physical channel for traverse of packets through packet switching connections and circuit switching connections with circuit setup in turn. Virtual circuit switching has the feature to reduce the latency and power consumption in the network. We focus on some network topologies and some algorithms to improve the performance of the NOC by literatures we cannot reduce the route to link latency and power.

The traffic will be reduce by the bufferless connections in the network to reduce the power consumption. This is explained by two methodologies: one method focus on optimizing route by micro architecture.

To achieve high throughput and low latency we use switch allocator. Another method focus on integrating the bypass path for complex routing process. A circuit setup which we have leads to the waits the packet until its time interval is out and given path to new packet of circuit configuration. The hybrid scheme based on VIP connections will trace the traffic pattern to set low latency and power consumption possible through the algorithm. The communication problem in circuit PS connections restrict the number of CS connections limit the latency and power consumption in NOC. The related work done in the existed system to reduce the latency and power is as follows:

- In this paper, we first introduce the 4*4 mesh connected network with PE, Agents and Linked connections and the related modified router architecture is introduced to reduce the latency and power by the use of bypass path through the crossbar switch.
- To reduce the latency and power we introduce the virtual channels at the path of the virtual circuit switching.

This paper focuses on the modified router architecture which is different from hybrid virtual circuit switching. The novel switching mechanism is used to keep the problems of hybrid scheme out and heavy congestion in traffic to reduce the latency and power. The paper is organized as follows: section III describes design and algorithms used in the design. Section IV describes verification through simulation. Section V describes the implementation flow details on the FPGA. We present the performance details of the system in section VI. The concluding remarks are given in the section VII

III Design:

In this Design the DyNOC architecture contains processing elements interconnected with links in a 2D mesh network in the 4X4 format. All processing elements are Linked with their neighboring processing elements. A global route interconnects all the peripheral processing elements. Global route is the significant design feature of this project which is facilitated by a simple routing management switch at the periphery. Apart from the interconnections, every PE has an input bus and an output bus through which the host can inject original packets and also, read the processed packets. The arrangement is as depicted in the Figure 1.

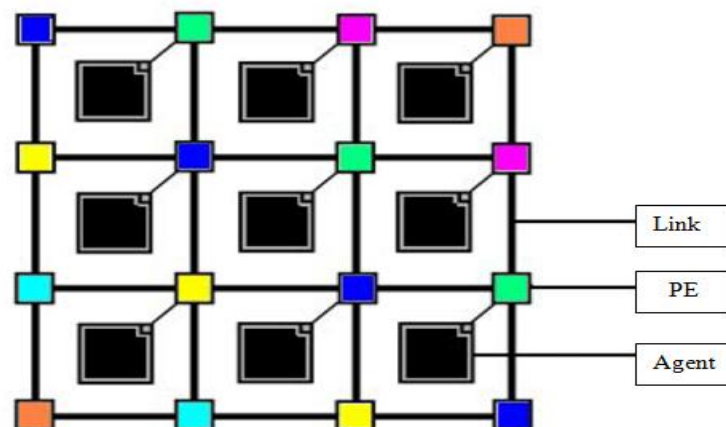


Figure 1: Arrangement of PEs in 4X4 mesh optimized for FPGA implementation

The control state machine is responsible for controlling the processing related activities such as depicted in Figure 2.

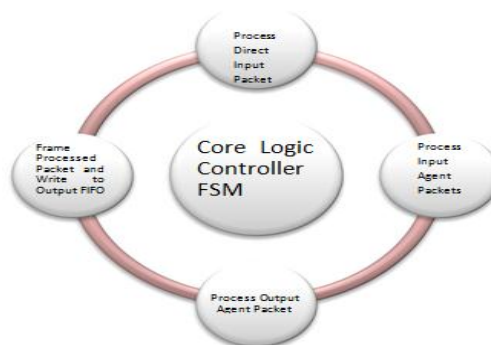


Figure 2. Activities of the Core Logic Controller FSM

The incoming packets are structured in a certain specific way to maintain their identity. For the data to be processed we add many headers. The following picture shows how the packets look like. After the packets are received and processed, the headers should be updated. This is what a framer does. Output Agent is responsible for managing all the outgoing packets. This employs the round robin algorithm and priority based routing as a secondary algorithm to send outgoing packets to either the neighboring elements or the external packet manager to the host. Core Logic is the functional block. In this project, we will implement a cryptography core for evaluation purposes. It has encryption and decryption blocks [2] which give us an intrinsic advantage for evaluating our design. North, East, West, South ports through which processing element will talk to the neighboring elements and also connected to the global interface. Packets arrive into the processing element through the input port and the packet will read outside through the output port. The protocol used here is a round-robin method, however any other protocol can be chosen to be implemented depending upon specifications. Since importance is given to the inbuilt input and output agents in this design, its internal architecture should be given apt importance. Both input and output agents have internal FIFOs to handle packet asynchronicity. Figure 1 and 2 shows the control flow of the input and output agents.

The input agent has a FIFO and looks for any requests from the neighboring elements. If there are any requests, depending upon the FIFO status, the incoming data is captured and a grant signal is asserted.

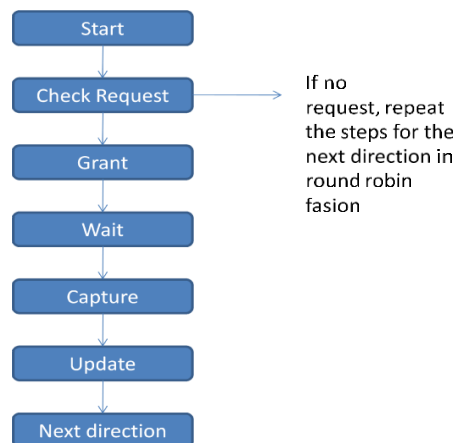


Figure 3: Control logic flow of the input agent

Output agent has two FIFOs: a buffer FIFO and another FIFO for storing the processed packets that are to be read out. The buffer FIFO has all the unprocessed packets forwarded from other PEs and they are read in OA (output agent) cycle of the core controller logic FSM. See Figure 4.

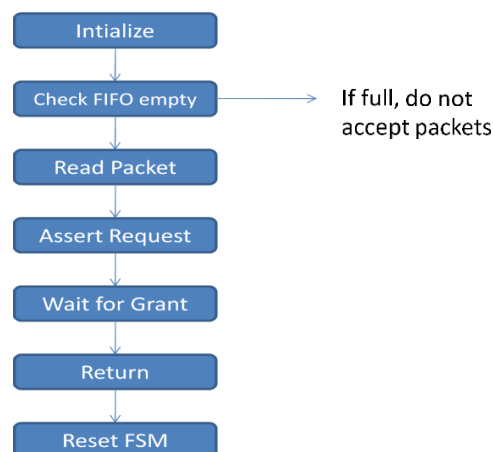


Figure 4: Control logic flow of the output agent

In summary, the controller FSM checks for the input packets, reads it and updates the inputs of the processing core, which in this case is a cryptography core [2]. Then, once the processing is complete the data is reframed into a packet and forwarded to the output agent. If any of the packets are forwarded by the neighboring element they are loaded into the FIFO of the input agent. If an unprocessed packet was not forwarded to the neighboring PE, it is buffered into buffer FIFO of output agent. The design also considers intimating the host if all the FIFOs come to a saturation. It is important that the design should be hardware implementable. Figure 5. Shows the FPGA implementation ready PE along with its ports. Figure 5. Show the top-level design of the NOC enclosing processing elements connected in the form of a mesh as shown in Figure 1. These are generated by the Xilinx ISE 13.2 tool. As this design is created to be tested on an FPGA, it is

important that some communication feature should be used to transfer some live data from a host computer to the FPGA board. The setup is shown in Figure 7.

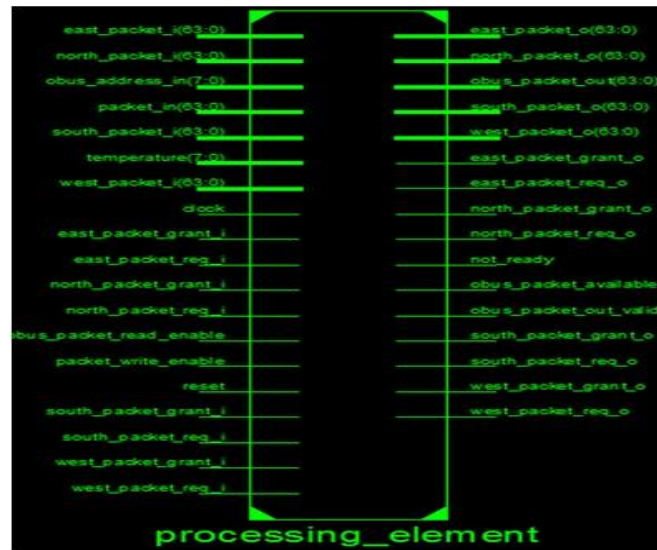


Figure 5. PE with its ports

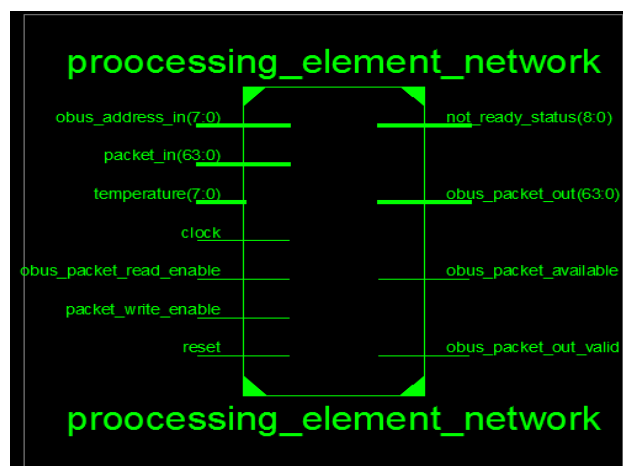


Figure 6. NOC top-level enclosing the PEs

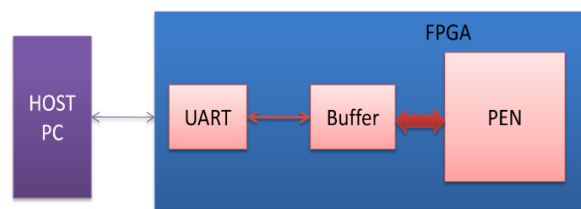


Figure 7. FPGA and PC for real-time sending and receiving data

IV Verification:

Two separate test benches are written one for the PE alone and another for the complete network. Testing process includes creating packets of randomized data and injecting into the design.

Figure 8. Shows how packets are written injected into the PEs. Simulation is done using the Xilinx ISE design suite. The control state machine cycles through the signals coming through the input port, input agent and the output agent's buffer FIFO.

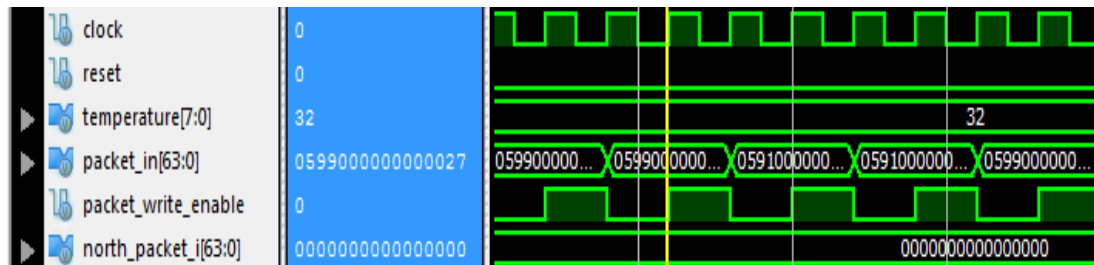


Figure 8. Packets being written into the PE

Figure 9. Shows how the unprocessed packet is forwarded to the neighboring elements. In the following situation, since the temperature is high at an index of 100 PE at location first row and first column designated as PE11, is unable to accept data for processing and whatever packet is accepted is sent to the output agent for forwarding. Here data packets are pumped in continuously to do a stress test on PE11. As a result, the packets are placed on all the busses on north, east, west and south and request signal is asserted. In Figure 9., it can be clearly seen that the element to the east of PE11 has given a grant and it is continuously accepting data. This is expected to happen until PE to the east of PE11 becomes busy.

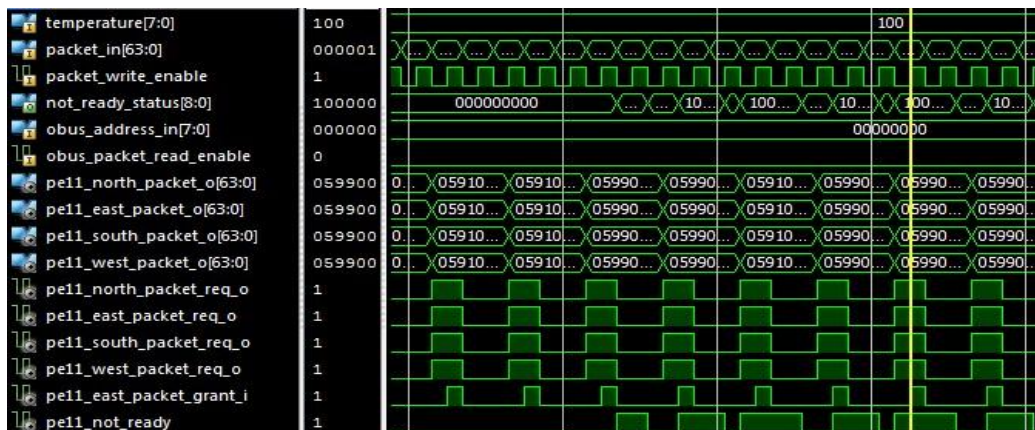


Figure 9. Unprocessed packets being distributed to neighboring elements

The design created for the FPGA implementation as shown in figure 9 has the following behavior. The packet is sent into the FPGA through UART communication. The packet is accepted by the input controller (Figure 1) for distribution to the NOC. Once the packet is processed the obus_packet_available signal is asserted because of which the output controller (Figure 1) will read the packet from the respective PE by addressing it. The data will be available on the obuf_dout

signal as shown in the below Figure 10. Finally, the read out packet is sent back to the host which initiated the process.

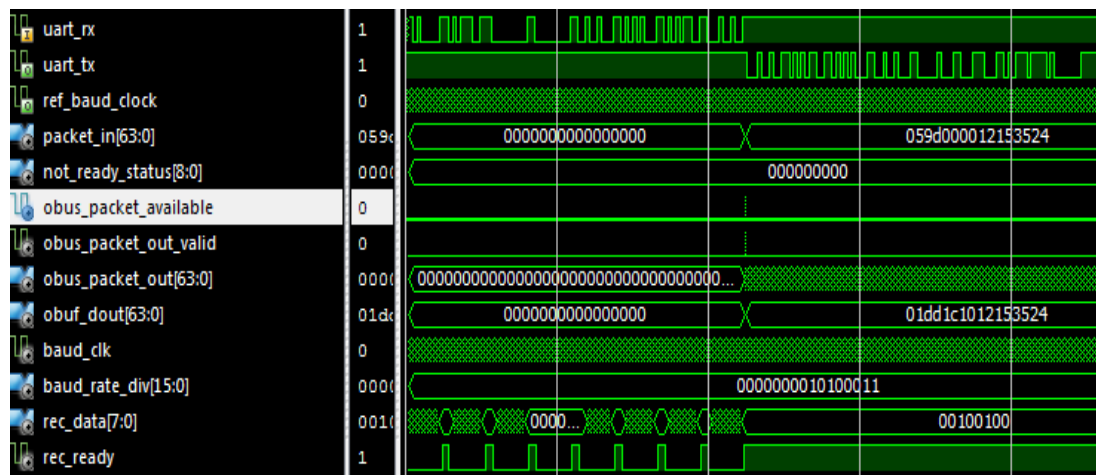


Figure 10. Real time data processing scenario

V Performance analysis:

In work [6] a 3x3 NoC is implemented. The following table gives a clear picture of the comparative study done with respect to our implementation. We are able to achieve consistent results across various buffer depths. It is possible because the memory in our design is optimized no matter what the buffer size and is always implemented in the BRAMs of FPGA. The logic in such a type of implementation does not get affected and thus gives consistent results.

Buffer Depth	Area[6]	Area	Frequency[6]	Frequency	Max. Delay[6]	Max. Delay(ns)
4	28427	2449	109.417	152.76	3.244	5.266
8	28724	2449	109.404	152.76	3.244	5.266
16	31083	2449	109.377	152.76	3.244	5.266
32	31503	2449	109.377	152.76	3.244	5.266

Table 1. Consistent area and operational frequency of our design

A very interesting study is carried out in [7] and we compare the design with the implementation. Figure 11. and 12. Shows that our design implementation is relatively better in most of the cases.

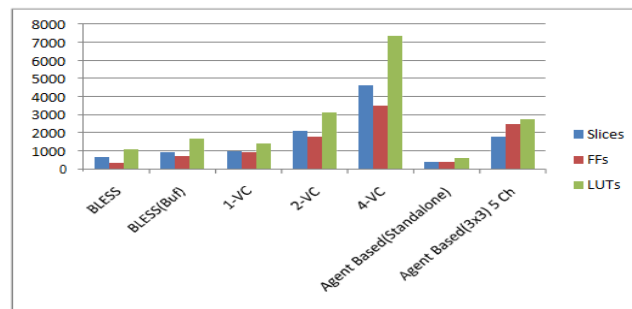


Figure 11. Comparison of agent based design with respect to area related resources

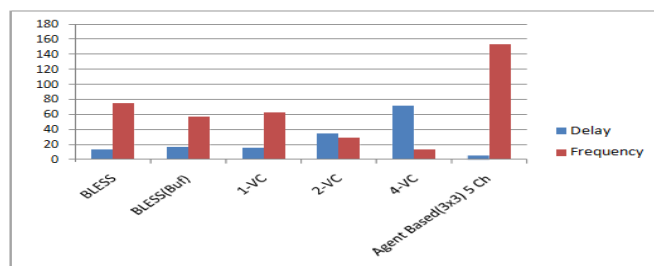


Figure 12. Comparison of agent bases 3x3 NOC with design in [7] with respect to delay and frequency

Another design in [9] uses a 2x2 network named EFASBRAN is compared with our implementation. As our implementation is 3x3 mesh, we configured it to work in the 2x2 mesh network to compare with the work in [9]. Figure 13. Shows the comparison results. It can be noted that our implementation consumes far lesser resources.

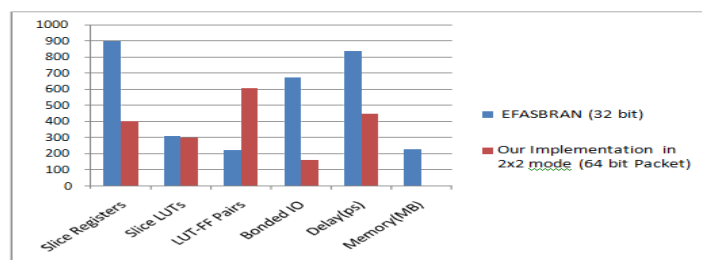


Figure 13. Comparison of our work with EFASBRAN [9]

It must be noted that we used a 64 bit packet where as a lot of compared work uses only 32 bit packets. The operation frequency is not published in [9], however we have documented that our design in a similar configuration works at 735.132 MHz. From simulation results given above the communication between processing elements will take about 6 cycles in our implementation and in [9] it is published that it takes about 13 cycles. We conclude that this decrease in number of cycles in our implementation is because of absence of the external router logic which can induce more latency.

VI Conclusion:

In this work, we have presented a technique to create an “agent” based framework for NOC implementation. This is an attempt to get rid of external routers and allow the data communication to happen from an external layer of the processing elements. The systems route the packets with awareness of the state of the neighboring elements. It can also be noted that this framework is suitable for heterogeneous implementation of NOCs as well. The data integrity issues are handled by framing data along with appropriate headers. FIFOs are used to handle data rate mismatch issues. We have designed the PEs to be self contained so that they can be individually controlled. This helps in scalability and when not required they can be turned off so that power can be saved. This will also allow scale-up and scale-down of the system with ease. This is because the strategy of NOC should be made available for all families of FPGAs small or big. Multi-port top level packet injection allows quick transfer of packets into the networked processing elements.

Future work should involve testing the network for different loads and studying the performance of the design for various applications. Fault tolerance capability can also be included in the design. This should give way to an improvised scalable networking mechanism on chip. Apart from these, we also foresee through intuitive analysis that several other issues should be addressed such as scale-up modality, a deeper study on the impact of the state of neighboring elements on routing, timing closure for FPGA and ASIC implementation, deadlock recovery, load balancing, energy factor, etc.

VII References:

- [1] A Comparison of Network-on-Chip and Busses, http://www.artemis.com/noc_whitepaper.pdf, 2005.
- [2] Joshu Holden et al., “A simplified AES algorithm”, 2010
- [3] Geng Luo-Feng, Du Gao-ming, et al., “Design and Performance Evaluation of a 2D-MeshNetwork on Chip Prototype Using FPGA”, IEEE 2008
- [4] Kavyashree G S, et al., “Architectural based congestion management for Network-on-Chip implemented on FPGA”, IEEE 2016
- [5] Graham Schelle and Dirk Grunwald, “Exploring FPGA Network on Chip Implementations Across Various Application and Network Loads”, IEEE 2008
- [6] Ayas Kanta Swain, et. al., “FPGA Prototyping and Parameterized based Resource Evaluation and Network on Chip Architecture”, IEEE 2016

- [7] Yu Cai et al., “Comparative Evaluation of FPGA and ASIC Implementations of Bufferless and Buffered Routing Algorithms for On-Chip Networks”, IEEE 2015
- [8] Kumar et al., “MACS: A Highly Customizable Low-latency Communication Architecture”, IEEE 2016
- [9] E Lakshmi Prasad et al., “EFASBRAN: Error Free Adaptive Shared Buffer Router Architecture for Network on Chip”, ELSEVIER Science Direct 2016
- [10] Luca P. Carloni et al., “Networks-on-Chip in Emerging Interconnect Paradigms: Advantages and Challenges”, IEEE 2009
- [11] Maria Elisabete Marques Duarte, “Networks on Chip (NOC): Design Challenges”, ICCA 2003
- [12] Graham Schelle and Dirk Grunwald, “Exploring FPGA Network on Chip Implementations Across Various Application and Network Loads”, IEEE 2008
- [13] Xinchao Gu et al., “Research on congestion perception and control of network on chip”, IEEE 2016
- [14] Junxiu Liu, et al., “Fault-Tolerant Networks-on-Chip Routing With Coarse and Fine-Grained Look-Ahead”, IEEE 2016
- [15] Yu-Yin Chen et al., “Path-Diversity-Aware Fault-Tolerant Routing Algorithm for Network-on-Chip Systems”, IEEE 2017
- [16] Mayank Kumar, “FPGA Based Design of Area Efficient Router Architecture for Network on Chip (NoC)”, IEEE 2016
- [17] Yohta Kurokawa et al., “A Prototype Router Circuit for Extended 2D Mesh Network on Chips”, IEEE 2016
- [18] Zi-Xu Wu et al., “Exploration of A Reconfigurable 2D Mesh Network-on-Chip Architecture and A Topology Reconfiguration Algorithm”, IEEE 2012
- [19] F.F Zakaria, et al., “Cooperative Virtual Channel Router for Adaptive Hardwired FPGA Network-on-Chip”, IEEE 2016
- [20] Aline Vieira de Mello, et al., “Evaluation of Routing Algorithms on Mesh Based NoCs”, Faculdade de Informática – PUCRS, 2004

