DESIGN OF AN EFFICIENT ROUTER FOR NETWORK ON CHIP DESIGN

by

CHRISTOFORAKIS IOANNIS

BSc. Technological Institute of Crete, 2012

A THESIS

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF APPLIED INFORMATICS
AND MULTIMEDIA

SCHOOL OF APPLIED TECHNOLOGY

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

2014

Approved by:

Supervisor
Kornaros Georgios

Copyright

CHRISTOFORAKIS IOANNIS

Abstract

The increased demand for on-chip communication bandwidth as a result of the multi-core trend has made packet switched networks-on-chip (NoCs) a more compelling choice for the communication backbone in next-generation systems [1]. However, NoC designs have much power, area, and performance trade-offs in topology, buffer sizes, routing algorithms and flow control mechanisms—hence the study of new NoC designs can be very time-intensive. In this thesis, we explore the design space of FPGA-based NoC router. A packet switched NoC router is implemented on Xilinx FPGA devices using parameterized VHDL models. To reduce the area and increase the speed, novel techniques are employed such as utilizing both edges of the clock. The buffer size of Virtual Channels (VCs) can be configured in terms of number of flits depending on the designer's needs. The router employs minimal number of control fields in the packet while header and data can be sent together in one flit. Credit based flow control is used to accelerate the packet transfers. The proposed router design is evaluated based on area, frequency, and latency. Implementation results show that it is comparable and even superior to widely referenced, previously proposed on-chip routers.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude and appreciation to my supervisor Dr. George Kornaros for his guidance and support, which helped me complete this work. He is the one who introduced me to this fantastic field that was very suitable to my interests and encouraged my creativeness. The courses i took with him, working with him as a graduate assistant throughout the period of my Master's program at the TEI Of Crete, as well as his thesis advice, helped me develop the skills needed to conduct my research. My appreciation also goes to the respected Effie for her help. I would like to thank my friends and colleagues for their company and interesting discussions, especially Ohonas, Dimitrios. Lastly, my thanks to my family and Kallia.

Chapter 1 - Introduction

Constantly shrinking transistor dimensions enable ever-increasing density on modern microchips: each new technology node facilitates additional cores in chip multiprocessors. A system on chip is an integrated circuit that integrates all components of a computer or other electronics system into a single chip. The major challenge the designers of these systems must overcome is to provide functionally correct and reliable operations of the interacting components. On-chip physical interconnections will present a limiting factor for performance and energy consumption [1][52]. Initially the chip designing was aimed at producing chips that contained a single standalone design. As the number of transistors that is fabricated onto a single chip increased, the concept of system on chip was made possible. In a system on chip (SoC), multiple standalone designs, referred to as cores or Intellectual Property cores are stitched together on a chip to provide a functional system. NoC is an approach to design the communication subsystem between intellectual property cores in a system on chip. The communication strategy in system on chip uses dedicated buses between communicating resources. This will not give any flexibility since the needs of the communication, in each case, have to be thought of every time a design is made. Another possibility is the use of common buses, which have the problem that it does not scale very well, as the number of resources grows. NoC is intended to solve the shortcomings of these, by implementing a communication network of switches and resources [2, 3].

Network-on-chip routers communicate via a common inter-connect, connecting processor cores, memory controllers, and so on. At each node (usually a core or memory), a network interface controller connects the core to the local router, and converts messages from the core into data packets of varying size for the network. These packets are further divided into flits, the smallest unit of data traveling in the network, which dictates the width of a link connecting two routers. Routers then direct traffic within the network, moving flits from source to destination according to the information encoded in each packet, usually located in the header (first flit) of the packet. In particular, in wormhole routing [36], a single packet’s flits may be spread across

multiple routers as they traverse the network, until all the constituent flits are collected at the destination. Compared to bus-based systems, network-on-chip designs have the advantage of allowing many messages in flight simultaneously thus providing efficient communication among many nodes [53].

The NoC based system on chips imposes various design issues on the fabrication of such integrated chips. Firstly, the suitable topology for the target NoCs such that the performance requirements and design constraints are satisfied. Secondly, the design of network interfaces to access the on chip network and routers to provide the physical interconnection mechanisms to transport data between processing cores[5]. Thirdly, the selection of communication protocols which are suitable for on chip interconnection networks. Finally, as technology scales and switching speed increases, future network on chips will become more sensitive and prone to errors and faults. Fault tolerance is becoming critical for on-chip communications [4].

Today's SoCs need a network on chip IP interconnect fabric to reduce wire routing congestion, to ease timing closure, for higher operating frequencies and to change IP easily[55]. Network on chips are a critical technology that will enable the success of future system on chips for embedded applications. This technology of network on chips is expected to dominate computing platforms in the near future.

1.2 Thesis Contribution

The main contribution of this thesis is to design a configurable Router for Network on Chips capable of providing a high frequency with as low a cost design as possible. As it is demonstrated in the following sections, the suggested architecture is added to the existing, as architecture with a simple but reliable functionality. It is also an architecture that can contribute in the designing of bigger networks, as it does not require a big area in contrast to other architectures. An emphasis is given, in this architecture presented, in the existence of Virtual Channels and their width. That is because; we give the designer the option of a big or small VC width and simultaneously the increased or decreased requirement in area on chip.

Another important contribution our architecture has to offer is a solution to the designing of reconfigurable NoCs. A Router that will have a design flexibility so as to be implemented

with different kinds of Flow Control and different Routing techniques. Generally, it can be used as a block in many FPGA devices without redesign. All of the widths and libraries are generic, written in VHDL and do not need any translation by those that may want to use it.

1.3Thesis Organization

The general approach of this thesis is to explore the design space of NoC routers, with more focus on FPGA-based NoCs, through the design, enhancements, and evaluation. The thesis is organized as follows: Chapter 2 introduces a background on NoC and FPGA aspects. Chapter 3 as well as a review of related research work. In Chapter 4, the designed routers architectures are described and functionally verified along with the techniques used to augment the performance and reduce the area. The evaluation methodology and attained results are presented in Chapter 5 including the comparison with some previous work results. Finally, Chapter 6 concludes all the contributions made in this thesis, and outlines future research directions.

Chapter 2 - Theoretical Background

SoCs consist of various IP blocks, routers, and physical links. IP block (in some literature terms such as processing element (PE) or virtual component are also used) is a general term used for representing such entities as DSPs, memory modules, MPEG decoders, processor cores and so on. Routers and links are employed for providing communication infrastructure for IP blocks in other words they organize a NoC. Routers have input and output ports which are connected to IP blocks and other routers in the system. The number of ports depends on the network topology. Physical links are actually copper wires utilized for data transmission between adjacent routers and IP blocks. The messages sent via physical links are divided into packets or flits depending on the used switching technique. Packet is a fixed length data block containing all the control and routing information, which gives the router an opportunity to decide where the incoming packet should be sent, in its header. Sometimes packets are further split into flits (flow control units). Unlike packets just the first flit called header flit involves routing information; its main task is reserving a path for other flits (data flits) following it, the last flit is a tail flit which releases all the resources reserved by header flit. Performance of NoC as defined in the previous chapter, depends on various factors such as network topology, routing strategy and switching technique. Hence, in this chapter, we give a brief theoretical background on these issues. Firstly, we introduce NoC topology and most popular topologies employed in NoCs are reviewed. Subsequently, switching techniques used in NoCs are presented. In the next section we describe widely used routing algorithms and review their performance characteristics briefly. Finally, we present brief information about traffic models for NoCs.

2.1 Topology

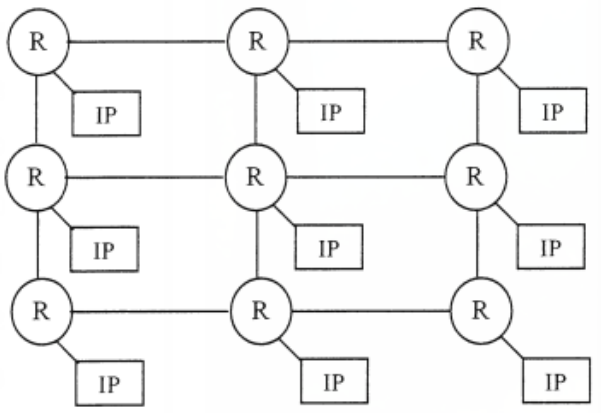
From the communication perspective, there have been various topologies for NOC architecture. These include mesh, torus, ring, butterfly, octagon and irregular interconnection networks [6]. Various researchers have exploited these different NOC topologies for their NOC implementations. Kim et al. have used a star-based NOC that communicated using the principle of CDMA (Code Division Multiple Access) [7]; Adriahantenaina et al. proposed a tree-based

implementation of NOC [8], where each node in the tree behaves as a router in NOC; Pande et al. compared various network topologies for interconnection networks in terms of latency, throughput, and energy dissipation [9]. Several researchers have suggested that a 2-D mesh architecture for NOC will be moreefficient in terms of latency, power consumption and ease of implementation, as compared to other topologies. The Octagon NOC demonstrated in [10] is an example of a novel regular NOC topology.

2.1.1 Mesh

This architecture is based on m x n mesh network where every router, except those at the edges, is connected to four neighbouring routers and one compution resource (IP) through communication channels [12]. This topology allows integration of large number of IP cores in a regular-shape structure. A channel consists of two unidirectional links between two routers or between a router and a resource. **Figure** shows a 3x3 mesh NoC with nine functional IP blocks.

Figure 2.1 A 3x3 mesh topology



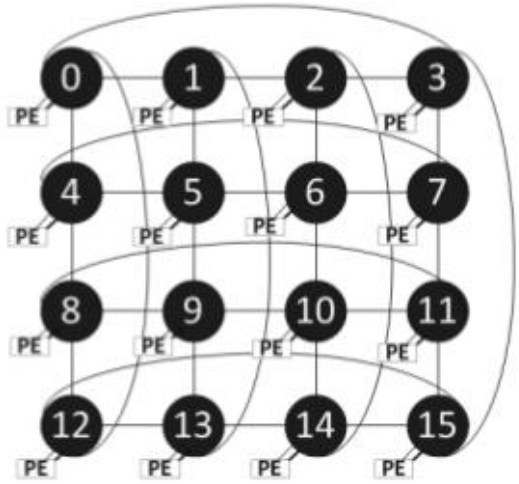
2.1.2 Torus

Torus [11] architecture is the same as regular mesh. However, unlike the mesh, where edge switches are connected only to two neighboring switches, the torus architecture uses wrap-around channels in order to connect the switches at the edges to the switches at the opposite

edge. The number of switches is equal to the number of IP blocks and every switch has five ports. Due to the long wrap-around channels the packet transmission delay may become significantly long and require usage of repeaters. This can be avoided by folding the torus as it is shown in Figure 2.1. Folding is done by shifting all nodes in even rows to the right and all nodes in even positions of each row down, next connecting all the neighboring nodes in newly gained rows and columns then pair-wise connecting edge nodes in rows and columns. Now wraparound links are significantly shorter and link propagation delays fit within a single clock cycle.

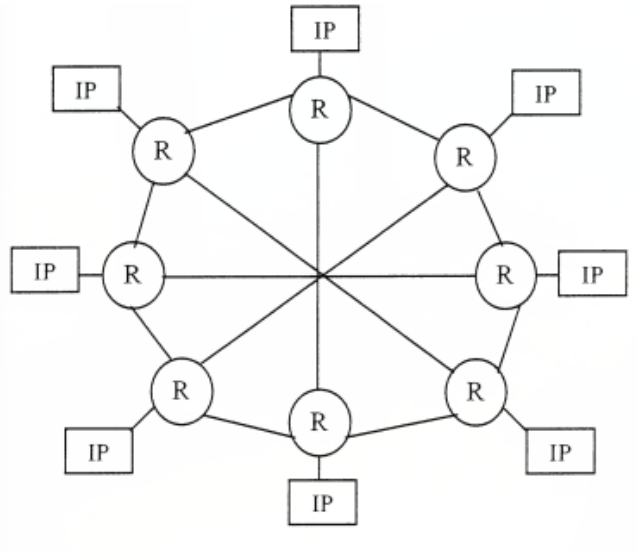
Figure 2.2 2D Torus with 16 nodes 1

2.1.3 Octagon



The Octagon [10] architecture has its own advantageous properties. Every pair of nodes has a maximum two-hop path to communicate with each other. The basic model consists of eight IP blocks and 12 bidirectional links, as shown in Figure 2.2. The nodes are arranged in a ring and there is a central connection point in the center of a ring. Every node is also connected to the neighboring nodes. The node consists of IP block and a switch. Every switch has three connection ports. Usually the architecture uses a simple short-path algorithm. The throughput is higher than of the shared bus and crossbar interconnect if properly designed. This requires a development of good interconnection scheduler.

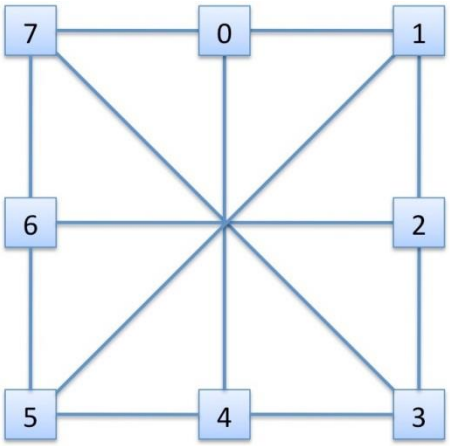
Figure 2.3 Octagon Topology



2.1.4 Spidergon

The Spidergon [12], architecture represented in **Figure 1** is a proposal of ST Microelectronics for the System on Chip domain. It is composed of an even number of nodes (N) and it is similar to a ring enriched by across links between opposite nodes. Some of the most interesting characteristics of the Spidergon's topology are: i) network with regular topology, ii) vertex symmetry (same topology appears from any node), iii) edge transitivity, iv) constant node degree (equal to 3) translating in simple router hardware and efficiency. High node degree reduces the average path length but increases complexity. Proposed algorithms for this architecture are the so Across First (aFirst), and Across Last (aLast).

Figure 2.4 Spidergon Topology



2.1.6 Other NOCs

There have been a sizeable number of proposals/implementations of NOCs in the literature. Without being exhaustive, these include:

A. A mesh-based NOC using Chip-Level Integration of Communication Heterogeneous Elements (CLICHÉ) [14];

B. Proteo, a flexible-topology NOC [15];

C. A guaranteed-throughput switch for a circuit-switched NOC, which supports both unicast and multicast [16];

D. HiNoc, a NOC offering both a low-overhead transmission service and one with guaranteed QoS, with a two-level asynchronous/synchronous hierarchy [17];

E. A reconfigurable circuit-switched NOC [18];

F. NOCs for the Princeton Smart Camera SoC [19];

G. A CMOS implementation of a NOC interconnecting multiple processing units of different clock frequencies [20];

H. MANGO (Message-passing Asynchronous Network-on-Chip providing Guaranteed services through OCPinterfaces) [21];

I. A connectionless NOC implementing Diff serv to provide QoS [22].

J. SoC BUS (Switched Network on Chip for Hard real time embedded systems) aims at providing the guaranteed throughput [21];
L. A SPIN interconnect template [13] (Scalable, Programmable, Integrated Network) uses fat-tree architecture [6].

2.1.5 Deadlock and Livelock

A deadlock occurs when a group of packets are unable to move in a network because they are waiting for each other to release resources in a cyclic fashion. Deadlock can be avoided by analyzing and breaking cyclic dependencies in the dependency graph of the shared network resources by applying routing restrictions or using additional hardware resources such as virtual channels [23]. Livelock occurs when packets continue to move through the network, but they do not make progress toward their destinations [6]. In adaptive non-minimal routing, livelock can occur because of its flexible capability of redirecting packets. One of the ways to avoid livelock in a network is to add a misroute count, which holds the number of times a packet has been misrouted [6]. Once the count reaches a threshold, no more misrouting is allowed.

2.2 Switching Techniques

Switching techniques can be classified based on network characteristics. Circuit switched networks reserve a physical path before transmitting the data packets, while packet switched networks transmit the packets without reserving the entire path. Switching techniques can be classified based on network characteristics. Circuit switched networks reserve a physical path before transmitting the data packets, while packet switched networks transmit the packets without reserving the entire path. Packet switched networks can further be classified as Wormhole, Store and Forward (S&F), and Virtual Cut Through Switching (VCT) networks (see Figure 2).

2.2.1 Store-and-forward

Store-and-forward. Store-and-forward routing is a packet switched protocol in which the node stores the complete packet and forwards it based on the information within its header. Thus the packet may stall if the router in the forwarding path does not have sufficient buffer space. The CLICHE [14] is an example of a store-and-forward NoC.

2.2.2 Wormhole

Wormhole routing [24] combines packet switching with the data streaming quality of circuit switching to attain minimal packet latency. The node looks at the header of the packet to determine its next hop and immediately forwards it. The subsequent flits are forwarded as they arrive. This causes the packet to worm its way through the network, possibly spanning a number of nodes, hence the name. The latency within the router is not that of the whole packet. A stalling packet, however, has the unpleasantly expensive side effect of occupying all the links that the worm spans. In Al-Tawil et al. [1997], a well-structured survey of wormhole routing techniques is provided, and a comparison between a number of schemes is made.

2.2.3 Virtual cut-through

Virtual cut-through routing [11] has a forwarding mechanism similar to that of wormhole routing. But before forwarding the first flit of the packet, the node waits for a guarantee that the next node in the path will accept the entire packet. Thus if the packet stalls, it aggregates in the current node without blocking any links.

Table 2.1 Cost and Stalling for Different Routing Protocols, as [31] proposed

Protocol	Per router cost		Stalling
	Latency	Buffering	
store-and-forward	packet	packet	at two nodes and the link between them
wormhole	header	header	at all nodes and links spanned by the packet
virtual cut-through	header	packet	at the local node

2.3 Flow Control

Flow control determines how network resources, such as channel bandwidth, buffer capacity, and control state, are allocated to a packet traversing the network. The flow control may be buffered or bufferless. The Bufferless Flow Control has more latency and fewer throughputs than thr Buffered Flow Control.

2.3.1 Buffered Flow Control

The Buffered Flow Control can be further categorized into Credit Based Flow Control, ACK/NACK Flow Control, STALL/GO Flow Control, T-Error Flow Control, and Handshaking Signal based Flow Control.

2.3.2 Credit Based Flow Control

In Credit Based Flow Control, an upstream node keeps count of data transfers, and thus the available free slots are termed as credits. Once the transmitted data packet is either consumed or further transmitted, a credit is sent back, used Credit Based Flow Control (QNOC[25],BCB[54]).

2.3.3 Handshaking Signal Based Flow Control

In Handshaking Signal Based Flow Control, a VALID signal is sent whenever a sender transmits any flit. The receiver acknowledges by asserting a VALID signal after consuming the data flit. used handshaking signals in their SoCIN NOC implementation [26].

2.3.4 ACK/NACK

In the ACK/NACK protocol a copy of a data flit is kept in a buffer until an ACK signal is received. On assertion of ACK, the flit is deleted from the buffer; instead if a NACK signal is asserted then the flit is scheduled for retransmission, this flow control technique used in XPIPES implementation [27].

2.3.5 STALL/GO

In the STALL/GO scheme, two wires are used for flow control between each pair of sender (producer) and receiver (consumer). When there is an empty buffer space, a GO signal is activated. Upon the unavailability of buffer space, a STALL signal is activated. None of the present NOC implementations have employed this flow control scheme.

2.3.6 T-Error Flow Control

The T-Error Flow Control scheme is very complex as compared to other flow control mechanisms. It aims at enhancing the performance at the cost of reliability. Real time systems operating in a noisy environment must avoid the use of this flow control mechanism. None of the present NOC implementations has employed this flow control scheme.

2.4 Virtual Channels

Wormhole switching has been proposed to reduce the buffer requirements and enhance the system throughput. However, each packet may occupy several intermediate switches and links at the same time. This may introduce the problem of deadlocks [28]. To avoid this problem, virtual channels are used. Virtual channel flow control exploits an array of parallel buffers at each input port. In this technique, there are multiple queues per physical channel and as a result if one packet is blocked other packets which belong to other virtual channels can use the idle

bandwidth. This improves the throughput of the network and reduces the average packet latency by allowing blocked packets to be bypassed.

In general, virtual channels offer many advantages such as avoiding deadlocks by breaking cycles in the resource dependency graph [29], optimizing wire utilization by letting several logical channels share the physical wires, improving performance by minimizing the stall frequency, and providing quality-of-service (QoS) by assigning different priorities to different data flows [30].

2.5 NOC Issues and Challenges

To enhance system productivity, it is very important that an architect be able to abstract, represent and address most of the design issues and concerns at a high level of abstraction. System-level design affords one the opportunity to review several different software-hardware architectures that meet the functional specifications equally well, to quickly trade-off among different QoS metrics such as latency, power, cost, size and ease of integration. Similarly, there are several issues related to NOC, such as the nature of the NOC link, link length, serial vs parallel links, bus vs packet-based switching, and leakage currents. In this section, we discuss these issues.

2.5.1 Serial vs Parallel Link

The transportation of data packets among various cores in a NOC can be performed by the use of either a serial or a parallel link. Parallel links make use of a buffer-based architecture and can be operated at a relatively lower clock rate in order to reduce power dissipation. However, these parallel links will incur high silicon cost due to inter-wire spacing, shielding and repeaters. This can be minimized up to a certain limit by employing multiple metal layers. On the other hand, serial links allow savings in wire area, reduction in signal interference and noise, and further eliminate the need for having buffers. However, serial links would need serializer and de-serializer circuits to convert the data into the right format to be transported over the link and back to the cores. Serial links offer the advantages of a simpler layout and simpler timing verification.

Serial links sometimes suffer from ISI (Inter-symbol Interference) between successive signals while operating at high clock rates. Nevertheless, such drawbacks can be addressed by encoding and with asynchronous communication protocols.

2.5.2 Interconnect Optimization

Communication in a NOC is based on modules connected via a network of routers with links between the routers that comprise of long interconnects. Thus it is very important to optimize interconnects in order to achieve the required system performance. Timing optimization of global wires is typically performed by repeater insertion. Repeaters result in a significant increase in cost, area, and power consumption. Recent studies indicate that in the near future, inverters operating as repeaters will use a large portion of chip resources. Thus, there is a need for optimizing power on the NOC. Techniques for reducing dynamic power consumption include approaches discussed in [32], [33]. Encoding is another effective way of reducing dynamic power consumption [34]. In order to make NOC architectures more effective, innovative ways will have to be introduced to minimize the power consumed by the on-chip repeaters.

2.5.3 Leakage Power Consumption

The leakage current, which was negligible relative to the dynamic switching current at larger transistor sizes (of 1 micron or more), is expected to dominate the current drain at sub-100 nm technologies. In a NOC, the link utilization rates vary and in many cases are very low, reaching a few percentage points. Networks are designed to operate at low link utilization in order to meet worst case scenario requirements, and thus having a higher link capacity helps reduce packet collisions. However, even when NOC links are idle they still will consume power in repeaters, due to the dominance of this leakage current at small feature sizes. Thus, new techniques will have to evolve which will help reduce the leakage power consumption to make the NOC architecture more effect

2.6 On-Chip Routers

In a switched network there is a need to find a route through several switches. Therefore the switches that are cross-points in the network also implement a routing function. They are called routers because of this functionality. The routing algorithm is a very important part of the router since its task is to route every packet towards the right direction. Some routing algorithms are able to tell which route is the fastest, not only, which way that is the shortest. The two main kinds of routing algorithms are static and adaptive routing. Static routing is when there are one, or possibly a few paths between sender and receiver that are fixed. In static routing algorithm, the routing changes very slowly, if at all, over time. When the routing is changed it is often a result of human intervention.

Adaptive, also called dynamic routing, on the other hand is when the routing algorithm alters the route of packets in a dynamic way. A dynamic routing algorithm changes the routes according to, for example, network traffic or due to changes of the topology.

A global routing algorithm has complete information about connectivity and link costs in the network. The algorithm can thereby compute the least-cost path between source and receiver. The calculation itself can be run at one site or at multiple sites.

Decentralized routing algorithms calculate the least-cost path by communicating with the neighbors. In the beginning the node only knows the costs of its own directly attached links, then through an iterative process of communication between nodes, the least cost path to a destination is calculated.

The router operation revolves around two fundamental regimes: (a) the datapath and (b) the associated control logic. The datapath consist of number of input and output channels to facilitated packet switching and traversal. Generally 5 input X 5 output router is used. Out of five ports four ports are in cardinal direction (North, South, East, Waste) and one port is attached to its local processing element.

2.6.1 Routing algorithms

Routing algorithms define the path taken by a packet between source and target switches.

They must prevent deadlock, livelock, and starvation [35][36] situations. Deadlock may be defined as a cyclic dependency among nodes requiring access to a set of resources, so that no forward progress can be made, no matter what sequence of events happens [6]. Livelock refers to packets circulating the network without ever making any progress towards their destination. Starvation happens when a packet in a buffer requests an output channel, being blocked because the output channel is always allocated to another packet.

Routing algorithms can be classified according to the three different criteria: (i) where the according routing decisions are taken; (ii) how a path is defined, and (iii) the path length. According to where routing decisions are taken, it is possible to classify the routing in source and distributed routing. In source routing, the whole path is decided at the source switch, while in distributed routing each switch receives a packet and defines the direction to send it. In source routing, the header of the packet has to carry all the routing information, increasing the packet size [36]. In distributed routing, the path can be chosen as a function of the network instantaneous traffic conditions. Distributed routing can also take into account faulty paths, resulting in fault tolerant algorithms. Depending how a path is defined, routing can be classified as deterministic or adaptive. In deterministic routing, the path is completely specified from the relative position of source and target addresses. In adaptive routing, the path is a function of the network instantaneous traffic. Adaptive routing increases the number of possible paths usable by a packet to arrive to its destination. However, deadlock and livelock situations can happen in fully adaptive algorithms [35], which limit its usage. Regarding the path length criterion, routing can be minimal or non-minimal [36]. Minimal routing algorithms guarantee shortest paths between source and target addresses. In non-minimal routing, the packet can follow any available path between source and target. Non minimal routing offers great flexibility in terms of possible paths, but can lead to livelock situations and increase the latency to deliver the packet.

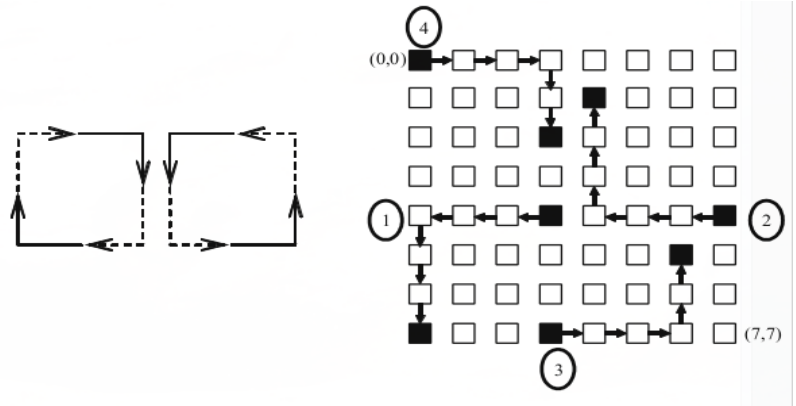
Glass and Ni proposed wormhole routing algorithms for mesh-connected networks, which are deadlock and livelock free [10]. These were proposed in minimal and non-minimal partially adaptive versions. When passing between switches in a 2D mesh, a packet can follow four directions: East, West, North, and South. Eight distinct turns are possible in the path followed by a packet, as shown in Fig. 1(a). Algorithms with no restrictions are named fully adaptive,

otherwise they are named partially adaptive. Fully adaptive routing algorithms are subject to deadlock conditions. As demonstrated in [10], if at least two turns are forbidden it is possible to implement deadlock free algorithms. This a sufficient condition for achieving freedom of deadlock.

2.6.1.1 XY Algorithm

This Section presents four routing algorithms, one deterministic (XY) and three partially adaptive - West first, North-last and Negative-first. Only minimal [10] algorithms are considered in this work, to avoid increased latency to deliver packets and livelock situations. Source and target coordinates are identified in the following discussion by the use of the notations (XS, YS) and (XT, YT), respectively. The XY algorithm is deterministic. Flits are first routed in the X direction, until reaching the YT coordinate, and afterwards in the Y direction, as shown in Fig. 2(b). If some network hop is in use by another packet, the flit remains blocked in the switch until the path is released. As illustrated in Figure 2.5(a) from [37] turns where the packet comes from the Y direction are forbidden (dotted lines).

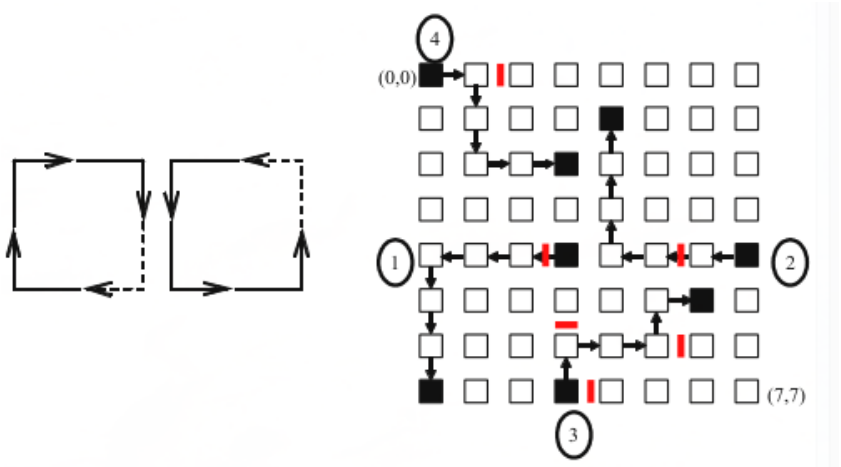
Figure 2.5 Continues lines represent allowed turns and XY routing algorithm like presented in [37]



2.6.1.2 West-First Algorithm

In the West-First algorithm if XT ≤ XS, packets are routed deterministically, as in the XY algorithm, (Fig. 3, paths 1 and 2). If XT > XS packets can be routed adaptively in East, North or South directions Figure 2.6 The total time to deliver an individual packet can be reduced using adaptive algorithms, since the packet can, in some situations, make turns to escape from blocking conditions.

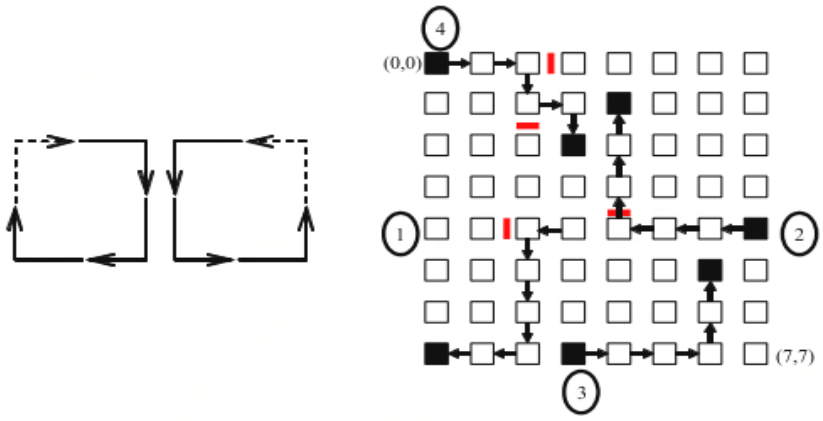
Figure 2.6 West-first routing algorithm. The prohibited turns are the two to the West [37].



2.6.1.3 North-Last Algorithm

In the North-Last algorithm if YT ≤ YS packets are routed deterministically (Fig. 2.7, paths 2 and 3). If YT > YS packets can be routed adaptively in West, East, or South directions (Fig. 4, paths 1 and 4).

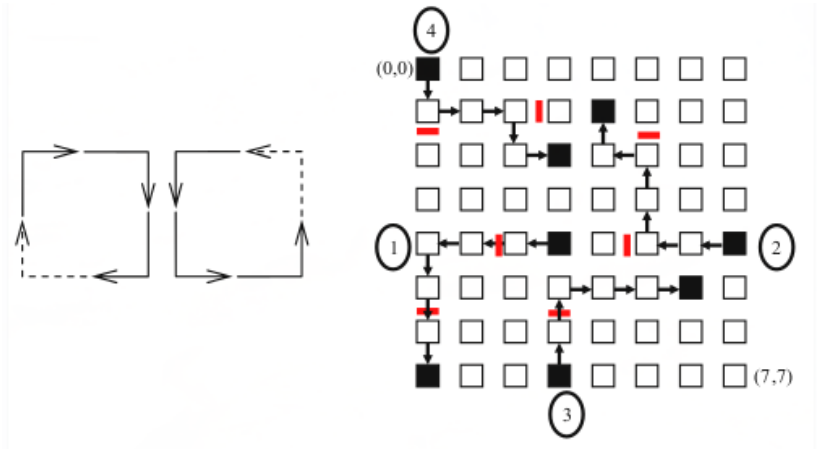
Figure 2.7 North-Last routing algorithm. The prohibited turns are the two when traveling North.[37]



2.6.1.4 Negative-First Algorithm

In the Negative-First algorithm packets are routed first in negative directions, i.e., to the North or to the West directions1. If (XT≤XS and YT≥YS) or (XT≥XS and YT≤YS) packets are deterministically routed, as illustrated in Figure 2.7 path 1 (source address (3,4) and target address(0,7)) and path 3 (source address (3,7) and target address (6,5)). All other conditions allow some form of adaptive routing, as illustrated in paths 4 and 2.

Figure 2.7 Negative-First routing algorithm. The prohibited turns are the two from a positive direction to a negative direction [37]



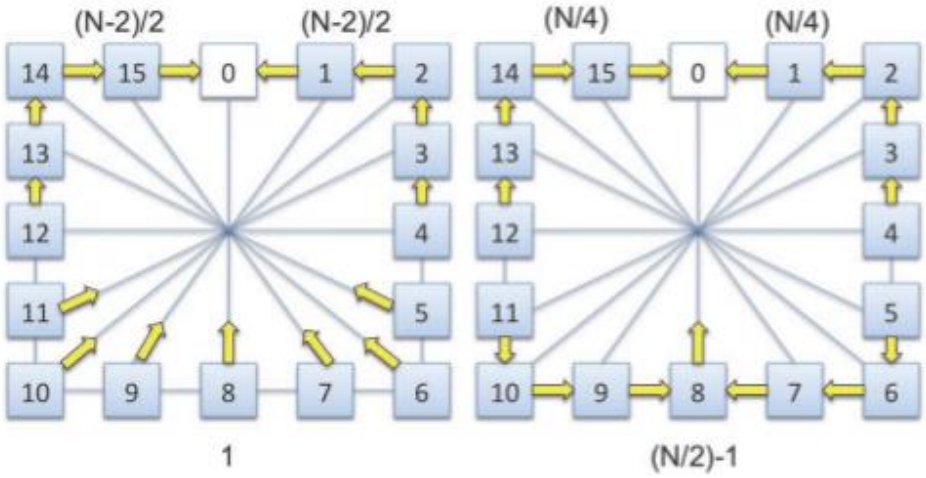
2.6.1.5 Spidergon routing algorithms

To the best of our knowledge the proposed algorithms for this architecture are the so called Across First (aFirst) and Across Last (aLast) algorithms [10]. Both algorithms are minimal source routing whose behavior is depicted in Figure 2.8. In the aFirst algorithm the across channel can be used only as a first step. Figure 2.8 shows a Spidergon NoC with node 0 as hot-spot. The arrows starting from each node indicate the path that packets would follow if they were generated by that node. If the destination node is on the source's half of the ring composing the Spidergon NoC, then the packet is forwarded along the clockwise (CW) or counter clockwise (CCW) direction accordingly to the minimal path restriction. If the packet's destination is on the opposite side of the ring the packet is first forwarded along the across channel and from there, along the ring towards its final destination.

In aLast algorithm depicted in Figure 2.8 a packet can traverse an across link only as last step. When a destination is on the source's opposite half of the network the packet is forwarded towards the node connected to the destination through the across link and from there finally delivered. Otherwise, as in aFirst the packet is forwarded along the ring channels in the clockwise or counter clockwise directions.

38]

Figure 2.8 Node's routing behavior towards node 0 using aFirst and aLast [38]



Chapter 3 - Related Work

There are many similar works that created new router architectures. Most of them intend to create architecture that offers greater reliability, scalability and adaptivity. Also the design of on-chip routers based on optimizing power consumption and chip area. Further the architectures of on-chip routers aimed to give results in which power consumption is reduced and silicon area is also minimized. Another key objective is the low latency and High-Throughput of these. In this section we review some of these architectures.

Adaptive routing: The quest for high performance and energy efficient NoC architectures has been the focus of many researchers. Fine-tuning a system into maximizing system performance and minimizing energy consumption includes multiple trade-offs that have to be explored. As with all digital systems, energy consumption and system performance tend to be contradictory forces in the design space of on-chip networks. In [39] proposed a typical state-of-the-art, wormohole-switched virtual channel flow control router. Authors proposed router can support deadlock-free fully adaptive routing for 2-D mesh and torus networks using hardware logic. The architecture uses a novel path selection scheme resulting in a 2-stage switching operation with a decomposed crossbar switch. They show that under non-uniform traffic our architecture reduces the overall network latency by a significant factor. They showed that the energy consumption when using adaptive routing is also reduced. Results also show a better performance in the presence of nearest-neighbor traffic, a particular benefit for NoC designs which emphasize spatial locality. The other way of adaptivity in routers is dynamic configuration like [40] which proposed a novel Reliability Aware Virtual channel (RAVC) NoC router micro-architecture that enables both dynamic virtual channel allocations and the rational sharing among the buffers of different input channels. In this work authors claim that router provides 7.1% and 3.1 % average latency decrease under uniform and transpose traffic pattern respectively. In [41] the main goal is with the use of different clocks in a head flit and body flits, to overcome performance degradation due to the routing decision time. The proposed mechanism is implemented for the adaptive wormhole router and the performance evaluation was completed using simulation. Theirs simulation results demonstrated the performance enhancements in terms

of maximum accepted traffic and the average latency in the range of accepted traffic and the average latency in the range of accepted traffic. Whole packet forwarding [42] is a novel VC re-allocation scheme for fully adaptive routing algorithms in wormhole networks. This scheme allows multiple packets to reside in one VC concurrently; it greatly improves VC utilization does not lead to. They further propose a novel fully adaptive routing algorithm that exploits WPF and provides routing flexibility with modest hardware overhead. Compared with conservative VC re-allocation, WPF improves the saturation throughput by 88.9% on average in synthetic traffic patterns and achieves 21.3% average full-system speedup. In 3D NoCs deterministic algorithms are usually used but DyXYZ algorithm [43] is coming to make a different because this is fully adaptive routing algorithm and route packets adaptively in the network. APSRA [44] adaptive algorithm is a new methodology for adaptive routing. In this work, authors presented results based on analysis and simulation-based evaluation demonstrate that routing algorithms developed using our approach significantly outperform general-purpose routing algorithms like XY and Odd-Even for a mesh topology NoC. They also show that routing algorithms generated by the APSRA methodology has even higher performance and adapt ity advantage over other deadlock-free routing algorithm for nonhomogeneous mesh NoCs.

Deterministic routing: Deterministic router architecture implement in [45]. The one to one unicast router architecture making use of virtual cut through switching mode was designed and the outputs where verified using simulation results. The desired wormhole switching mode multicast router architecture is the base of this architecture. Another way is to combine advantages of both deterministic and adaptive routing algorithms, like BIOS [46]. Authors proposed novel routing algorithm which improves both input and output selection. In fact, BIOS is based on the best input and output selection. Simulation results with different traffic patterns show that their routing algorithm achieves significant better performance than the other deterministic and adaptive routing algorithms.

Chapter 4 - NoC Router Design and Architecture

The router architecture is separated into two parts. One part is the router as a key component of the NoC. Router implemented with the following subcomponents: two demultiplexers, two buffers (virtual channels), one fifo block, one scheduler, and one LUT(rom based). The precise function of the router and its subcomponents explained in detail to this chapter.

4.1 Router Architecture

The router shown in figure 4.3 has 5 ports, where the fifth port is used to connect the router to its IP core (PE). The router forms the heart of the NoC backbone. It is responsible for transporting packets generated by IP cores. It is comprised of input buffers, LUT/ROM based, routing unit as shown in figure 4.3. There is one more external port used by processor to programming the Routing Table.

An arriving flit is searched in the RAM Table, and it is found then stored at the input buffer which is divided into virtual channels to prevent deadlock and increase throughput. LUT decodes the routing information in the received header flit in order to find the requested output port. Virtual channels provide multiple input buffers per physical input channel so that packets with different destination output ports can pass blocked packets in a router. Then the DEMUX 1to2 unit decides according LUT output, in which VC can proceed toward the output por. Once the output port number and VC buffer are allocated, the flit traverses the DEMUX 2to5 unit, towards the round robin scheduler.

4.1.1 Design Improvement Techniques

In order to refine the area and the speed of our router, some strategies during the design are applied. Although some of these techniques are mentioned previously, the most effective and innovative ideas are summarized here:

- Minimizing the number of control fields in the packet as well as eliminating the tail flit, while replacing them with some logic and counters in the routing unit to infer the start and

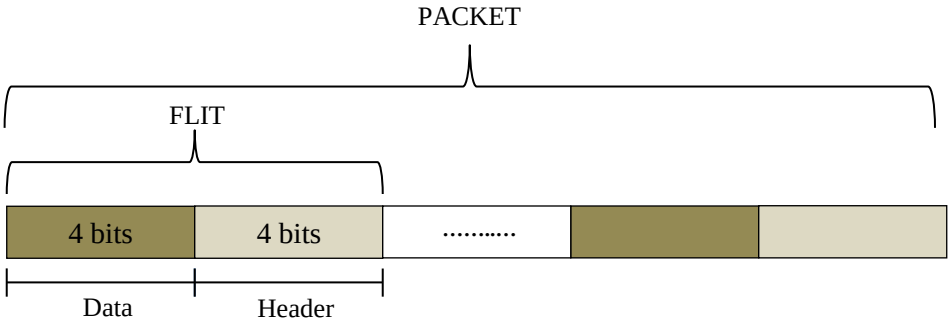
end of the packets, and consequently, reduce the FIFO size. Header and data can be sent together in one flit.

- Using the credit based flow control, so we are able to transmit the flit in only one clock cycle, instead of using the hand shaking protocol that needs at least two clock cycles.
- Utilize the rising edge of the clock to accomplish and synchronize some operations while employing the falling edge to undertake other operations. This technique enables us to squeeze a number of dependable operations in only one clock cycle either in the FIFO Finite State Machine (FSM) Read/Write operations, or arbitration and routing handling.

4.1.2 Packet Format

The Packet Format can be structured with different techniques depending on the needs in data, and the kind of information that needs to be transported through the Router. Regarding our structure, in the trial we present below, we have a one byte flit. The four least significant bits are the header of the flit and the four most significant are the data Figure 4.1. The remaining 8 bits of the packet may be the Source and the Target Address. In the classic version of the wormhole, it has to keep the state. In our version the routing continues regardless of whether the previous flit was lost or not. In addition, in a case of fault, the package is not lost because the following flits have their own header in order to be routed independently.

Figure 4.1 An example of a control packet structure for the proposed Router, assuming 8-bit flit



4.1.3 LUT-RAM table

The packet that is received from the router is 1 byte. The four less significant bits is the packet header and the remaining four most significant are the data payload.

When a packet moves from one port to another port of the router, the data come along with the header. Based on this header will be searching into the Look Up Table (LUT). Depending on the location in ROM, should find the packet header will fit the number of the output port and the number of VC which passed to the "Demux1 to 2". The routing table programming is done by an additional external port of Router.

The LUT includes 16x13 RAM, there are five port router, so router includes five LUT one for any port. For example, assume that packet "01010010" arrives to port 1. The Local enable port signal is asserted and the LUT search the "0010" header in RAM. If the header is found in the RAM, the renable signal asserted and the packet continues to "1to2 Demux" input. VC and Output Port number disclosed, from RAM Table, Figure 4.3. Santicipation of watching the header found in rom, in the case that header not included in the ROM Table, output signal will set at zeros. Three cycles are required until this function complete.

4.1.4 DEMUX 1to2

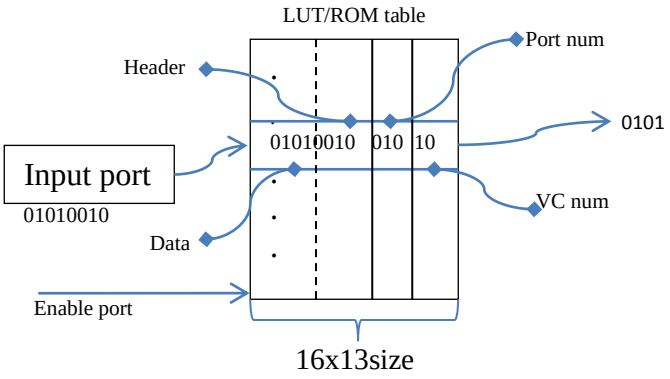
Demux 1to2 received as input the data from the output of the corresponding LUT, under the VC number, which also receives as input from LUT writes data to the corresponding buffer before enable port signal should be asserted. Input and output data of Demux is only the four most significant bits of packet that router port receives at beginning. In this example VC number is "10", i.e. Virtual Channel number two (VC 2)

4.1.5 Virtual Channels

Virtual Channel is FIFO buffers with registers which store up to 8 packets of 4bits, this can write a packet per clock cycle. It is written in memory of the VC and 2to5 Demux reads, after read enable signal triggered. From the moment that first demux take a flit, in this example "0101"

until prompted by the second demux required additional six clock cycles, squandered primarily for buffers reading.

Figure 4.2 LUT searching new input packet and VC/Output port number, in ROM TABLE



4.1.7 FIFO Memory

Fifo's memory are located before the five schedulers for each output port. When the DEMUX complete this operation will send flit at fifo in particular position corresponding to the output port of each flit. In our example, the output port is the "010" namely door 2, so the fifo will be read only by the scheduler that serves the door number 2. Additional four cycles required up to this point of path.

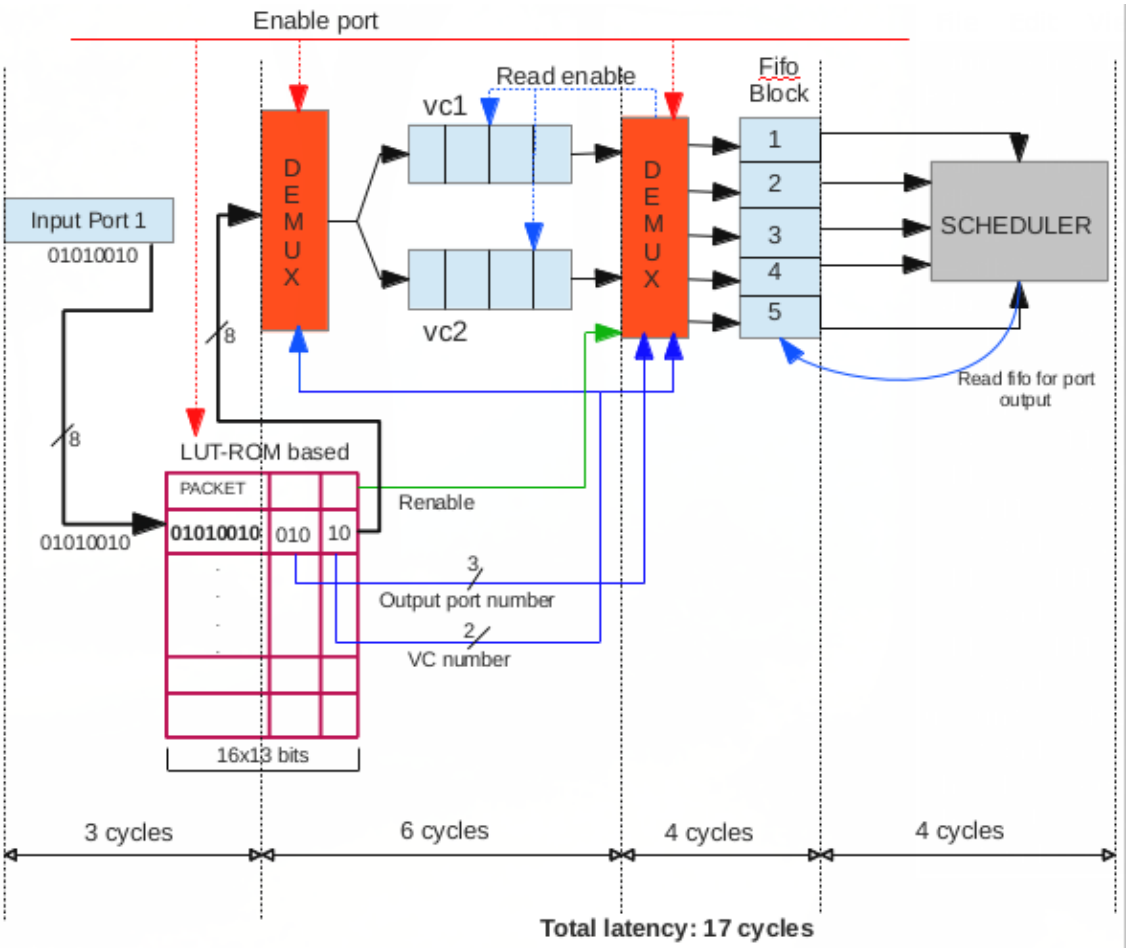
4.1.8 Scheduler

The logic design of an efficient and fast round robin arbiter in VHDL language relies on the capability to find the next requestor to grant without losing cycles and with minimal logical stages. Using the fastest logic constructs like Parallel Prefix Computation (PPC) and the most suitable architecture will result in a fast and efficient arbiter suitable for pipeline integration of a multiple queue structure. First we need to start with a clear definition of what we actually want the arbiter to do. The round robin arbiter design requirement can be summed up in a list:

One cycle calculation, so the arbiter can grant different requestors in each cycle. Wraparound functionality, meaning that the arbiter does not loose cycles at the end of each round when moving from a grant to the last active requestor, back to the first one. Work conserving

functionality, so no cycles are lost on requestors that are inactive. Our scheduler arbitrary all data/flits where in the corresponding position in each fifo memory, as we describe before, for this example, at the position 2 of the first port.

Figure 4.3 Router Design: Each unit with partial and total latency of a flit.



Chapter 5 - Evaluation and Results

This chapter presents the synthesis and operational results for implementation of the standalone configurable router as well as complete. NoC-based multiprocessor systems. The router created based on the implementation details described in the preceding chapters. Multiple configurations of the standalone router have been implemented in programmable logic and also synthesized for standard-cell VLSI using standard-cell approach in order to characterize the resource utilization and performance. Operational results are presented to demonstrate functionality and correctness of the router design and the NoC systems. The measurements and the results were done on three different Xilinx devices, so as compare our architecture with other published architectures. The devices we used to count the resource utilization are the Virtex-II, Spartan 3e, Virtex 6 and Virtex 7.

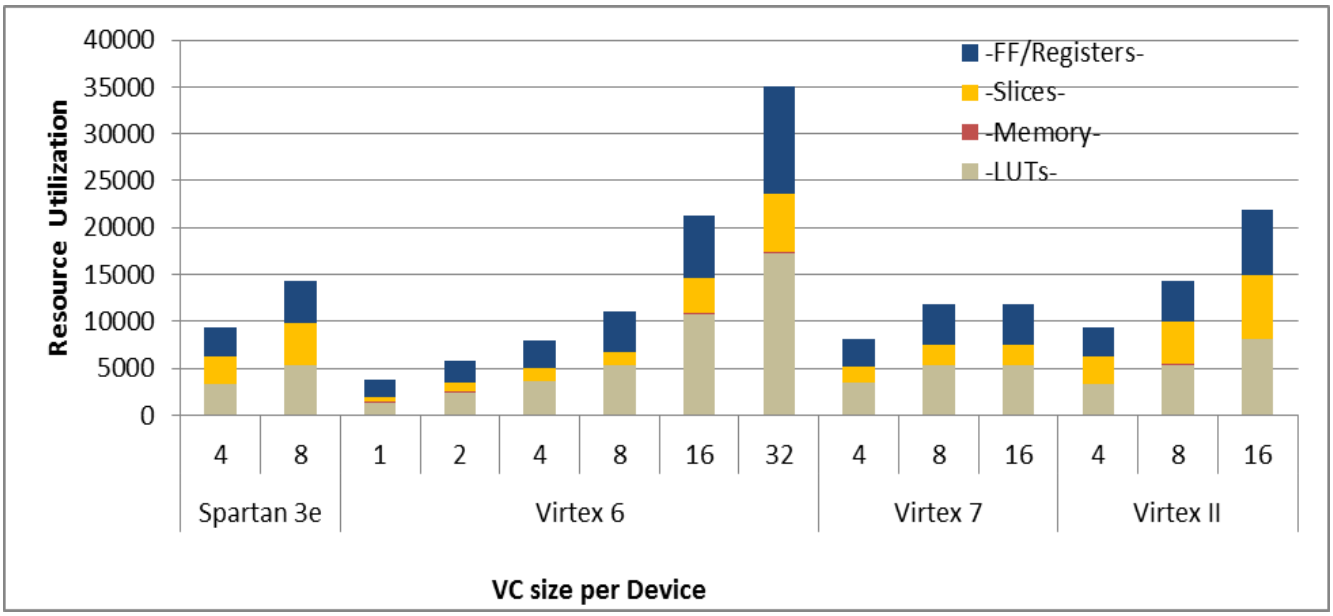
5.1 Configurable Router Implementation

This section presents the synthesis results for different implementations of the router in Xilinx FPGA devices in order to characterize the resource utilization and performance. With the configurable router intended for embedded NoC support within an FPGA, results are also provided from synthesis for standard-cell implementation in a 0.18-μm fabrication process.

The implementation follows the description provided in Chapter 4. The basic implementation is designed by using two Virtual Channels, with a buffer size width of 8, each of 1 byte. A similar implementation can be done by changing the buffer size to 1, 2,4,16 and 32 width, each of 1 byte, in characterizing resource utilization and performance. The relationship between resource utilization and varying virtual channel width will also be analyzed, followed by the effect from the addition of virtual channels on the router design.

The FPGA implementations of the standalone configurable router have been compiled

with Xilinx ISE 14.7 for an Xilinx Virtex6 XC6VCX75T device. All five bidirectional ports of the overall router and the routing functions for the supported network topologies are included in each standalone router implementation in order to reflect a complete router. For experimental purposes, VC buffers are implemented as FIFOs with 5 different depths. Figure 5.1 illustrates the



FPGA resource utilization breakdown in each one of the devices we performed the implementation on and for each different Virtual Channel size.

Figure 5.1 FPGA resource utilization, four devices resource divide by buffer size

Buffer storage is implemented using dedicated memory and the buffer control logic is fixed at a relatively small amount of 38 LEs per buffer, thus the logic resource utilization for buffering can be considered as being constant and independent of interconnect width. It is clear that, the bigger the buffer size of the VCs, the bigger the area occupied in the Routers total. Indicatively for the Virtex 6, when the buffer size offers only one memory slot, the Router occupies 603 slices. In the case of two memory slots it increases to 1014 and may reach 6228 Slices for 32 memory slots leading to the observation of a total increase of over 90%. Small fluctuations are observed per device. As we will later on see, compared to other architectures, there are important differences depending on the type of architecture. For example, the DART

Router [47] occupies 15% of the available slices of the Virtex 6, which is 5652, whereas the Kavaldjiev Virtual Channel Router occupies a maximum of 1325 of the Virtex-II. In the corresponding devices our architecture occupies 603-6228 Slices depending on the buffer size (1-32 slots) and 2839-6912 of a buffer size from 4-16 slots Table 5.1 . Another fact that occurs is that in devices of a newer technology, our architecture occupies a small area percentage of the FPGA. It is characteristic that on the Virtex 6 , even in the case of a buffer size of 32, the total area occupied reaches 16% of the available area

Table 5.1 Synthesis Results of Router on four Xilinx Devices, for any case of VC size

DEVICE	VC Size	FF/Registers	LUTs	Memory	Slices	Frequency(MHz)
Virtex6	1	1780	1360	30	603	439.049
	2	2305	2457	35	1014	440.47
	4	3040	3572	35	1342	440.47
	8	4360	5328	35	1343	446.275
	16	6750	10821	35	3733	437.031
	32	11476	17301	35	6228	442.288
Virtex II	4	3155	3318	35	2839	139.043
	8	4450	5400	35	4503	139.026
	16	6865	8050	35	6912	108.12
Spartan 3E	4	3150	3318	35	2873	161.655
	8	4450	5330	35	4468	159.872
Virtex 7	4	3040	3491	35	1588	548.878
	8	4335	5287	35	2222	548.878
	16	4335	5287	35	2222	548.878

The custom implementations of the standalone router have been generated with Questa Sim 10.2c_5 of ModelTech. Conservative operating conditions have been used to constrain the design to ensure correctness; hence there is opportunity for performance optimization in future work. Buffers are presently implemented with flip-flops for experimental convenience, but results are reported with emphasis on the logic for the router. Generation of custom memory modules is a future refinement of the configurable router towards embedded implementation.
Table 5.1 illustrates the measurements to do with the FPGA resource utilization for all the devices with a custom implementation. Specifically we measure the Flip-Flops, the Look Up

Tables, the memories and the Slices of an FPGA that are used by the total of available for the four different devices of FPGA, as we mentioned in the previous paragraph. The FIGURE, that is an outcome based on the Table 5.1, illustrates the Slices Utilization for custom implementation on Virtex 6. It becomes clear in this graph, that the Slices needed increase, as the buffer size increases. It is therefore a matter for consideration to choose whether we want a big buffer size, in order to have a respectively low drop rate, and to lose few flits or whether we want our architecture to occupy as little area as possible in the FPGA.

Figure 5.2 Slices Utilization for custom Implementation on four Devices

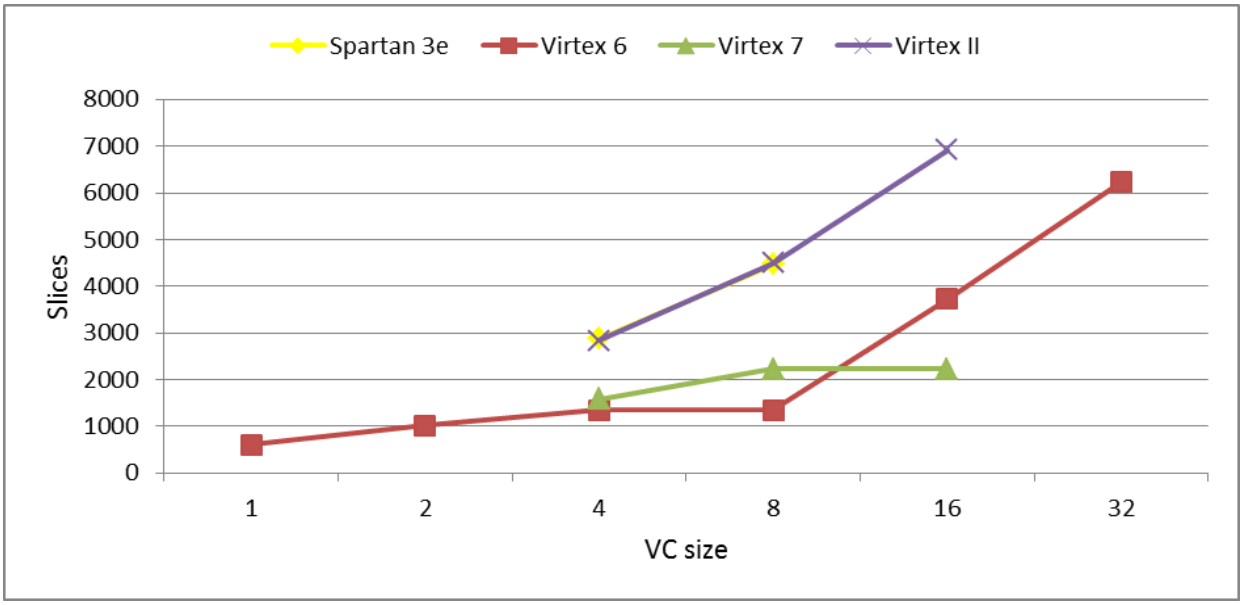
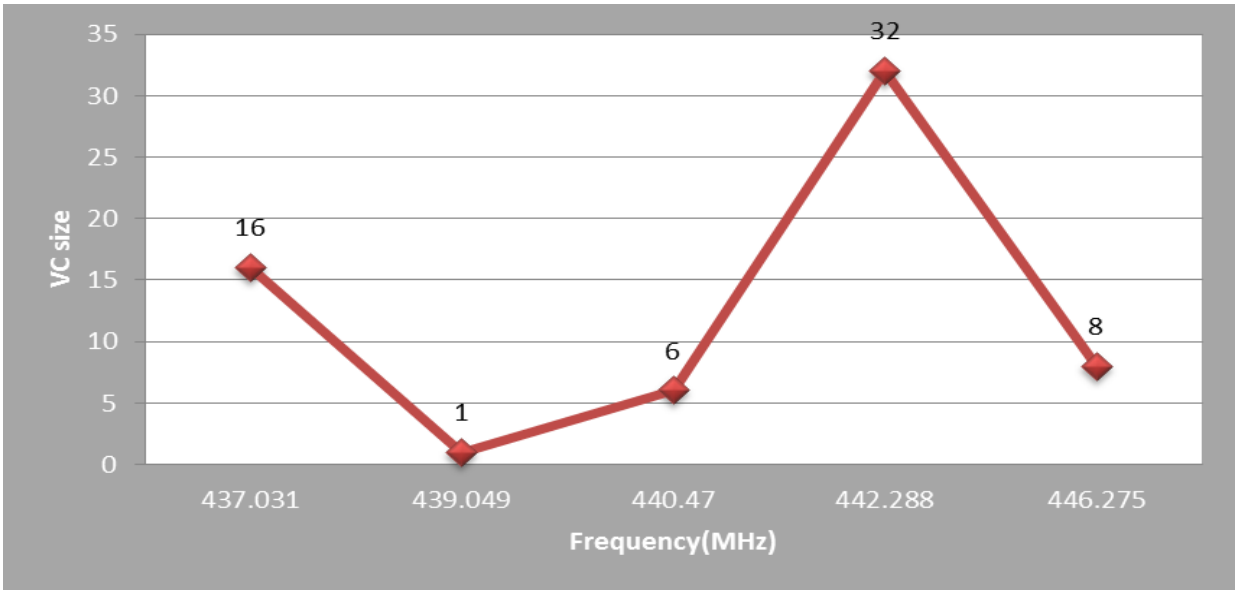


Figure 6.3 shows us the basic advantage of our architecture, the high frequencies that can be achieved by using any buffer size. For the Virtex 6 device, at 439 MHz we will find the buffer size 1 and at 436 MHz the buffer size is 2. There are different fluctuations in the frequency, but as anyone can notice they are not relevant to the proportionally or not increase of the buffer size. As we will later on see our architecture, in all the devices that were tested, offers higher frequencies than other known architectures that were tested on the same devices. A fact that, if combined with a low binding ability in areas on the device, can give a great advantage to the

NoC designers, especially those who need big and complex networks, as they can achieve high time frequencies with a low cost design as possible.

Figure 5.3 Frequency of Router on Virtex 6, per VC size



5.1.1Configurable Router Operational Results

Operational results are presented in this section for the standalone router implementation. QuestaSim is used to perform functional simulation for verification at the router level and to provide operational results. The majority of the router testing is performed by inspecting the individual output channel requests and their resulting packet routing and handshaking signals.

The Figure 5.4 presents the transportation of a flit from the input port “datai3” to the “datao1” output port, with a 17 cycle latency. This is the basic function of a Router regardless of the architecture and the options of input and output ports. We present the successful function of the Router in three different scenarios, in extreme cases that give us the chance to understand the correct function even in the most extreme cases of traffic. More so we present the throughput of the input and output ports in each case, as well as the total throughput of the router in the scenarios below.

5.1.1.1 Scenario 1: Data arrives from the same Input Port and end up from the same Output Port

Firstly, in this scenario where all the data (flits) arrive from the same input port and the same output port, we observe an increase in the latency by a cycle of each next flit. This is because the round robin scheduler needs this cycle in order to read the flit that came along with the previous flit but was impossible to assist together at that moment. Therefore, we can see that the third in line flit needs 17 cycles to find the ouput port. In the Figure 5.5 though we see the appearance of the drop rate, because when the injection rate reaches and exceeds 50%, the output ports throughput decreases by 40% . In addition, we observe, in the Figure 5.6 that the routers maximum throughput in this case can't exceed 31,25%. The drop rate is defined as the percentage of the packets that are lost or cannot be assisted by the Routers input port. Accordingly, the injection rate is defined by the amount of packets/flits that enter the Router (flit/cycle/IP) per cycle and port.

Figure 5.4 1st Scenario Simulation: D1 flit from port1, exit to port1. Port Number fended in Routing Table

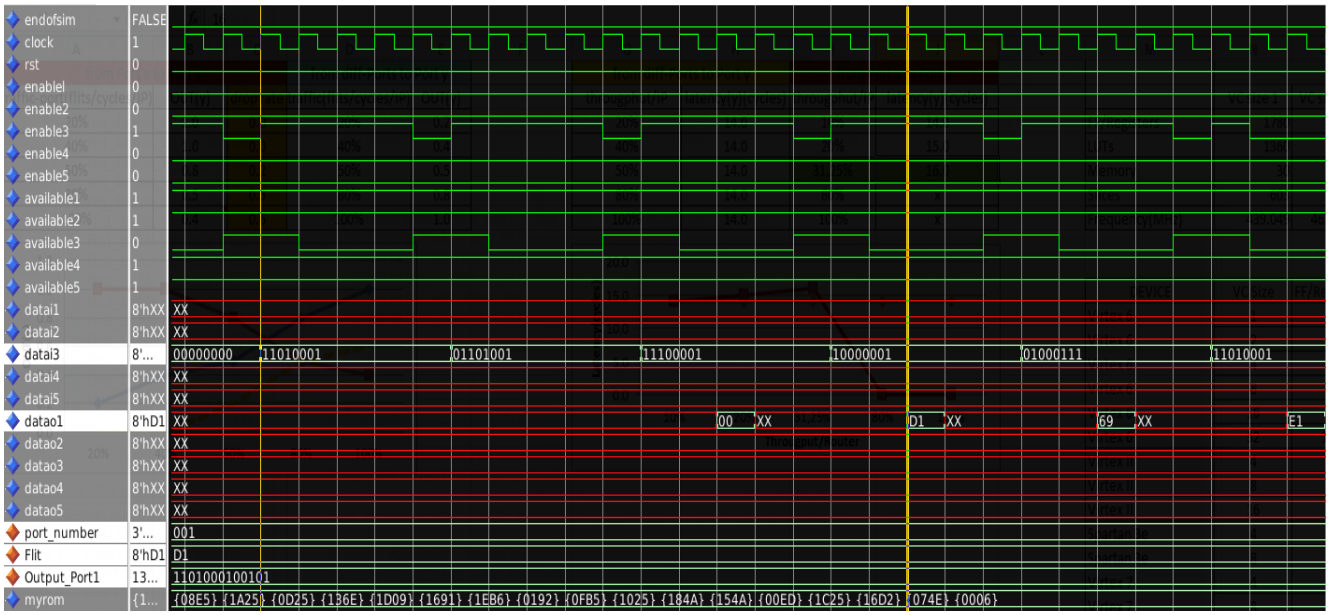


Figure 5.5 Drop and Injection rate measurement: Red line for Scenario 1, blue for Scenario 2 and orange line is Drop Rate

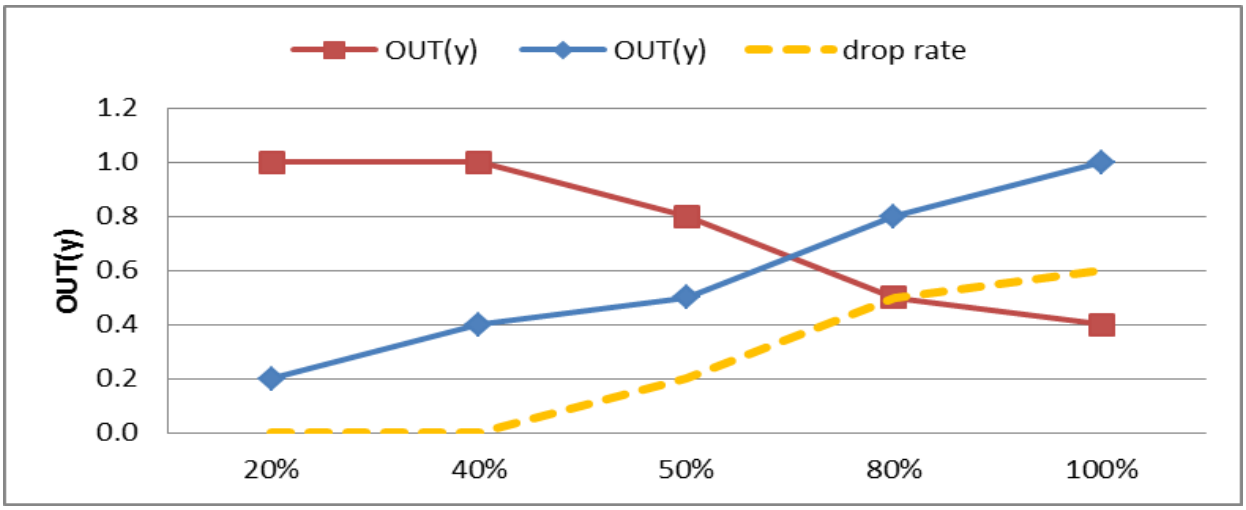
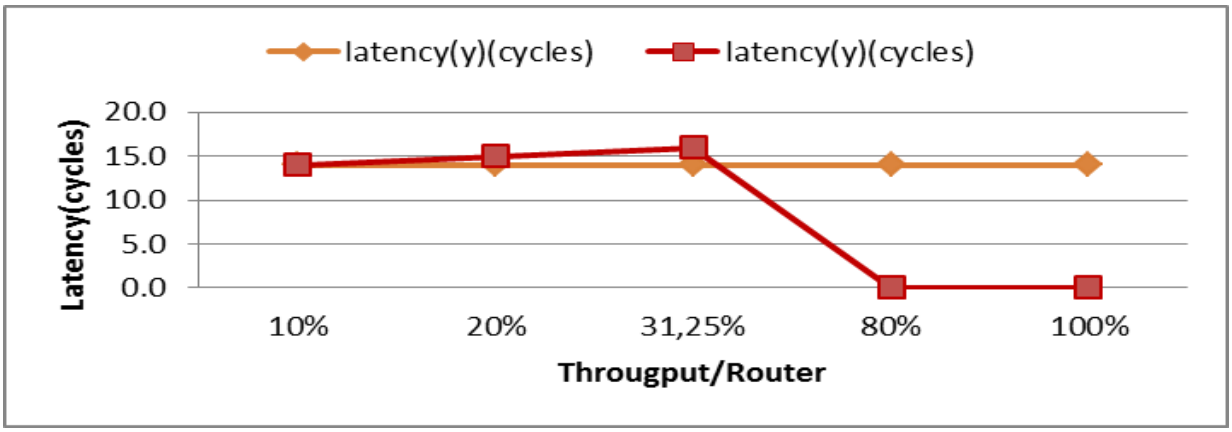


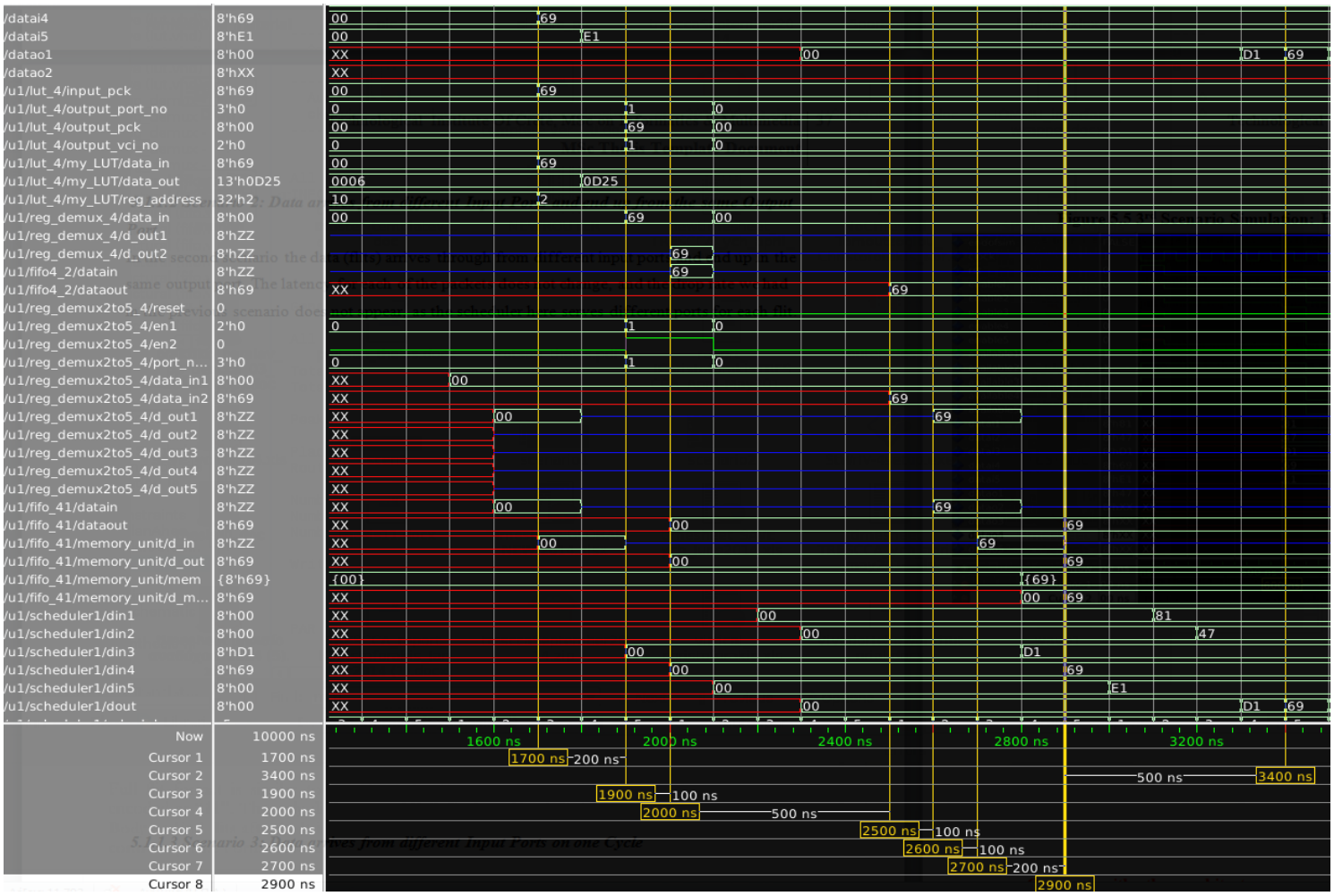
Figure 5.6 Latency measurement for two presented scenarios: Red line for Scenario 1 and orange for Scenario 2.



5.1.1.2 Scenario 2: Data arrives from different Input Ports and end up from the same Output Port

In the second scenario the data (flits) arrives through from different input ports and end up in the same output port. The latency for each of the packets does not change, and the drop rate we had in the previous scenario does not appear, as the scheduler here serves different ports for each flit, as shown on Figure 5.7. Specifically, about the package 0x69 that arrives at the datai4 input port. In two clock cycles the package has passed the Routing Tables check. The header of the package is in the RAM , therefore it enters the demux1to2 input port. After a cycle it exits the demultiplexer and enters the Virtual Channel that has been assigned to the header from the Routing Table. In this example, it arrives at the second Virtual Channel of the Routers fourth input port (fifo4_2). After five cycles, the read enable is received, the flit passes the entrances of the second demultiplexer, demux2to5. In an additional clock cycle it exits the demultiplexer at the exit that is corresponding to the output port the flit has to reach, namely the datao1 for this specific example. After another cycle, it is stored in the output ports fifo, namely fifo_41 in our example, and after two cycles it passes the round robin scheduler , that will in at least five cycles accommodate the flit.

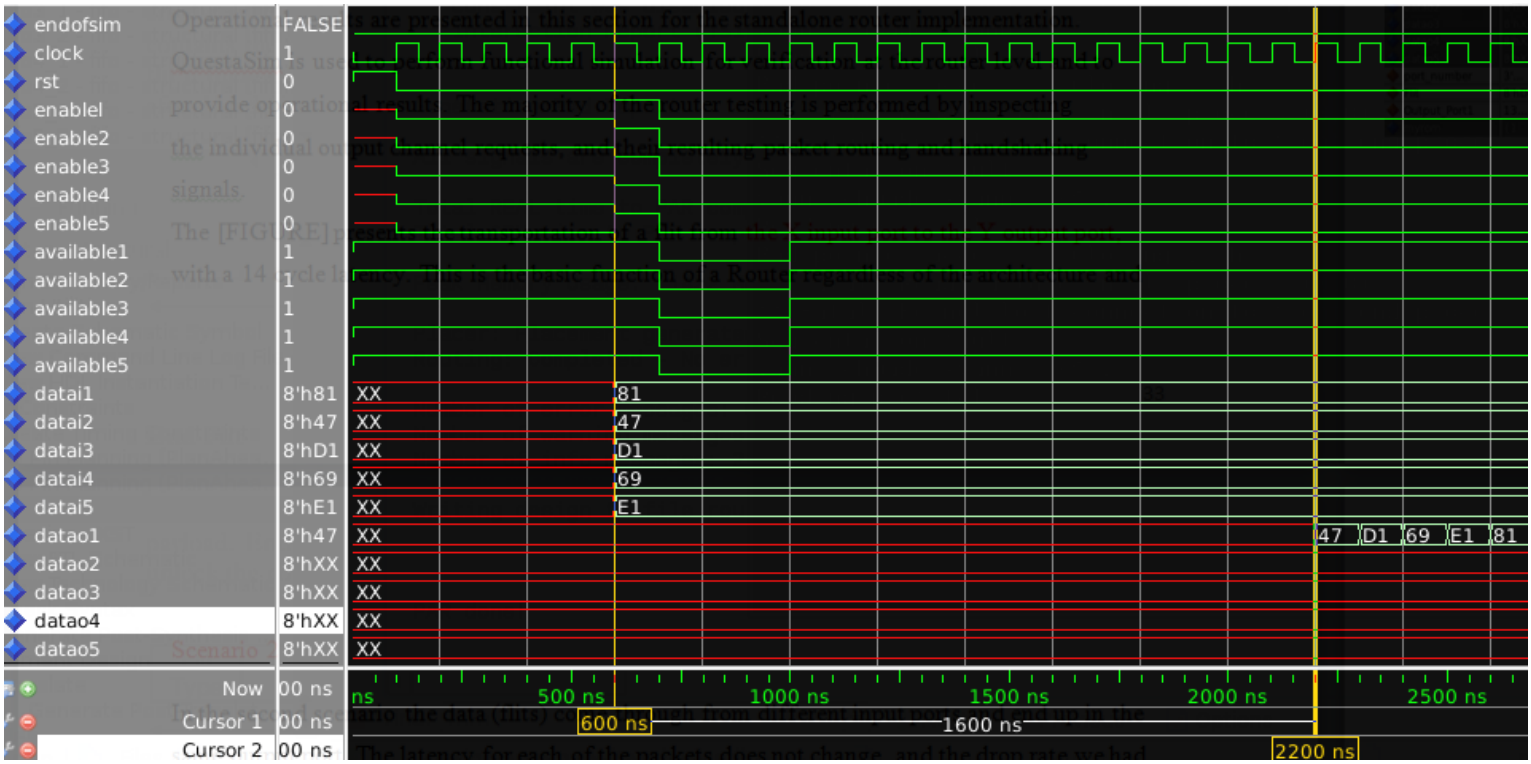
Figure 5.7 Second Scenario Simulation: Flits arrive from different ports to port "datao1". Flit 0x69 on this graph



5.1.1.3 Scenario 3: Data arrives from different Input Ports on one Cycle

In this scenario the flits arrive from different ports on the same clock cycle, ending up in the same output port, latency increased by a cycle for every flit. As it appears in figure 5.8, there is no indication of a router malfunction

Figure 5.8 Third Scenario Simulation: Flits arrive from different ports, at the same cycle to port "datao1"



5.2 Comparison with other Architectures

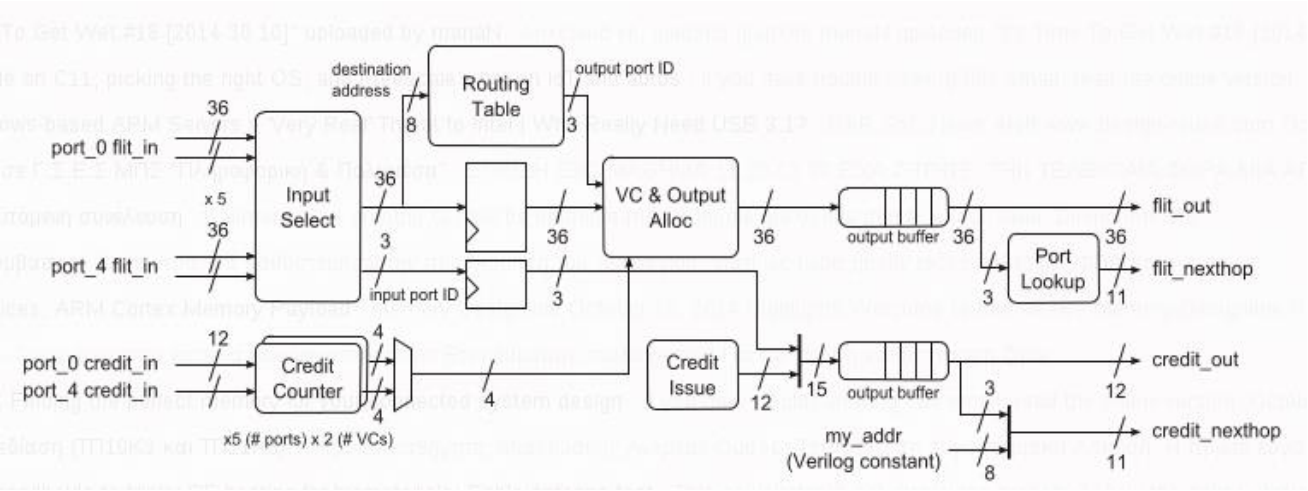
In this section we compare the resource utilization of our architecture to other know Router architectures. We also compare the maximum frequency our Router can reach to other models. Our comparisons are made to four of the most common Router models for NoCs. Firstly, we compare our architecture to the Router used by the DART simulator for the manufacture of a NoC[49]. A second architecture we compare to our own, is the Five Port Router presented by Swapna in 2008[50], an architecture which inspired us the simplicity of our model. Again in 2008, an equally groundbreaking architecture was presented by Everton Carara, Ney Calazans and Fernando Moraes, the High-Performance Router [51], whose frequency and resource utilization we compare the Xilinx Virtex-II device. Finally, we compare our architecture to MANGO, one of the most know architectures for Router architectures for NoCs, by Tobias

Bjerregaard and Jens Sparsø [52]. Table 5.1 presents the area results and the maximum frequency for each different buffer size. The comparisons below are based on these results.

5.2.2 Compare with DART

DART NoCs use the classic wormhole VC router, which is composed of per-VC flit buffers, routing logic, VC and switch allocators and a crossbar. Since the FQs model the flit buffers, the Router component only encapsulates the routing and allocation logic. Figure 5.9 shows the Router datapath. The number of ports is set to five in our current implementation, but can be changed by setting a HDL parameter. We use table-based routing. Hence any deterministic routing algorithm can be implemented. The table contents are configurable without reprogramming the FPGA. The configuration of the routing table also facilitates the simulation of a wide range of topologies.

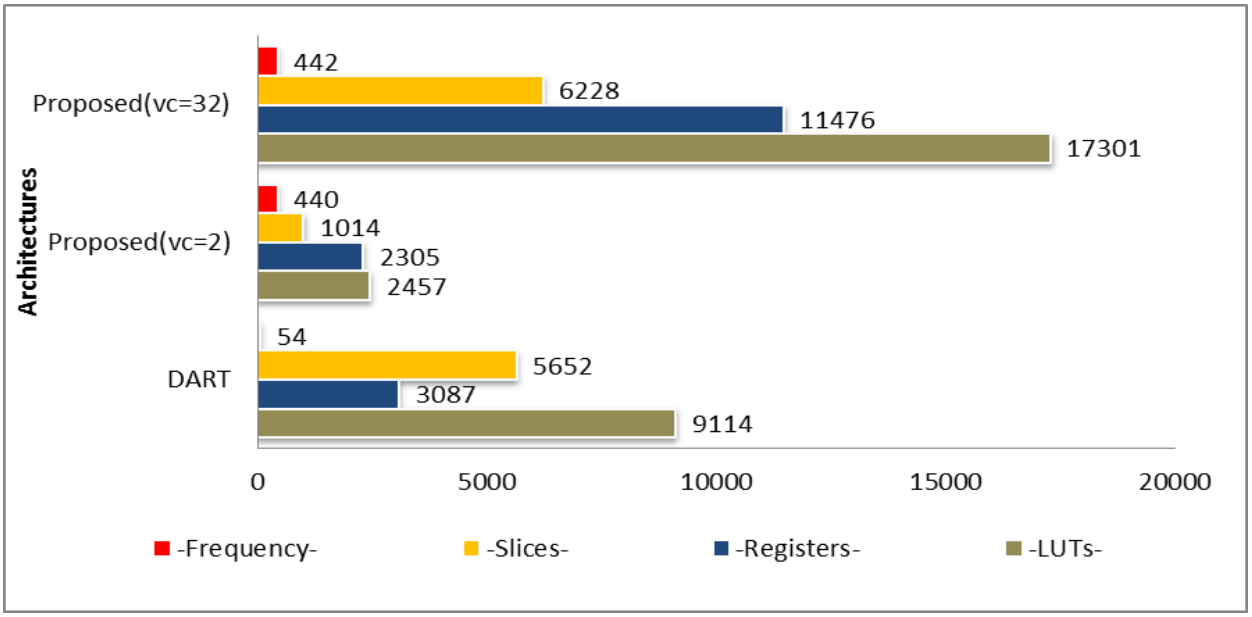
Figure 5.9 DART Datapath from [47]



In the Figure 5.10 below we see the difference between the two architectures on Virtex 6 device by Xilinx. Our architecture , with a maximum width in buffer size, reaches 6228 Slices whereas the DART Router reaches 5682. In fact , this comparison shows that the cost of the two Routers is almost the same , except that the architecture presented has a scope for reduction in any cost, by selecting a smaller buffer size. In the same graph, we see the vast difference in

maximum frequency our architecture can reach compared to that of the DART. 437 MHz is approximately the worst performance given to us by the Xilinx tools, opposed to approximately 50MHz by the DART.

Figure 5.10 Comparison between Dart and Proposed Architecture on, Frequency and Area



5.2.3 Compare with 5 Port Router Design [49]

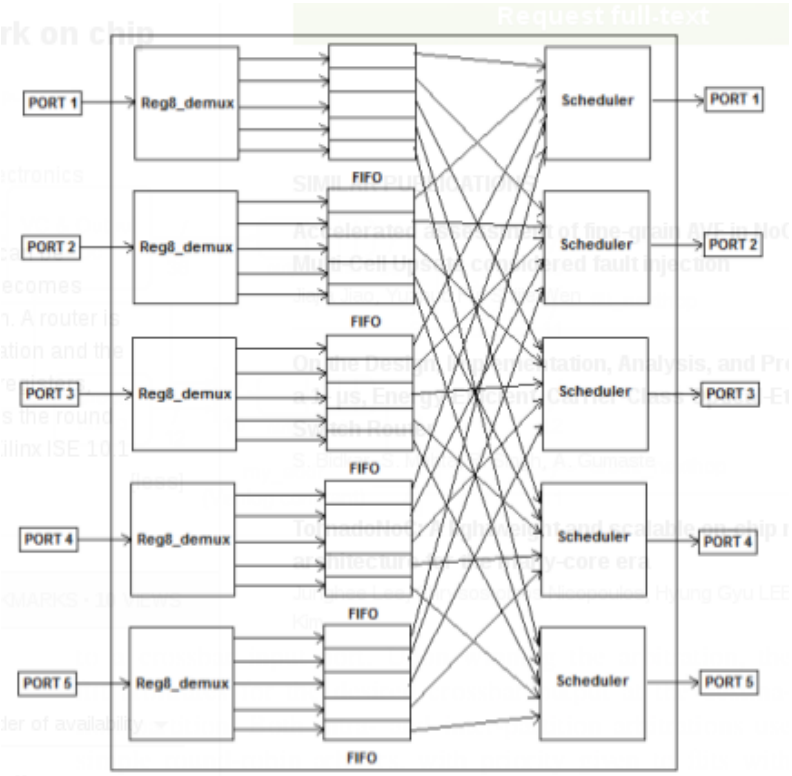
This router has 5 in/out port. The local port in utilized combined register and demux design The router ports The router architecture to connect the correspondent circle to the processing element (PE) and other ports are for connecting to other routers.

Figure 5.11 shows the internal architecture of the designed router. The building blocks of router consist of mainly three parts

1. Registers and demultiplexers
2. First In First Out Registers
3. Schedulers.

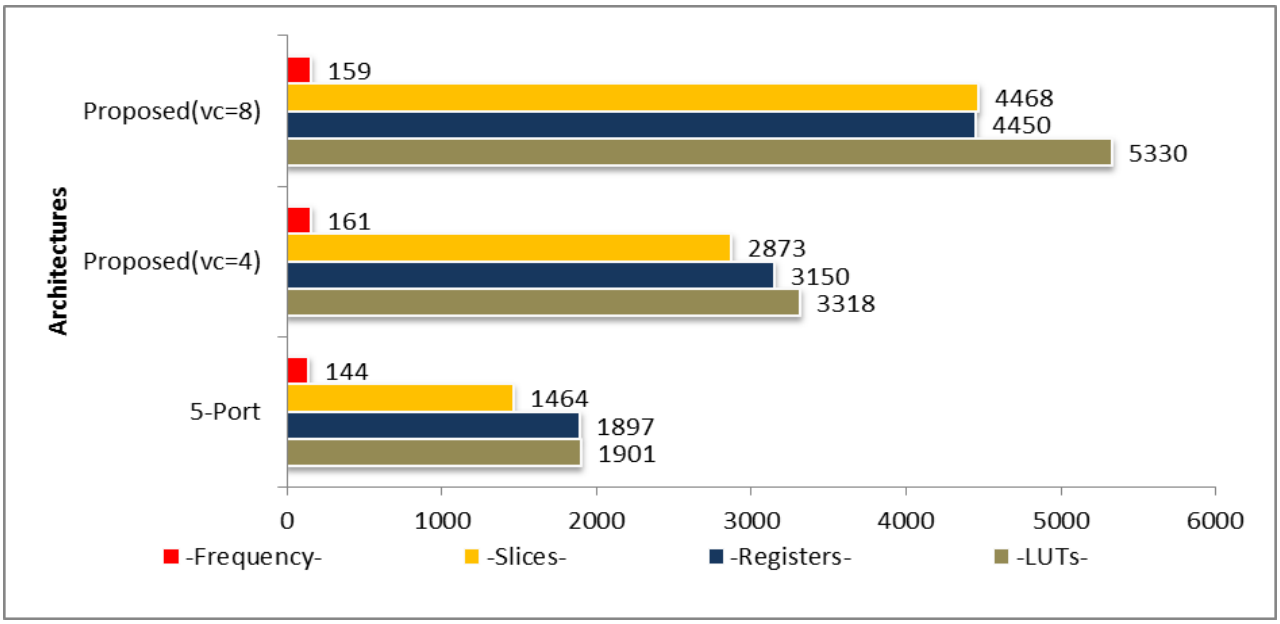
It is an architecture close to ours. The basic difference is that it does not support Virtual Channels or any other Control Flow technique.

Figure 5.11 Five-Port Router Architecture



In Figure 5.12 we can see the comparison of the architecture we present and the five-port architecture. Firstly , even though we have designed a more complex model, we measure a frequency of 161,655 MHz compared to the other architecture , on the SPARTAN 3E device by Xilinx. The difference is certainly noticeable in the Slices needed for our architecture, as the buffer size 8 is almost triple (it reaches 4468 opposed to 1464). The difference makes sense since our architecture is designed with Virtual Channels as we previously noted.

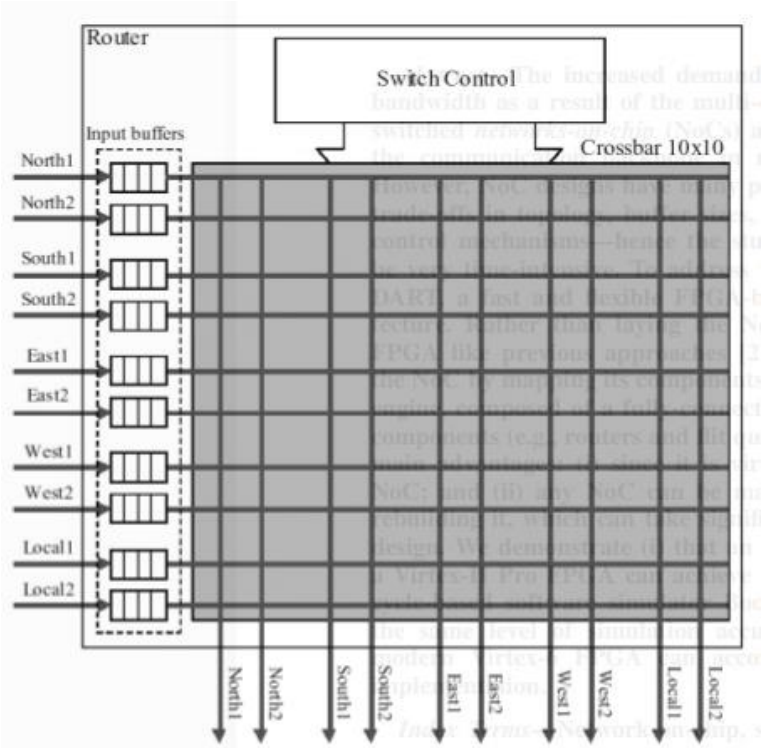
Figure 5.12 Comparison between 5-Port and Proposed Architecture on, Frequency and Area



5.2.4 Compare with Carara's Router [50]

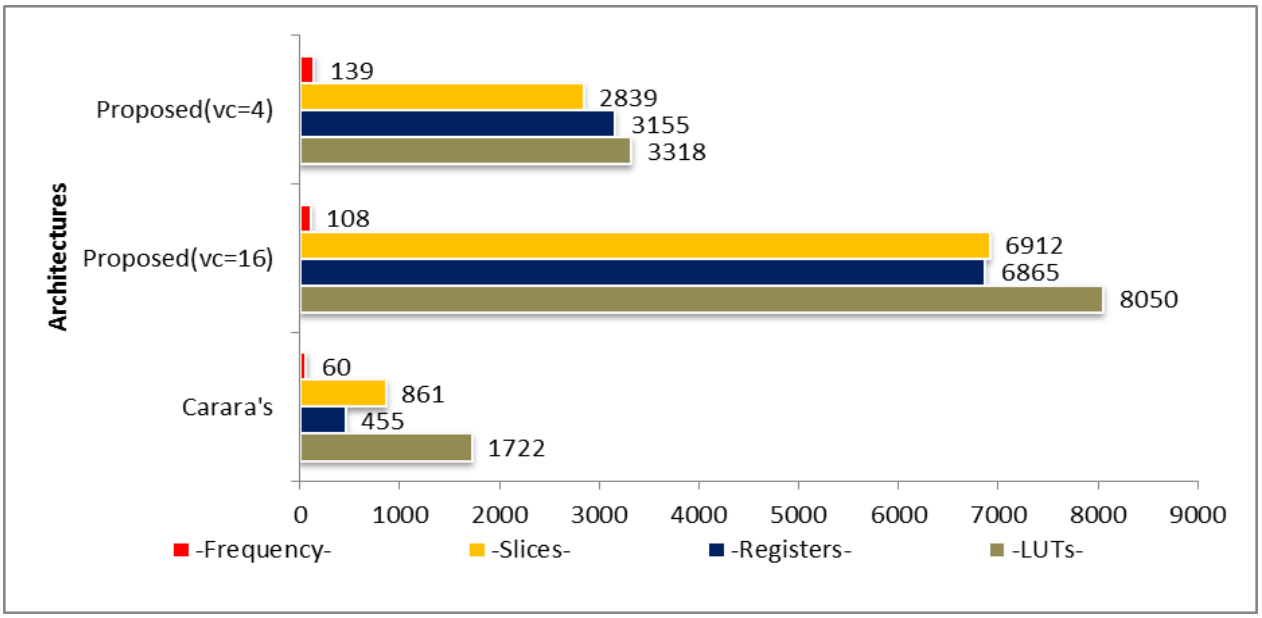
Carara's proposed methods to increase the overall performance of NoC routers. Results showed significant performance gains, demonstrating the effectiveness of the propositions, even with higher injection rates and flows competing for the same physical channel. Both replicated channels and circuit switching achieve latency reduction through congestion reduction. Replicated channels increase router bandwidth, whereas circuit switching coupled with a session protocol layer maximizes the physical channel utilization. The Figure 5.13 depicts the basic architecture of the Five-Port Router with replicated Channels.

Figure 5.13 Carara's Router Architecture with replicated channels



The architecture of this specific Router provides the convenience of low requirement in area of the device, with just 861 slices for the architecture supported by VCs, as it was measured on the Virtex-II by Xilinx. Regarding the maximum frequency, it reaches 50-60 MHz, whilst for us the minimum for this device was measured at 108,12 MHz as we can see on Figure 5.14

Figure 5.14 Comparison between Carara's and Proposed Architecture on, Frequency and Area

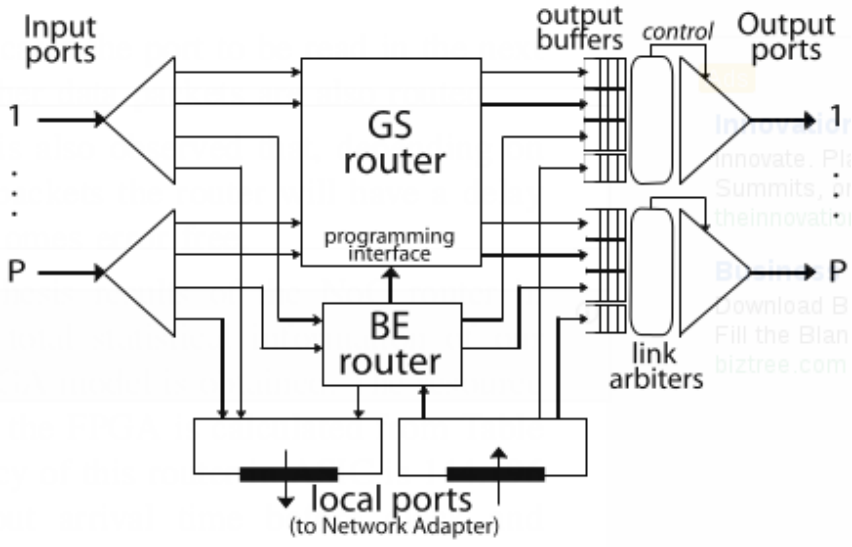


5.2.5 Compare with MANGO Router.

The MANGO architecture is one of the most widespread architectures for NoCs, something that is noted in previous sections. Tobias Bjerregaard and Jens Sparsø from Informatics and Mathematical Modelling Technical University of Denmark (DTU), also created the Router that can support this specific architecture. The MANGO NoC consists of network adapters (NA), routers and links. Each IP core is connected to the network through an NA, providing high level communication services, i.e. OCP transactions, on the basis of primitive services implemented by the network. Each NA, which also performs the synchronization between the clocked IP core and the clockless network, is connected to a router. The routers are connected by links in a grid- type structure, either homogeneous or heterogeneous. To keep speed up, long links can be implemented as pipelines. Figure 5.15 shows a conceptual picture of the MANGO router. The router implements a number of unidirectional ports. Two of these are local ports which connect to the NA. The local ports consist of a number of physical interfaces. The remaining ports are network ports which, via point- to-point links, connect the router to

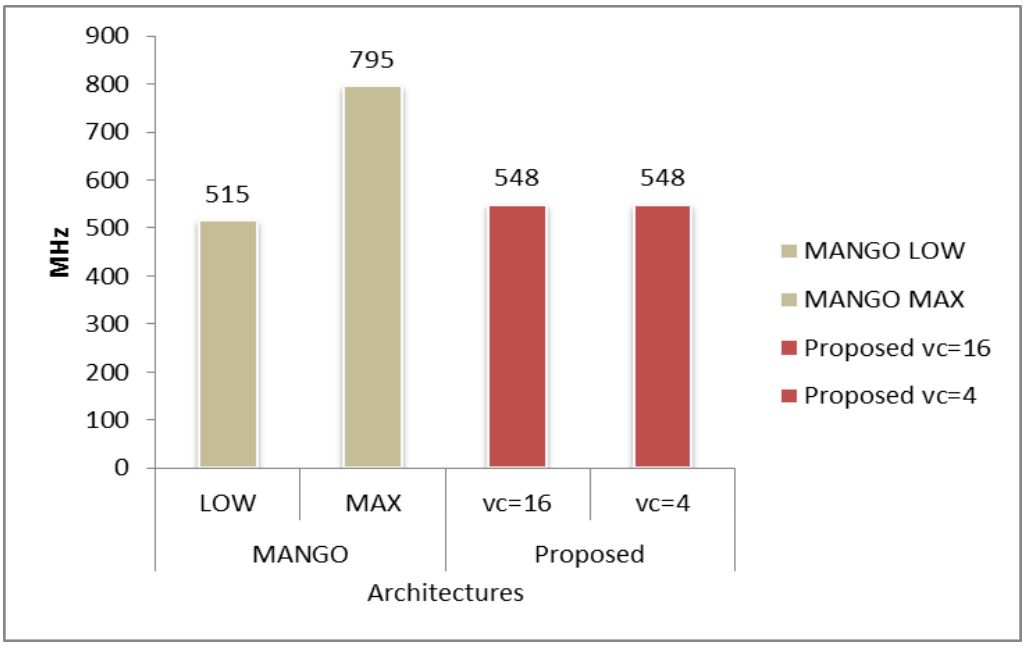
neighboring routers. Each of these ports implement a number of independently buffered VCs. Network ports and local ports implement the same type of interface. It is the function of the NA to translate communication to and from network packet format.

Figure 5.15 MANGO Router Architecture [51]



As we can see in Figure 5.16 we do not have sufficient data on the MANGO resource utilization for the Virtex 7 FPGA of Xilinx. Therefore we only compared the frequency of the two architectures, which shows that the Router with a frequency of 548 MHz exceeds the MANGOs output, given by its designers it reaches 515 MHz in worst case for each port. Its designers also claim that under typical timing conditions it can reach 795 MHZ.

Figure 5.16 Comparison between MANGO and Proposed Architecture on Frequency



Comparison Figures show that our architecture offers a combination of High Performance and low cost area based on its technology. It is obvious that in future work we should be concerned as much on the further optimization of the performance and the support of more VCs, with the addition of output buffers the minimum request in device area.

Chapter 6 - Conclusions and Future Work

With current FPGAs that deliver high bandwidth, high level of system integration, and convenient flexibility with reduced cost and high frequency, FPSoCs are promising to provide reliable and practical digital systems for contemporary applications that require intensive computations and limited dimensions.

Since NoCs are a favorable approach to overcome the limitations of traditional bus-based and point-to-point on-chip communications used with SoCs, more research is needed to explore the design space of FPGA-based NoCs, and to offer more efficient solutions to existing NoC drawbacks. This thesis tries to contribute to this research by exploring the design space of NoC routers, as a dominant component of the network, and introduces novel techniques that enhance the area of the router and boost its performance.

We implemented parameterized VHDL models of NoC-Routers on FPGA Devices. The focus of our approach is to minimize the area of the router because area is at a premium on an FPGA. At the same time, we introduced good techniques to speed up the router.

Our router in general is a 5-ports packet switched router with deterministic XY-Routing algorithm and dynamic round robin arbitration scheme. The depth of input buffers (VCs), located at the input ports and the flit size (channel width), is parameterized by means of VHDL generics. We presented the general architecture of the router as well as the architectural and functional differences between four different devices with different buffer size in any device. The strategies that followed to decrease the area and increase the frequency in the general router design are:

- Minimize the number of control fields in the packet as well as eliminate the tail flit, while replacing them with some logic and counters in the routing unit to infer the start and end of the packets, and consequently, reduce the FIFO size. Header and data can be sent together in one flit.
- Use the credit based flow control, so we are able to transmit the flit in only one clock cycle, instead of using the hand shaking protocol that needs at least two clock cycles.

- Utilize the rising edge of the clock to accomplish and synchronize some operations while employing the falling edge to undertake other operations. This technique enables us to squeeze a number of dependable operations in only one clock cycle either in the FIFO Finite State Machine (FSM) Read/Write operations, or arbitration and routing handling.

The functionality of our router design was verified for a different-generic size of VCs buffers, QuestaSim simulator, and it was synthesized for the Xilinx FPGA Devices (Virtex-II, Spartan 3E, Virtex 6 and Virtex 7) using the ISE 14.7 synthesis tool. The simulation also demonstrates the feasibility of the dual-clock mechanism. Comparison of the synthesis results, with other published routers, were conducted and discussed, based on two metrics: area, frequency. Synthesis results evaluations show that our router is significantly superior to widely referenced, previously proposed routers. We also conducted throughput and injection rate measures, additional with the drop rate measure.

In short, we can summarize our future work in the following points:

- Develop our deterministic router to be an adaptive router that can provide quality of service.
- Use real applications or standard bench marks, if found in the future, to test our router and compare them to other proposed work.
- Evaluate the reduction of the latency results from compacting the effective length of body flits in the time domain (via simulation).
- Is for study the effect on the correlation of the Network Interface buffers with internal buffers (the size), the throughout for each port.

References

[1] Luca Benini and Giovanni De Micheli, 'Networks on chips: a new So paradigm," Computer, vol. 35, no. 1, pp. 70-78, 2002

[2] Rickard Holsmark and Magnus Hgberg, 'Modelling and Prototyping of a Network on Chip," Master of Science Thesis, 2002 Electronics

[3] Muhammad Ali, Michael Welzl, Martin Zwicknagl, 'Networks on Chips: Scalable Interconnects for Future Systems on Chips," 4th European Conference on Circuits and Systems for Communications, pp. 240-245, 2008

[4] Xinan Zhou, 'Performance evaluation of network-on-chip interconnect architectures," Master of Science Thesis, 2009 Electrical Engineering

[5] G.Kornaros, D. Pnevmatikatos, "Real-Time Monitoring of Multicore SoCs through Specialized Hardware Agents on NoC Network Interfaces", Proc. of 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshop (IPDPS Workshops), 2012, pp. 248-255

[6] W. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.

[7] D. Kim, Manho Kim, and G.E. Sobelman, "CDMA-based NoC architecture", Proc. IEEE Conference on Circuits and Systems, vol. 1, pp. 137-140, 2004.

[8] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C.A. Zeferino, "SPIN: a scalable, packet switched, on-chip micro-network, Proc. IEEE Conference on Design, Automation and Test, pp. 70-73, 2003.

[9] P. Pratim Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", IEEE Transactions on Computers, vol. 54, no. 8, pp. 1025-1040, 2005.

[10] F. Karim A. Nguyen, and S. Dey, "An interconnect architecture for networking systems on chips", IEEE Journal on Micro High Performance Interconnect, vol. 22, issue 5, pp. 36-45, Sept 2002.

[11] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection. Proceedings of the 38th Design Automation Conference. June 2001

[12] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, "Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC", Boca Raton, FL, USA: CRC Press, Inc., 2008.

[13] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections", Design, Automation and Test in Europe Conference and Exhibition 2000. March 2000, pp. 250-256.

[14] S. Kumar, A. Jantsch, J-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology", IEEE Computer, pp. 117-124, 2002.

[15] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo network-on-chip", International IEEE Proceeding on Circuits and Systems, vol. 2, pp. 113-116, May 2003.

[16] J. Liu, L.-R. Zheng, and H. Tenhunen, "A guaranteed-throughput switch for network-on-chip", Proc. The International Symposium on System-on-Chip, pp. 31-34, 2003.

[17] H. Zimmer, S. Zink, T. Hollstein, and M. Glesner, "Buffer-architecture exploration for routers in a hierarchical network-on-chip", Proc. 19th IEEE International Symposium on Parallel and Distributed Processing, pp., 1-4, April 2005

[18] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit, "An energy efficient reconfigurable circuit-switched network-on-chip", Proc. 19th IEEE International Conference on Parallel and Distributed Processing Symposium, pp. 155-163, 2005.

[19] J. Xu, W. Wolf, J. Henkel, S. Chakradhar, and T. Lv. "A case study in networks-on-chip design for embedded video", IEEE Proc. on Design Automation and Test in Europe Conference, vol. 2, pp. 770-775, February 2004.

[20] S. J. Lee, K. Lee, S. J. Song, and H.-J. Yoo, "Packet-switched on-chip interconnection network for system-on-chip applications", IEEE Trans. On Circuits and Systems-II, vol. 52, no. 6, pp. 308-312, 2005.

[21] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless Network-on-Chip", Proc. Of IEEE on Design Automation and Test, vol.2, pp. 1226-1231, 2005.

[22] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, "Providing QoS to connection-less packet-switched NoC by implementing DiffServ functionalities", Proc. 2004 International Symposium on System-on-Chip, pp. 37-40, 2004.

[23] W. J. Dally, "Virtual-channel flow control," IEEE Transactions on Parallel and Distributed Systems, no. 2, pp. 194-205, 1992.

[24] William J. Dally and Charles L. Seitz, "The torus routing chip", Journal of Distributed Computing, 1(3):187-196, 1986.

[25] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip", Journal of Systems Architecture, Volume 50, Issue 2-3 (Special Issue on Network on Chip), pp. 105-128,February 2004.

[26] C. A. Zeferino and A. A. Susin, "SoCIN: A parametric and scalable network-on chip", Proc. 16th Symposium on Integrated Circuits and Systems Design, pp. 169-175, 2003.

[27] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip", IEEE Circuits and Systems Magazine, vol. 4, Issue 2, pp. 18-31, 2004.

[28] Luca Benini, and Giovanni De Micheli, "Networks on chips: A new SoC paradigm", in Proceedings of IEEE Computer 35, no. 1, pp. 70-78, 2002.

[29] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", IEEE Transactions on Computers, pp. 547-553, 1987.

[30] F. Feliciian and S.B. Furber, "An asynchronous on-chip network router with quality-of-service (QoS) support", in Proceedings of the IEEE International SOC Conference, pages 274-277, 2004.

[31] T. Bjerreegaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip", In Journal ACM Computing Surveys (CSUR) Surveys Homepage archive Volume 38 Issue 1, 2006 Article No. 1

[32] N. Chabini and W. Wolf, "Reducing dynamic power consumption in synchronous sequential digital designs using retiming and supply voltage scaling", IEEE Transactions on VLSI Systems, vol. 12, no. 6, pp. 573-589,2004.

[33] L. Yan, J. Luo, and N. K. Jha, "Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems", Proc. IEEE International Conference on Computer Aided Design, pp. 30-37, 2003.

[34] G. Kornaros, "Temporal Coding Schemes for Energy Efficient Data Transmission in Systems-on-Chip", in Proceedings of IEEE Workshop on Intelligent Solutions in Embedded Systems Ancona, Italy, June 25-26, 2009.

[35] Liang, J., Swaminathan, S., Tessier, R. "aSOC: A Scalable, Single-Chip communications Architecture", In: IEEE International Conference on Parallel Architectures and Compilation Techniques, Oct. 2000, pp. 37-46.

[36] Ni, L. M.; McKinley, P. K. "A Survey of Wormhole Routing Techniques in Direct Networks", IEEE Computer Magazine, v.26 (2), February, 1993, pp. 62-76.

[37] A. Vieira de Mello, L. Copello Ost, F. Gehm Moraes, N. L. V. Calazans, "Evaluation of Routing Algorithms on Mesh Based NoCs", Faculdade de Informática – PUCRS, Technical Report series num. 040, May 2004.

[38] N. Concer, S. Iamundo, L. Bononi, "aEqualized: a Novel Routing Algorithm for the Spidergon Network On Chip", DATE, 2009

[39] Jongman Kim, Dongkook Park, T. Theocharides, N. Vijaykrishnan and Chita R. Das, "A Low Latency Router Supporting Adaptivity for On-Chip Interconnects", DAC 2005, June 13–17,

[40] M. H. Neishaburi, Z. Zilic, Reliability Aware NoC Router Architecture Using Input Channel Buffer Sharing, ACM GLSVLSI 2009, May 10-12.

[41] Seung Eun Lee and Nader Bagherzadeh, "Increasing the Throughput of an Adaptive Router in Network-onChip (NoC)", CODES+ISS 2006, October 22-25

[42] S. Ma , N. E. Jerger , Z. Wang, "Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-Chip", 18th IEEE International Symposium on High Perfomance Computer Architecture, 2012.

[43] M. Ebragihimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, H. Tenhunen, "DyXYZ: Fully Adaptive Routing Algorithm for 3D NoCs", 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2013

[44] M. Palesi, R. Holsmark, S. Kumar and V. Catania, "Application Specific Routing Algorithms for Networks on Chip", IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 3, March 2009

[45] L. Rooban, S. Dhananjeyan, "Design of Router Architecture Based on Wormohole Switching Mode for NoC", International Journal of Scientific & Engineering Research Vol, 3, Issue 3, Mar. 2012

[46] E. Behrouzian-Nezhad and Ahmad Khademzadeh, "BIOS: A New Efficient Routing Algorithm for Network on Chip, Contermporary Engineering", Vol.2, 2009, no.1, p. 37-46

[47] D. Wang, Charles Lo, J. Vasiljevic, N. E. Jerger, and J. Gregory Steffan, "DART: A Programmable Architecture for NoC Simulation on FPGAs", IEEE Transactions On Computers, 2014, 63(3), pp. 664-678

[48] N. Kavaldjiev, Gerard J. M. Smit, P. Jansen, "A Virtual Channel Router for On-Chip-Networks", IEEE International SOC Conference, 2004

[49] S. Swapna, A.K. Swain, K.K. Mahapatra, "Design and Analysis of Five Port Router for Network On Chip", Microelectronics and Electronics (Prime Asia), Asia Pacific Conference on Postgraduate Research 2012.

[50] E. Carara, N. Calazans, F. Moraes, A New Router Architecture for High-Perfomance Intrachip Networks, Journal Integrated Circuits and Systems, vol.3, 2008.

[51] T. Bjerregaard and J. Sparsø, A Router Architecture for Connection-Orented Service Guarantees in the MANGO Clockless Network-on-Chip, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, DATE, 2005

[52] Miltos Grammatikakis, G. Kornaros, M. Coppola, "Power-Aware Multicore SoC and NoC Design", Multiprocessor System-on-Chip, Book Chapter 8, pp. 167-193, Eds Michael Hübner , Jürgen Becker, Springer, 2011

[53] G. Kornaros and T. Orfanoudakis, "Design and Implementation of High-Speed Buffered Crossbars with efficient Load Balancing for Multi-Core SoCs", Journal on Microprocessors and Microsystems, vol. 34 (7-8), 2010, pp. 301-315

[54] G. Kornaros, "A Buffered Cross-Bar Switch Fabric Utilizing Shared Memory", In Proc of the 9th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), 2006, pp. 180-188

[55] G. Kornaros, I. Christoforakis, M. Astrinaki, "An automated infrastructure for real-time monitoring of multi-core systems-on-Chip", Proc. of the 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2012, pp. 56-61