NOC ROUTER AND ARBITER FOR FPGA

By

Khaled Abdullah Helal Kelany

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfilment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

NOC ROUTER AND ARBITER FOR FPGA

By

Khaled Abdullah Helal Kelany

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfilment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Prof. Hossam A. H. Fahmy
Professor
Electronics and Communications Engineering Department
Faculty of Engineering, Cairo University

Assist. Prof. Hassan Mostafa
Assistant Professor
Electronics and Communications Engineering Department
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

NOC ROUTER AND ARBITER FOR FPGA

By

Khaled Abdullah Helal Kelany

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfilment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Approved by the Examining Committee:

Prof. Hossam A. H. Fahmy, Thesis Main Advisor

Prof. Amin M. Nassar, Internal Examiner

Prof. Mohab H. Anis, External Examiner
(Faculty of Engineering, AUC)

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

Engineer's Name: Khaled Abdullah Helal Kelany
Date of Birth: 16/06/1989
Nationality: Egyptian
E-mail: khaled.a.helal@ieee.org
Phone: +201062297707
Address: Electronics and Communications Engineering Department,
Cairo University,
Giza 12613, Egypt

Registration Date: 01/10/2013
Awarding Date: / /2016
Degree: Master of Science
Department: Electronics and Communications Engineering

Supervisors:

Prof. Hossam A. H. Fahmy
Assist. Prof. Hassan Mostafa

Examiners:

Prof. Hossam A. H. Fahmy (Thesis main advisor)
Prof. Amin M. Nassar (Internal examiner)
Prof. Mohab H. Anis, Faculty of Engineering, AUC (External examiner)

Title of Thesis:

NOC ROUTER AND ARBITER FOR FPGA

Key Words:
NoC; Router; Arbiter; FPGA; Switch

Summary:
The advancement in semiconductor technology has led to a high density chip which moves the bottleneck from the on-chip computation systems to the on-chip communication systems. This advancement gives FPGA a chance to compete with AISC. However, the conventional communication paradigms failed to fulfill the on-chip system needs. Networks-on-Chips (NoCs) are considered a promising solution for on-chip communications challenges. This thesis investigates developing a high performance NoC that targets FPGA.

Acknowledgments

First, praise be to Allah, Lord of the worlds, who said: ”I am as My servant thinks I am”.

Second, I would like to express my special thanks of gratitude to my advisors, Dr. Hossam Fahmy, and Dr. Hassan Mostafa, who helped me gain knowledge in so many aspects of life and gave me a great opportunity to do this thesis, I am truly appreciative to them.

Finally, I would also like to thank my parents and my brothers, Hany and Tamer, who made me who I am today, as well as, my friends who helped me a lot in finalizing this work.

Dedication

I dedicate this work for my family, for their endless support.

I dedicate this work for my supervisors, for their sincere guidance.

Table of Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

Symbols	Description
ASIC	Application-Specific Integrated Circuit.
BSV	Bluespec System Verilog .
CONNECT	CONfigurable NEtwork Creation Tool.
DAMQ	Dynamically Allocated Multi-Queue.
DSM	Dual-Split-Merge .
DVOQR	Dynamic Virtual Output Queues Router.
EVC	Efficient Virtual Channel .
FC-CB	Fully Connected Circular Buffere.
FIFO	First In First Out.
Flit	Flow Control Digit.
FOM	Figure of Merit.
FPGA	Field Programmable Gate Array.
HDL	Hardware Description Language.
HOL	Head of Line.
IP	Intellectual Property.
LUT	Look-up Table.
NI	Network Interface.
NoC	Network on Chip.
PE	Processing Element.
PS	Priority-Select .
RAM	Random Access Memory.
RRA	Round-Robin Arbiters.
RTL	Register Transfer Level.
SAF	Store And Forward.
SCB	Self-Compacting Buffer.
SoC	System on Chip.
SOTA	State-Of-The-Art.

UDB	Unified Dynamic Buffer .
UDBA	Unified Dynamic Buffer Allocator.
VC	Virtual Channel.
VCT	Virtual Cut Through.
ViChaR	Virtual Channel Regulator.
VOAQ	Virtual Output Address Queues.
VOQ	Virtual Output Queueing.
WH	Wormhole.

List of Publications

Published:

[1] K. A. Helal, S. Attia, T. Ismail, and H. Mostafa, "Comparative review of NoCs in the context of ASICs and FPGAs," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2015, pp. 1866–1869.

[2] K. A. Helal, S. A. Attia, T. Ismail, and H. Mostafa, "Priority-select arbiter: An efficient round-robin arbiter," in *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International*, IEEE, 2015, pp. 1–4.

Submitted:

[1] K. A. Helal, S. Attia, T. Ismail, and H. Mostafa, "Dual Split-Merge: A High Throughput Router for FPGAs," *IEEE Transactions on Very Large Scale Integration Systems*, 2016.

Abstract

The advancement in semiconductor technology has led to a high density chip which moves the bottleneck from the on-chip computation systems to the on-chip communication systems. This advancement gives FPGA a chance to compete with AISC by increasing the embedded various processing elements with flexible interconnection infrastructure. However, the conventional communication paradigms failed to fulfill the on-chip system starvation for high speed, high bandwidth, low overhead, and modular communication system. Networks-on-Chips (NoCs) is considered a promising solution for on-chip communications challenges correlated with technology scaling. In this thesis, we investigate developing a high performance NoC that targets FPGA.

This thesis compares and discusses the NoC parameters that give high performance and efficiency for FPGA-based and ASIC-based NoC. In addition, it introduces a new FPGA-based NoC router architecture that outperforms the previous router architectures found in the literature. Moreover, this thesis proposes a new implementation architecture for Round-Robin arbiter that significantly reduces the area and operating frequency when compared to the previous implementations found in the literature. This arbiter greatly enhances the operating frequency when embedded with the proposed NoC router architecture. Finally, the thesis equips the proposed NoC router architecture with one of the dynamic memory schemes to further reduce the used area.

Chapter 1

Introduction

As the integrated circuits technology scales, the number of transistor that can fit in a single chip increases and the question is no longer how to increase it, but how to efficiently use it. In current systems-on-chip (SoC), the designers tend to use more logic and intellectual property (IP) cores to implement modern applications that need broad processing. These complex systems need more intensive interconnections.

On chip interconnections significantly affect the performance of the applications implemented on the chip. Nevertheless, the direct interconnections and classical bus-based interconnections have major disadvantages such as the synchronization errors correction, high delay and high power consumption[1]. Here, network-on-chip (NoC) arise as a solution to interconnection problems [2].

NoC overcomes the limitations of bus-based interconnection by introducing an efficient way to realizing interconnections on SoC [3–5]. NoC has a high performance interconnection, with low power and scalability properties [6]. It is also easy to scale as it provides multiple connections between processing elements (PEs). In addition, it decreases system complexity by proposing a level of abstraction by separating data communication from data computation.

In NoC architectures, PEs are connected via a packet-switched network on a single chip. It also increases bandwidth by pipelining data transmission in single or multiple channels in parallel.

However, NoCs have various problems such as latency and area increases, higher power consumption and congestion. On the other hand, the platform implementation medium has a significant effect on SoC applications. Most NoC researches are directed to the ASIC platforms because its advantages over the other competitor, FPGA platforms, in high operating frequency and small area and power consumption. However, FPGAs become more attractive for SoC designers as it provide many preferences as we will discuss in chapter 2.

1.1 Thesis Goals

Some researches now are directed to FPGA-based NoC as the current designs need many improvements and ASIC-based NoC designs may not be applicable for FPGA-based ones [7].

There are many parameters that affect NoC performance, these parameters can be classified into two categories; the parameters related to network structure such as latency, throughput and bandwidth, and the parameters related to implementation such as operating frequency, area and power consumption [7].

In this thesis, we are working on:

1. Enhancing those parameters to elevate the overall performance.
2. Exploring the methods that provide a fast and yet small area NoC that is targeting FPGA platform.
3. Developing FPGA-based NoC system by introducing a new router architecture and improving bottleneck system components.
4. Improving the critical components to efficiently support NoC without the need of implementing hard-embedded routers, namely, we are focusing on the arbiter and buffer units of the NoC routers.

The results included in this thesis are based on simulating parameterized Verilog models of NoC-Routers on Virtex-6 LX760 FPGA (part xc6vlx760, speed grade -2), close to the technology used in recent NoC researches. As well as the simulation on Virtex-5 LX330T FPGA (part xc5vlx330t, speed grade -2) and UMC 65nm Typical as they both represent similar technology feature length to provide a fair comparison between the implementation on FPGA and ASIC platforms respectively.

1.2 Organization of the thesis

The thesis is organized as follows: Chapter 2 presents a background on NoC and FPGA. Chapter 3 compares the NoCs that target the FPGA with the NoCs that target the ASIC. In chapter 4,we propose a very high throughput router in which both the network part and the chip part are improved. In chapter 5, we propose a new Round-Robin arbiter architecture that provides significant performance improvements over previous architectures. Chapter 6 discusses augmenting the proposed router with dynamic buffers. Finally, Chapter 7 concludes the thesis contributions and possible future work directions.

Chapter 2

Background

This chapter provides a quick background of packet-switched NoC design and a discussion about the characteristics of FPGA platforms and how they differ from ASIC platforms.

2.1 On-Chip Interconnect Architecture

Traditionally, SoC's cores were communicating through shared buses, point-to-point connection, or a mixed system of shared buses and point-to-point.

Shared buses are based on a single bus or multi buses as figure 2.1 shows. The advantage of this paradigm is the simplicity, but it has many disadvantages like small bandwidth, no concurrent communications, and it does not scale well, which make it suitable for small systems only

Point to point also has the advantage of large bandwidth and low latency, however, it causes routing problems and the number of connections scales quadratic with the number of cores. That make it as well suitable for small systems as figure 2.2 shows.

2.2 NoC

NoC consists of network interfaces (NIs) that provide an interface between cores that need to communicate, network elements (routers) that transmit and receive the messages from neighboring routers or NI's, and links connect routers and NIs. The idea of NoC is based on traditional computer networks, but with many differences. Figure 2.3 shows a sample of NoC.

NoC provides high bandwidth, low latency, low area-overhead, low power, and simple scaling paradigm when compared with shared bus or point-to-point connection. Nevertheless, it has many issues that need to be considered which make it a good field of research. In the following section, we introduce some terminology used in NoC design:

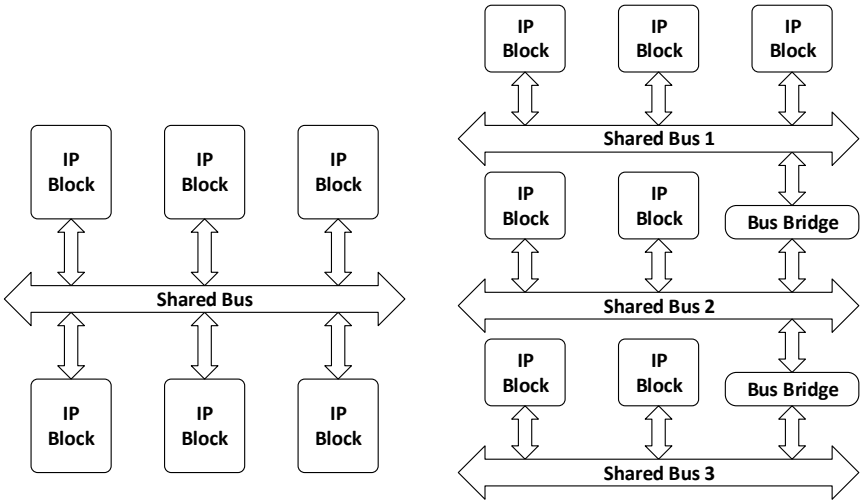


Figure 2.1: Single bus and multi buses

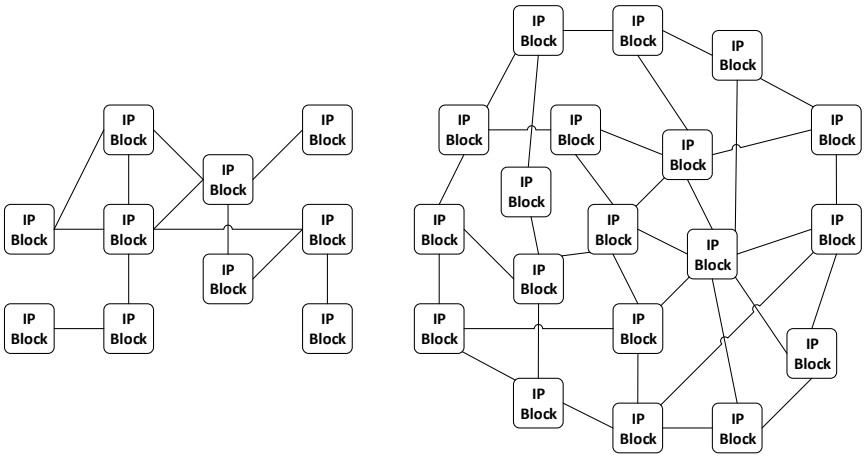


Figure 2.2: Point to point

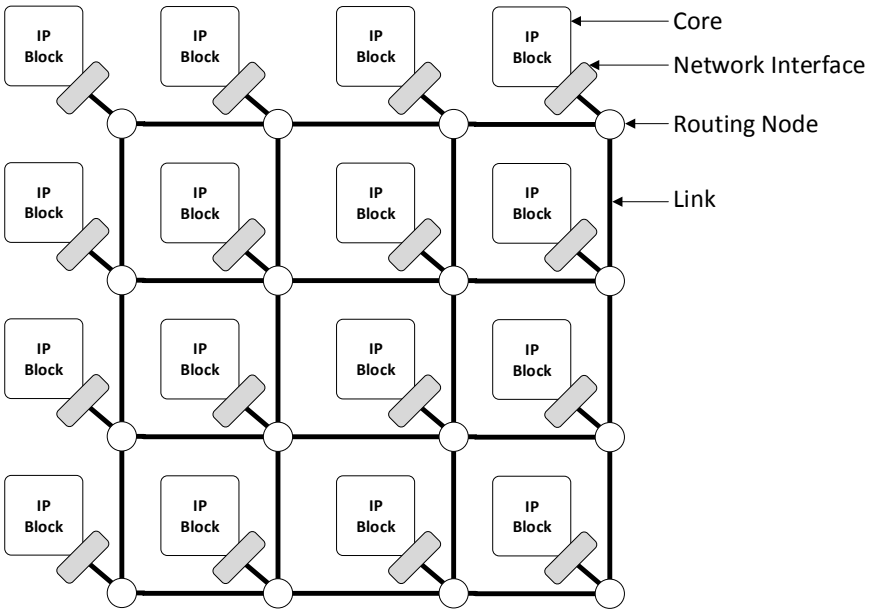


Figure 2.3: Sample of NoC. The sample shows 16 IP blocks that communicate through a network of NI's (rectangular shapes), routers (circles), and links (solid lines).

Deadlock:

Deadlock happens when two packets are acquiring some resources and each waiting for the other to release these resources before proceeding. Which causes the two packets to suspend.

Livelock:

Livelock happens when a packet continuously loops around its destination and fails to reach it.

Starvation:

In multi priority system starvation happens when low priority packets are suspended permanently because the higher priority packets are acquiring the network resources continuously.

2.2.1 NoCs' Parameters

NoC design is characterized by its parameters; these parameters can be summarized as follow:

2.2.1.1 Topology:

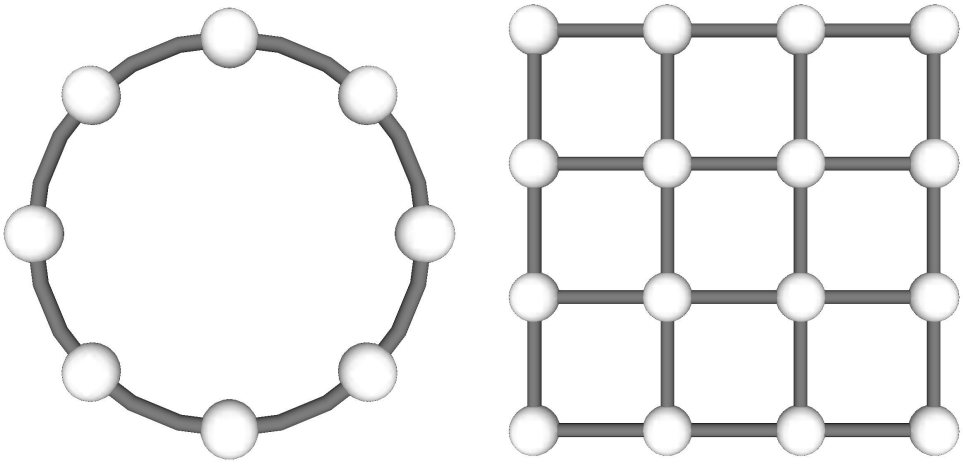
NoC are basically composed of a group of shared nodes (routers) and links between nodes to connect the processing elements (PE). The topology of the network refers to the distribution and the arrangement of the routers, links, and processing elements (PE).

The key of a good topology is to achieve the bandwidth and the latency required by the application at the lowest cost using available, or targeted, technology. Moreover, network topology has an impact on the overall performance as it allows the network to scale regularly. In order to achieve the maximum bandwidth, the topology should saturate the bandwidth at the midpoint of the system [8].However, to minimize the latency, the topology should compromise between minimizing the average distance between nodes, hop count, and minimizing the serialization latency. In other words, given a limited capacity of the wiring, tendency to increase the links will result in narrowing the link width which will increase the serialization, and hence increase the buffering of the large packets. Decreasing the hop count, the number of nodes and links the message traverse from source to destination, will result in increasing the number of links connected to each node which may complicate the implementation of nodes.

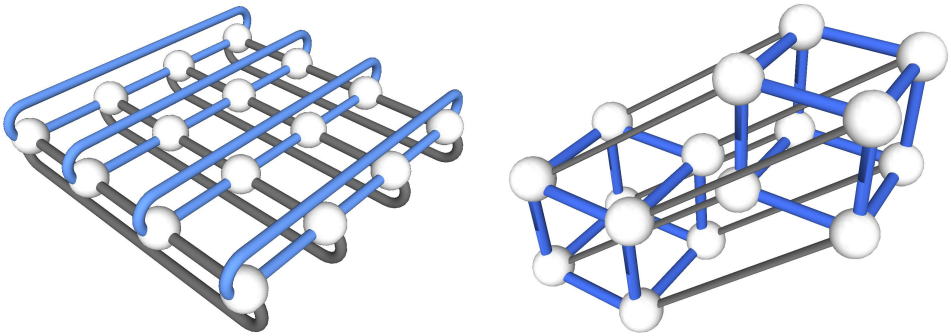
Topology could have high dimension orientation like three-dimension networks or it could be two-dimension networks or even one-dimension networks. Figure 2.4 shows examples of one-dimension, two-dimension, and three-dimension topologies. Ring topology is an example of the one-dimension topologies, mesh and torus are examples of the two-dimension topologies, and hypercube topology is an example of the three-dimension topologies. In the figure, the circles refer to the network nodes and the lines refer to the bidirectional links. However, in two-dimension platforms, like FPGAs, it is not preferable to use a high dimension network [9] because it uses a long wire connection, which increases the delay, and power consumption. On the other hand, two-dimension networks such as mesh, torus and Flattened butterfly are promising topologies for FPGA [7]. Moreover, mesh topology has the advantage over torus and Flattened butterfly because even though torus and Flattened butterfly have low latency measured in number of cycles, they are suffering from a low operating frequency that result from the long wires used [7]. In this thesis, we will use Mesh topology as the default network topology to measure the performance as we will discuss in the next chapter.

2.2.1.2 Flow Control:

The flow control ensures that the network resources are free from the conflict that may keep a channel idle. The flow control method determines how packets are assigned to network resources, including buffers, ports, channels and control logic. In other words, it determines when to hold the transmission if buffers or any other resources are unavailable to avoid data loss.



a) Ring b) Mesh



c) Torus d) Hypercude

Figure 2.4: Topologies

To avoid buffer overflow, NoC uses flow control protocol. The most common two flow control protocols are credit-based flow control [10] and valid/backpressure flow control [8]. The credit-based flow control receiver keeps sending information, or credits, for each portion of packet it receives and sender keep counting the credit to figure out the space available at receiver buffers, while in backpressure flow control the receiver sends a single bit to the sender to indicate whether the buffer is full or not. Backpressure flow control is simpler and provides relatively good performance.

2.2.1.3 Switching Techniques:

Switching Techniques refer to the way packets pass from source to destination. The effectiveness of a switching strategy is measured by its ability to provide fairness between network packets and avoid any deadlock. The most known techniques are circuit switching (CS) and packet switching (PS).

In CS, the network resources are reserved from source to destination, so it needs time to setup the connection before streaming the message and release the connection after streaming ended. CS provides guaranteed bandwidth, however, it may make reserved resources unusable when transmission is idle which decreases resources utilization, also setup and release times increase the latency.

On the other hand, PS sends packets directly without resource reservation, which increases resource utilization and decreases the average delay. PS can be classified to three methods:

Store-and-forward (SAF) [10]:

In store-and-forward, the message is decomposed into packets, each packet sent sequentially after subdividing it into smaller units called flow control digits (flits) according to bus width. However, it's not allowed to the router to forward the packet immediately until the entire packet received and stored in the buffer. Therefore, it requires large buffers and has high latency.

Wormhole (WH) [8]:

In wormhole, each packet is decomposed into flits. Unlike Store-and-forward, each flit can be foreword immediately without the need of storing the whole packet, so it need smaller buffers and has lower latency. However, it is prone to deadlock.

Virtual cut through (VCT) [10]:

In virtual cut through, the flits can be forwarded if the next router has sufficient space in its buffer to hold the whole packet. Although virtual cut through has low latency, it requires large buffers.

Compared with virtual cut through and store-and-forward, wormhole routing is suitable for NoC because it needs relatively small buffers and has low latency. The problem of deadlock can be solved using a deadlock free routing algorithm or introducing the concept of virtual channels.

Figure 2.5 shows the differences between Store-And-Forward (SAF), Wormhole (WH) and Virtual Cut Through (VCT). In the figure, the space and time are shown on vertical and horizontal axis respectively. The space represents the channels used by the packet. The packet is a group of six flits starting with the head flit. The time is measured in cycles.

2.2.1.4 Virtual Channels (VCs) [11, 12]

Virtual channels are the virtual link that connects two adjacent switches. Each physical channel, which physically connect two adjacent switches, could have one or more virtual channels each with a different buffer space. VCs contribute in solving the problem of deadlock and head-of-line (HoL) blocking, the blocking of data destined for a port of the router behind data waiting for another port to be available, so it increases the network performance at the expense of logic that is more complicated and larger buffer space.

2.2.1.5 Routing Algorithms

Routing Algorithms are responsible for determining the path that the packet will take from the source to the destination. In computer networks, routing algorithms are sophisticated which is not applicable in NoCs due to resource limitation of the SoC.

Routing Algorithms can be classified into Deterministic and Adaptive or into Table-driven and Source routed. Deterministic routing is fixed regardless network traffic state while Adaptive routing can change according to the network traffic. In Table-driven, the route is determined using a table in each router that determines the outgoing port based on the destination address of the packet. On the other hand, Source routed depends only on the destination address and the location of current router to determine the path. If there are multiple paths available from source to destination, an efficient routing algorithm, regardless the given traffic, balances the network load across the links uniformly.

One of the most commonly used routing algorithm is Deterministic Source routed algorithm called XY algorithm [13] used with two-dimension networks, in which, the

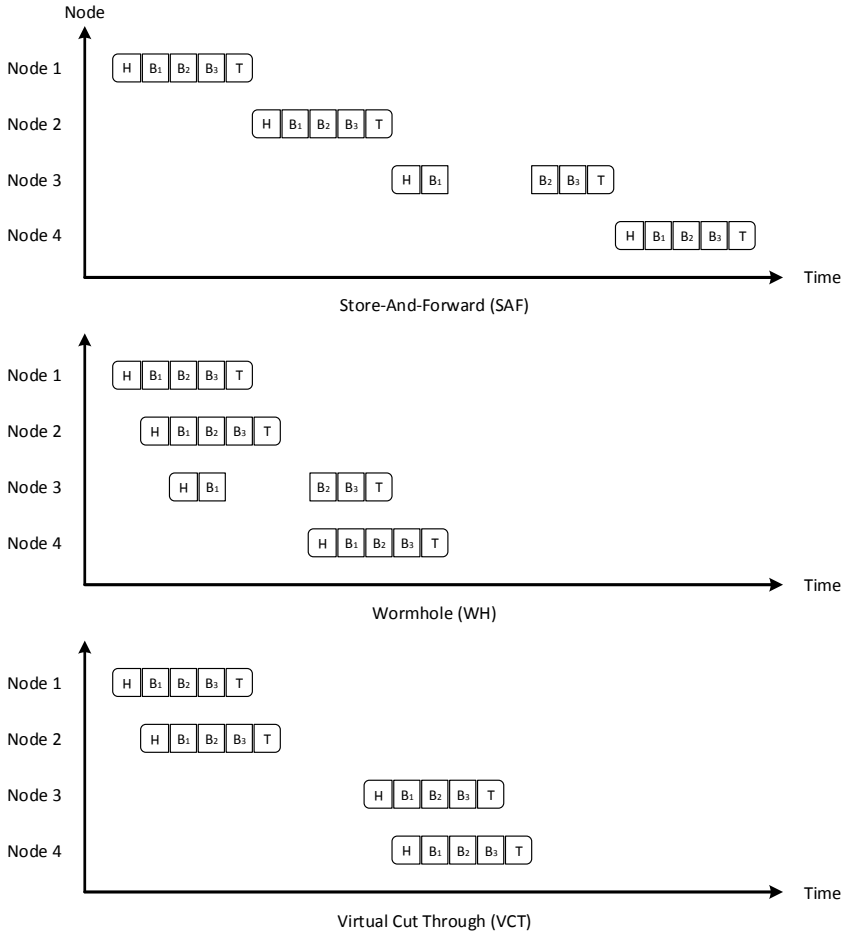


Figure 2.5: Store-And-Forward (SAF), Wormhole (WH) and Virtual Cut Through (VCT): The figure shows an example of packets traverse through four nods using SAF, WH, and VCT switching technique. The example assumes that the third node will have only two flit space in its buffers for three cycles.

packet is routed along x-axis first then along y-axis to reach the destination. XY algorithm provides high throughput with low resource usage and deadlock free network. In this thesis, we will use XY algorithm as the default algorithm in implementing networks.

2.2.1.6 Buffer Size

Buffers main function is to control the flow of packets by buffering it until other resources are available. It also represents the major area consumer in the network. Buffer Size is a function of packet size, flit size and switching technique and it has a significant effect on network latency and throughput.

2.2.1.7 Link Width

It divides the flit into a one or more phits (physical units) and directly affect the network bandwidth and the buffer size.

2.2.1.8 Arbitration

When the router receives multiple requests on some resource it must grant one of these requests, so it uses arbiters to do so. The arbiters are either static or dynamic. In static arbiter, the priority is fixed, while in dynamic arbiter, the priority changes at run time. Although dynamic arbiters are more complicated and requires more logic circuits, they are usually used because they adapt to network conditions. Many dynamic arbitration techniques are available such as First Come First Served and Priority Based arbiter, but Round Robin arbiter is the most used one because it guarantee the fairness among all requests. Further details will be presented in the next chapters.

2.3 FPGA platform

FPGAs is fabricated, ready to be electrically programmed silicon devices. It can be used to implement almost any type of digital system. In contrary with ASICs, which depends on previously designed and optimized standard cells to implement logic design, FPGAs map logic design into Look-up tables (LUTs), function generators that can implement any arbitrarily Boolean function of a certain number of inputs, registers, I/Os and IPs connected through huge amount of wires and switches. Figure 2.6 shows an example of simple FPGA architecture.

FPGA is programmed using Hardware Description Language (HDL), such as VHDL or Verilog, and then compiled into a bit stream that program FPGA configuration SRAM, finally the compiled code is downloaded to the FPGA. FPGA compilation runs through many steps; synthesizing, mapping, resolving constraints, placement, routing, and floor planning.

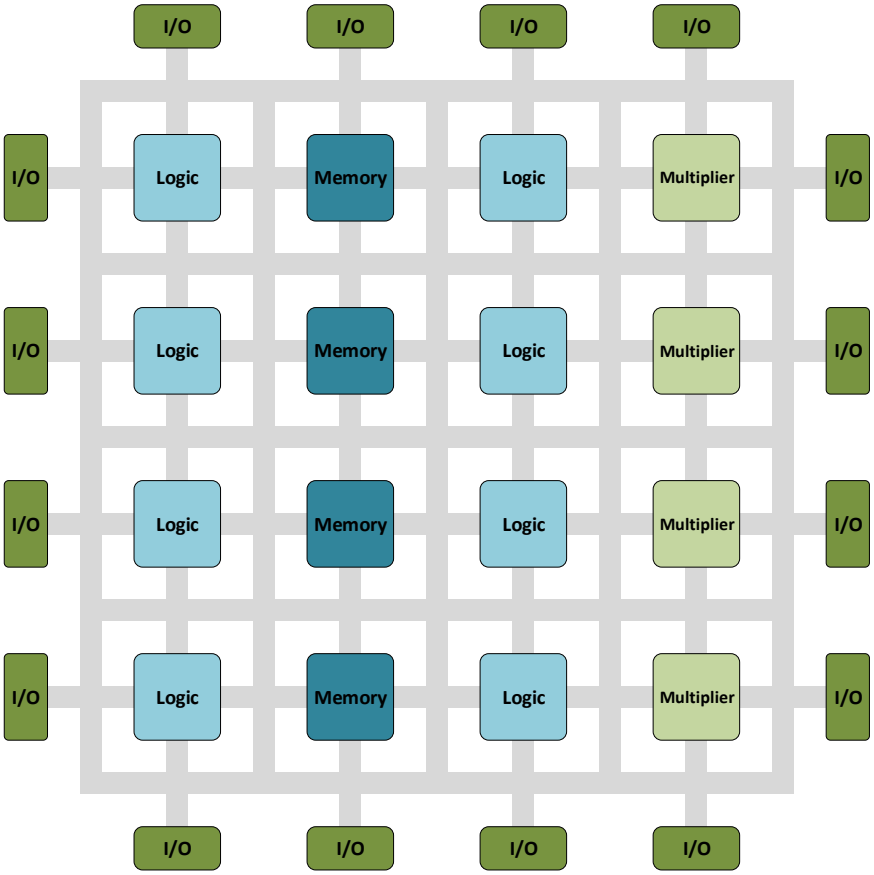


Figure 2.6: FPGA architecture example

FPGAs have many advantages over ASIC, such as:

- fast prototyping,
- low cost,
- the cost gap decreases with technology scaling ,
- less non-recurring expenses,
- smaller time to market,
- simpler design methodology,
- easy to upgrade,
- less interaction with the manufacturer, and
- dynamically reconfigurable.

Nevertheless, the most attractive feature of FPGAs is the reconfigurability, which make FPGAs more suitable for modern sophisticated application. Currently FPGAs include advanced modules such as multipliers, memories, and microprocessors, they also support partial reconfigurability.

However, FPGAs require a larger area than ASICs as the implementation of logic design using only LUT consumes area 35 times larger than ASICs and using heterogeneous FPGA blocks reduces this ratio to 25 [14]. In addition, long interconnections in FPGAs increase critical path delay, so the average ratio of the critical delay in circuits implemented by only LUTs to same circuit implemented in ASICs is 3.4, and this ratio increases to 3.5 when using heterogeneous FPGA blocks [14]. Moreover, the dynamic and static power consumption in FPGAs is larger than ASICs; for dynamic power, FPGAs consume 14 times power than ASICs when using only LUTs and from 7.1 to 14 times power when use heterogeneous blocks, and for static power, FPGAs consumes 87 times power than ASICs [14]. Finally, the use of heterogeneous blocks, like hard multipliers and memories, reduces FPGAs area and power, but relatively has a small effect on delay.

FPGAs have different architectures such as symmetrical arrays, row based, sea of gates, and hierarchical Programmable Logic Device (PLD). However, they basically consist of a group of heterogeneous programmable logic blocks, including functionally complete logic block, multiplier and memory. These programmable logic blocks are connected by a programmable routing interconnection. Moreover, the programmable logic blocks are surrounded by programmable input/output cells which connect the FPGA with the external world.

To sum up, FPGAs have many advantages, but there are many problems that prevent it from spread out. In order to overcome those problems specially the speed problem, NoC is used to replace ordinary interconnect buses.

2.4 FPGA NoC

FPGA implement flexible and reconfigurable interconnections between the relatively large numbers of its logic elements, hundreds-of-thousands in current FPGAs. In conventional FPGAs, this interconnection consists of wires of different lengths and multiplexers, these multiplexers can be programmed to combine the wires and create flexible interconnection. The same goes for the interconnection between the logic elements and any other embedded modules or I/O's.

2.4.1 Current FPGA Interconnect Problems

Programmable interconnection provide a crucial function to FPGA, however, it faces many challenges [15]:

2.4.1.1 Interconnect scaling

Beside the metal wiring interconnection, the multiplexers are implemented using pass-transistors [16] because they are used to enable very small switch. With technology scaling, metal wires and pass transistor's performance and reliability have been decreased [17]. Hence, the interconnections delay become more dominant in the critical path delay of the design to be implemented, which limits the FPGA speed.

2.4.1.2 Design hurdles

The most accurate estimation of the interconnections delay is available at the last compilation stage, the placement and routing stage, which means that if the design did not meet the minimum timing requirements, the design need to be modified and the whole compilation process need to be repeated which consumes time and effort. In addition, every time FPGA is programmed, each interconnection multiplexer need to be configured, that slows the programming process.

2.4.1.3 Bandwidth demands

In order to support the modern complicated designs, FPGA embed different types of memories, hard computing units, such as multiplier and processor cores, and fast I/O interfaces. These modules require fast and wide datapath to transfer data to/from the logic

elements and off-chip systems as well. However, accomplishing that require the use of many FPGA logic and interconnections.

2.4.1.4 Modularity

Conventional FPGA interconnections have a two level abstraction which makes it difficult to treat the required design as a group of modules that can be optimized independently. However, modularity can facilitate parallel compilation, partial reconfiguration, and optimization.

2.4.2 Embedded Networks-on-Chip Solution

Implementing NoC on FPGA is a promising solution for interconnection challenges. This can be achieved by dedicating NoC for the system-level interconnections which require a high bandwidth transfer, while the conventional interconnections will be used for the low bandwidth transfer. For more understanding, we may use the analogy with a big city transport network; the small roads are suitable for low traffic and relatively short distances, while the highway is more efficient for higher traffic and long distance requirements.

2.4.3 Network Architecture

In contrary to ASIC-based NoC, where the application is specified and the traffic density and pattern are predictable, FPGA applications are not known during the manufacturing. Therefore, we have to design the FPGA-based NoC so it can serve any application that can be configured on the FPGA.

There are three types of FPGA embedded NoC; hard, soft, and mixed NoC. Hard NoCs consist of pre-fabricated hard routers using ASIC-flow and hard links. Soft NoCs consist of soft routers, which use the programmable FPGA resources, and soft programmable links. Mixed NoCs, on the other hand, consist of hard routers and soft links.

2.4.3.1 Soft NoCs

In this architecture, we implement the NoC design, usually written in HDL, using the FPGA fabric resources, such as logic blocks and interconnections, without making any changes to the FPGA. Soft NoCs have both advantages and disadvantages, the advantages can be summarized in their re-configurability and the avoidance of FPGA architecture modification. The disadvantages are the relatively large area and power and low speed, the fundamental disadvantages of the FPGA. In soft NoC, the buffers consume the largest fraction of the area. Typically, there are three FPGA resources that can be used as a buffer:

1. Registers:
   Although there is an abundance of registers in the FPGA, the registers are separately distributed and use them as a buffer consumes a large area.

2. LUTRAM:
   LUTRAM is responsible for configuring the LUT, however, it can be used as a RAM memory.

3. BRAM:
   The embedded block RAM is dense enough to be suitable to the router buffers.

2.4.3.2 Mixed NoCs

In mixed architecture, the router is embedded as a hard module at fabrication while the FPGA interconnections are used as links, soft links. Hard router requires an interface with the interconnections, similar to the programmable multiplexer used to connect the logic elements with the interconnections. The motivation is to decrease the area occupied by the router and yet keep the flexibility of the links. The simplest way to embed the hard router is to replace the equivalent number of logic elements from the FPGA fabric with the hard router and maintain the remaining of the FPGA structure and interconnections. The hard router provides a higher speed by operating at higher frequency, however, the speed is limited by soft links. Soft links can support almost any NoC topology, while the routing algorithm can be reprogrammable to support different topologies, by using reprogrammable table driven routing algorithm. The limitation to use any arbitrary topology is the number of ports in the hard router.

2.4.3.3 Hard NoCs

Hard NoC are implemented using both hard routers and hard links. The links are dedicated wires to connect router according to predetermined topology, separated from the FPGA interconnections, therefore it save the area and work at higher speeds. Nevertheless, the routers need to interface the FPGA in the same way the interconnections connect FPGA components, through programmable multiplexers. Now that we removed the speed limitation imposed by the soft links, we can dedicate a clock network with higher frequency for the NoC. Moreover, we may disjoin the router power supply using separate power grid with a lower voltage.

In the thesis, we are targeting Soft NoCs to impose minimum changes in the current FPGA architecture, however, the ASIC implementation results are included for the purpose of comparing.

Chapter 3

NoCs in the context of ASICs and FPGAs

3.1 Introduction

NoC is a wide research area, in this chapter; we will focus on comparing the NoCs that target the FPGA (FPGA-based NoC) with the NoCs that target the ASIC (ASIC-based NoC) and investigating the NoC optimum design parameters. In addition, we propose an evaluation methodology for NoC performance considering different design parameters. Finally, we conclude with design recommendations and trade-offs for various NoC design parameters for FPGA-based and ASIC-based NoC.

A lot of research has been done on NoC. However, most of the researches are directed to ASIC-based NoC rather than FPGA-based NoC. On the other hand, since the purpose of the research that is directed to FPGA-based was prototyping, simulation, and emulation, it focused on mapping the ASIC-based NoC design to the FPGA. Which make the performance and efficiency of the mapped NoC not the first priority. However, few papers provided a study for FPGA-based NoC in which the unique hardware characteristics and generic applications of FPGAs are taken into consideration.

Designing a NoC process passes through selecting from a wide range of parameters (the topology, number of nodes, the size of queuing buffer, switching technique, etc.). Selecting the optimal parameters for ASIC-based NoC strongly influenced by the application that needs to be implemented. However, if the parameters did not achieve the minimum required performance, repeating the whole design flow is a very costly process especially after fabrication. In contrary, it is a more challenging process for FPGA-based NoC as the application is not known prior. Which forces the NoC design to be more generic and suitable for a broad space of applications. With the highest performance in mind, the selection of optimal parameters is an exhausting work.

3.2 Background

The designer can vary many parameters in NoC design. These parameters vary in their effect on overall performance. Since studying all these parameters is exhausting and inefficient, we selected four parameters that have a significant effect on NoC performance:

1. Topology: The topology represents a trade off between the average latency and the total area occupied by the NoC. A topology can provide relatively low average latency by decreasing the number of hops between nodes, at the expense of increasing the number of links between nodes. This will increase the wiring and complicate the layout, besides consuming a large portion of the area. Most NoCs designers avoid the wiring and layout problems by selecting a topology with few links such as mesh, tree, and toris.

2. Number of nodes: Increasing the number of nodes connected by the NoC increases the total throughput, but on the other hand, it consumes large area. In contrast with the ASIC-based NoCs, where the number of nodes is not difficult to deduce, the number of nodes in the FPGA-based NoCs can vary significantly. The extreme case is to associate each logic element of the FPGA with a node in the NoC, which consumes an area larger than the FPGA itself.

3. Virtual channels: Increasing the virtual channels enhances the performance, improves the wire utilization, decreases the impacts of the head-of-line blocking, and helps in solving the deadlock problems. The virtual channels are implemented by dedicating flit buffers for each virtual channel. Therefore, the number of virtual channels directly affects the buffers size and the required area.

4. Queuing buffers: Although NoC buffers organization and sizing are critical to realizing the optimal network performance, they consume the largest percentage of area and power of NoC [18]. The efficient buffer organization has a good cost-performance trade-offs. This is fulfilled by highly utilizing the buffer resources.

NoC performance evaluation involves measuring of many parameters, some of these parameters are related to the network architecture and others are related to the implementation of the network. The following section classifies the performance measures to network performance measures and implementation measures. First, network measures which include:

1. Latency: The packet latency is the difference between the injection time of the head flit into the network and the arrival time of the tail flit at the destination node [19]. The latency is measured in clock cycles in order to separate the implementation, which will reflect on the frequency of operation, from network performance. The network packets experience different latency based on the number of hops between

the source destination and the network traffic, which affect the queuing delay. Since we are concerned with the average latency, we average all the packet latencies.

2. Throughput: Throughput is the maximum traffic the network can transport [20]. The number of data units (messages, packets, or flits) per time unit (clock cycle) measures the throughput. Increasing the number of nodes will give a false indication that the network throughput is increasing; consequently, the throughput is normalized to the number of nodes. Therefore, it is measured in the number of data units per time unit per node.

3. Load-latency curves: A good network has high throughput with low latency. Therefore, in order to show both the throughput and the latency, the load-latency curve, that shows the throughput versus the latency, is used. Therefore, the maximum throughput with the lowest latency can be extracted.

Second, implementation measures which include:

1. Maximum operating frequency: It is the maximum frequency the NoC can operate with, usually imposed by the network routers. Increasing the network complexity may enhance the latency and the throughput; however, it will negatively affect the operating frequency, which degrades the overall performance. For example, a modification on the network may decrease the latency, measured in cycles, by increasing the complexity, and hence decreasing the operating frequency. After de-normalizing the latency to be measured in seconds instead of cycles (by substituting the cycle with the real time in seconds at the maximum operating frequency) we may find that the latency actually has increased and the modification was not necessary.

2. Area: It is the equivalent silicon area of the NoC. The area is strongly proportional to the power and the cost. The attempts to enhance the NoC performance by increasing the buffer size, the number of nodes, or the number of VCs will directly reflect on the area consumption.

3.3 Simulation setup

To explore the difference between the FPGA-based NoC and ASIC-based NoC, we use two NoC implementations, one designed to target the ASIC platforms and the other designed for the FPGA platforms, SOTA [18] and CONNECT [21] routers. The two selected NoCs are highly flexible and written in a fully synthesizable RTL. Which facilitates simulating various network configurations by simply varying the NoC design parameters. The two network architectures will be explained in more details in the next chapter.

In order to achieve homogeneous results, a generic network packet generator is used for SOTA and CONNECT simulation. The generator generates a uniform distribution traffic. The network performance and the synthesis results on FPGA are measured for both SOTA NoC and CONNECT NoC. Network performance includes the throughput (maximum flit injection rate) and the average latency besides the load-latency curves. Network performance results are obtained using MODELSIM 10.3c cycle-accurate RTL simulator. FPGA implementation results include FPGA resource usage percentage, the number of LUTs, and the maximum clock frequency when implemented on Xilinx Virtex-6 LX760 FPGA (part xc6vlx760, speed grade -2). The implementation results are given by Xilinx ISE 14.6 tool.

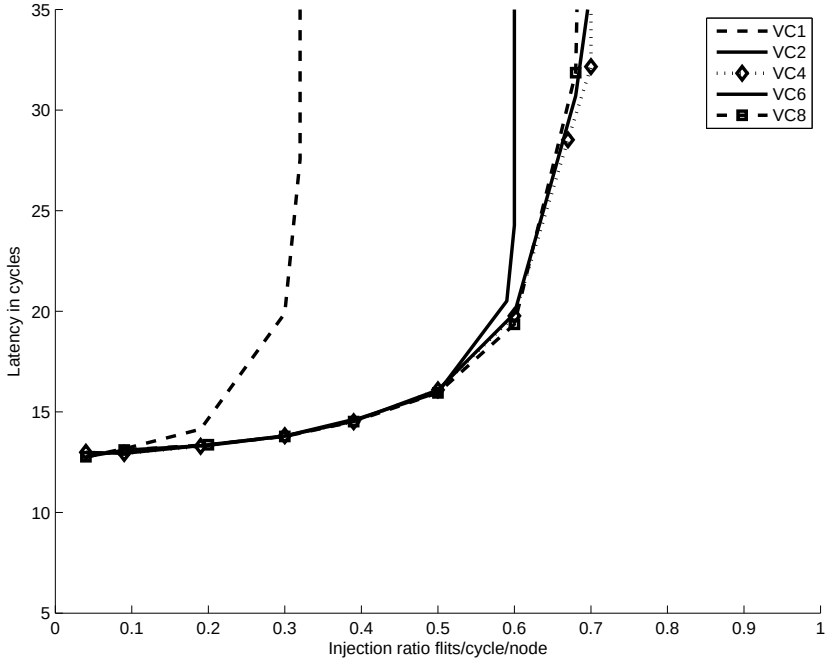
3.4 Simulation Results

This section shows and discusses the simulation results after varying SOTA and CONNECT NoCs design parameters mentioned in the last section; virtual channels, buffers depth, the number of nodes, and NoC topology.

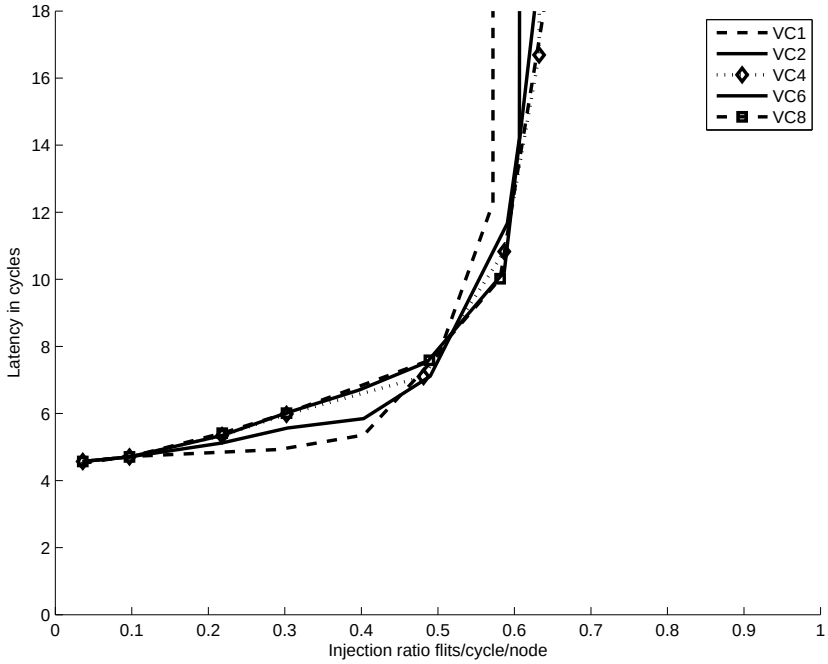
1. Varying virtual channels: Figure 3.1 shows the load-latency curves when the number of virtual channels is varied. The topology selected for simulation is MESH as it gives the best performance for the given uniform traffic pattern, will be discussed later. In addition, the selected number of nodes is 16 for the same reason. According to curves, increasing the number of VCs improves the performance, which is expected because of the benefits mentioned earlier about the increasing of VC number and its effect on throughput, head of line blocking, etc. However, increasing the number of VCs more than four VCs gives nearly the same performance besides increasing the complexity of the design. Which in turn decreases the operating frequency and increases the area. The load-latency curves after the operating frequency consideration are shown in Figure 3.2.

2. Varying buffer depth: Figure 3.3 shows the load-latency curves when the buffer depth, measured in the number of flits the buffer can hold, are varied. The curves show that increasing the buffer size improves the throughput. However, the throughput saturates at higher buffer sizes and the performance is no more improves. While the performance remains the same, increasing the buffer depth more results in unnecessary area increment. Therefore, the optimum buffer depth is measured using the following figure of merit (FOM):
$$FOM = \frac{Throughput\,per\,node}{Area\,per\,node}.$$
The FOM for different buffer sizes is shown in Figure 3.4.

3. Varying the number of nodes: Increasing the number of nodes increases the total network throughput, the throughput of each node times the number of nodes. However, the maximum throughput of each node decreases with the number of

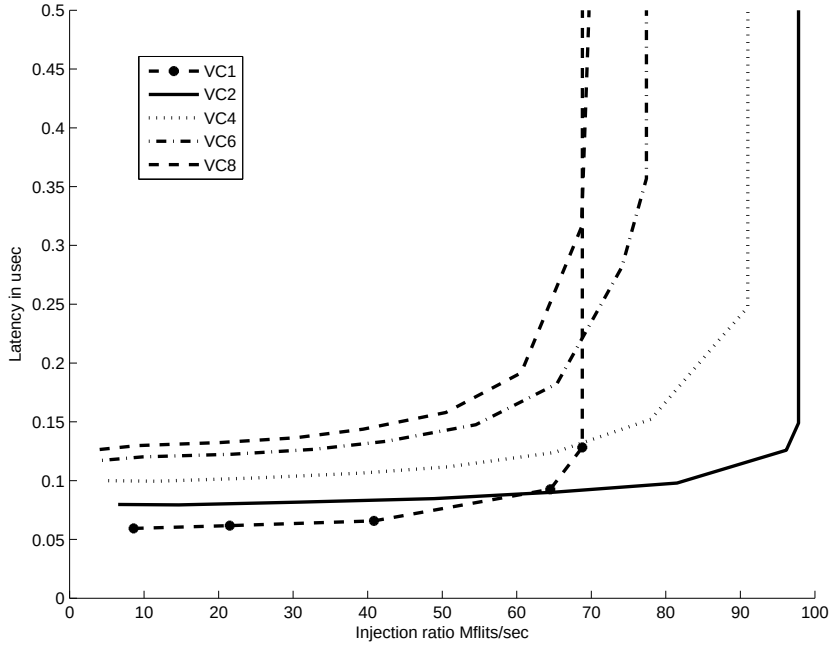


(a) SOTA MESH

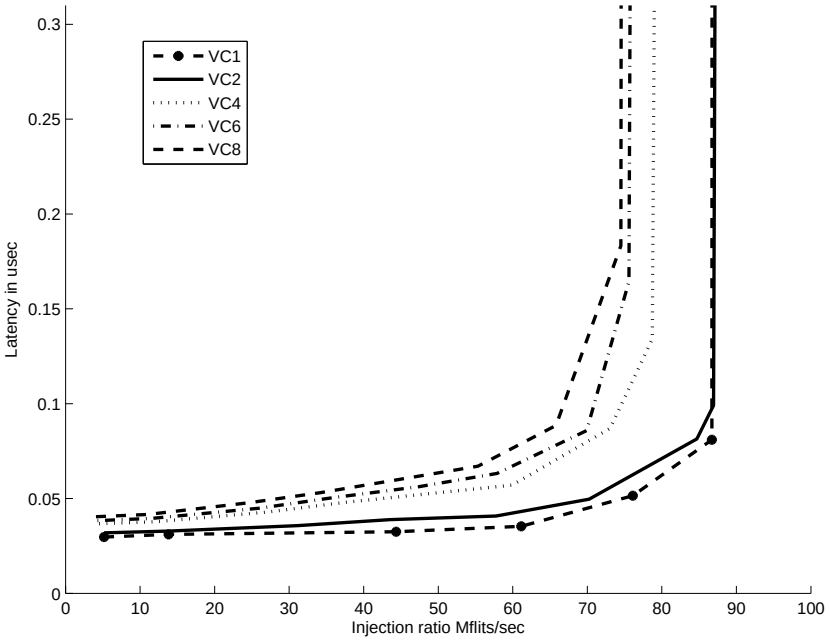


(b) CONNECT MESH

Figure 3.1: The load-latency curve when varying VC.

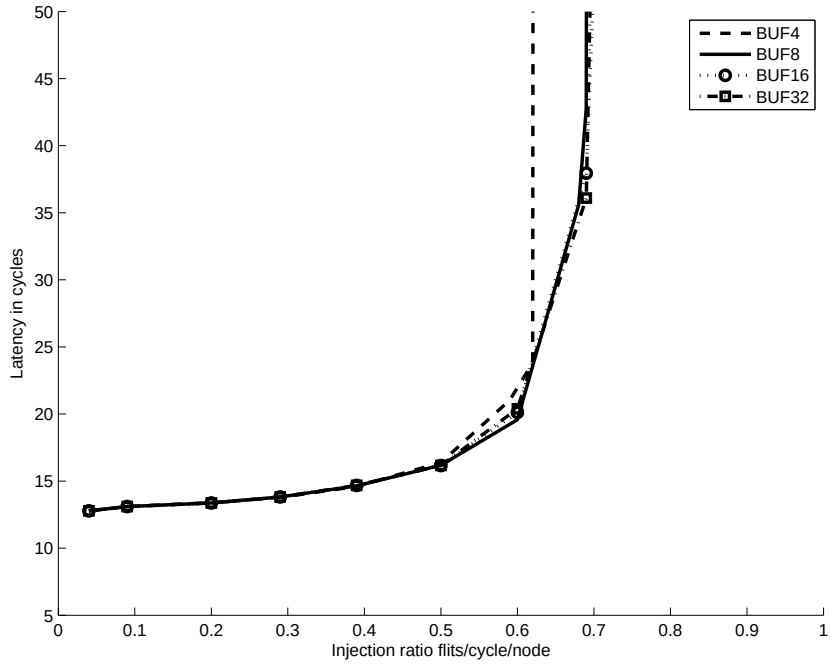


(a) SOTA MESH

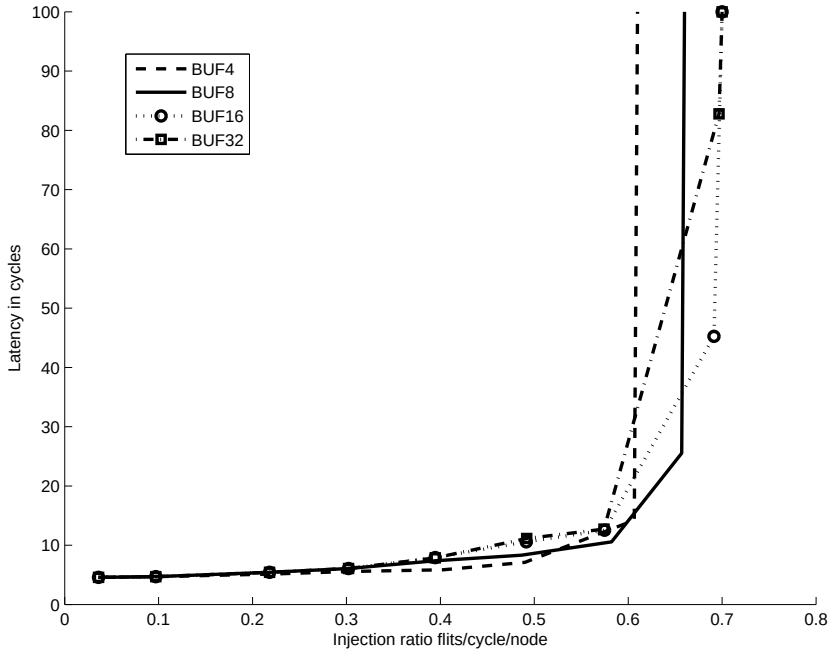


(b) CONNECT MESH

Figure 3.2: The load-latency curve when varying VC after considering the operating frequency

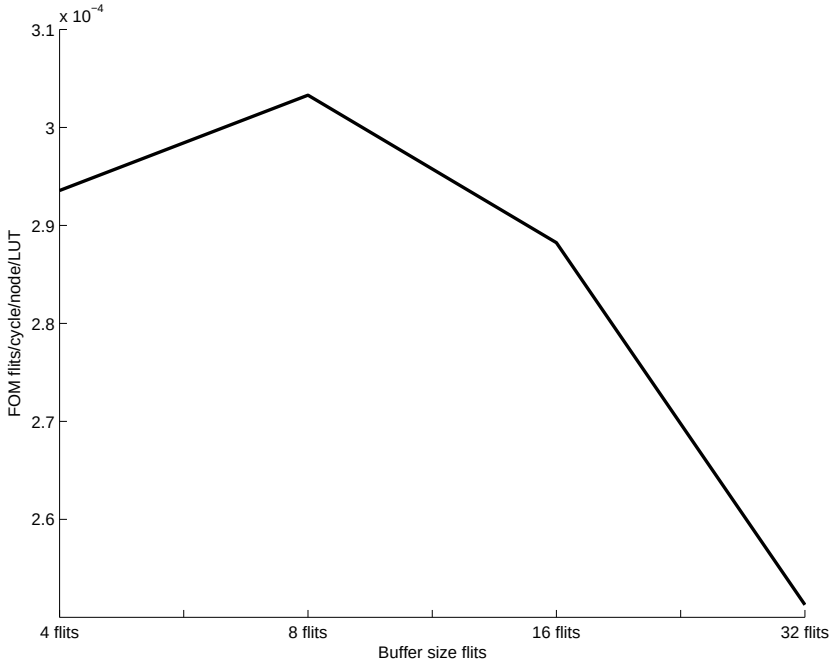


(a) SOTA MESH

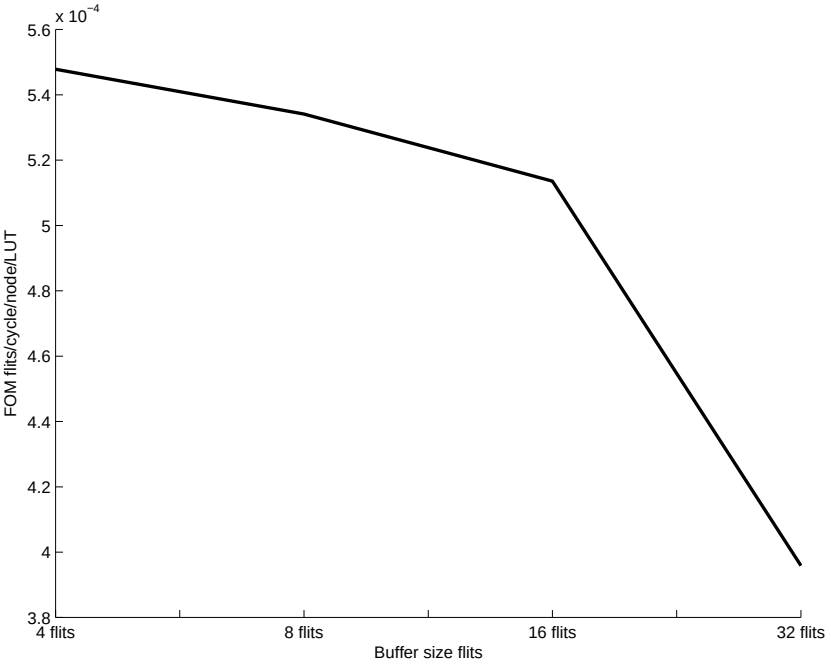


(b) CONNECT MESH

Figure 3.3: The load-latency curve when varying buffer depth



(a) SOTA MESH



(b) CONNECT MESH

Figure 3.4: The FOM for different buffer sizes

	CONNECT					SOTA				
Num. of VCs	1	2	4	6	8	1	2	4	6	8
Throughput (Flits/Cycle/node)	0.57	0.61	0.63	0.63	0.66	0.32	0.60	0.70	0.71	0.69
Operating Frequency (MHz)	152	143	125	119	113	215	163	130	109	101

Table 3.1: Num. of VCs Simulation Results

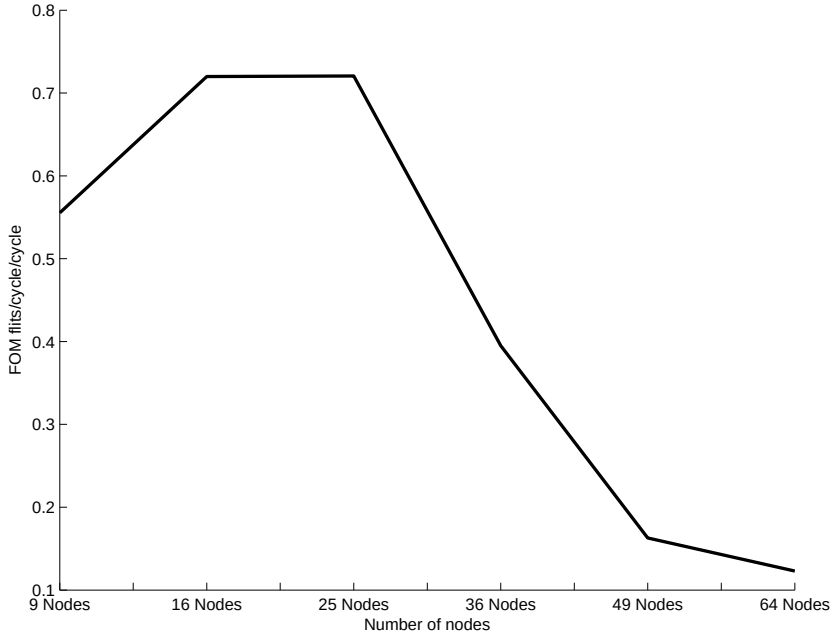
node increment. Furthermore, the latency increases because the average number of nodes the packet has to go through increases. Therefore, the optimum number of nodes is a compromise between the latency and the total network throughput. Thus, the suitable Figure of merit used to determine the number of nodes is as follows:

$$FOM = \frac{Throughput\,per\,node * number\,of\,nodes}{Average\,latency}.$$

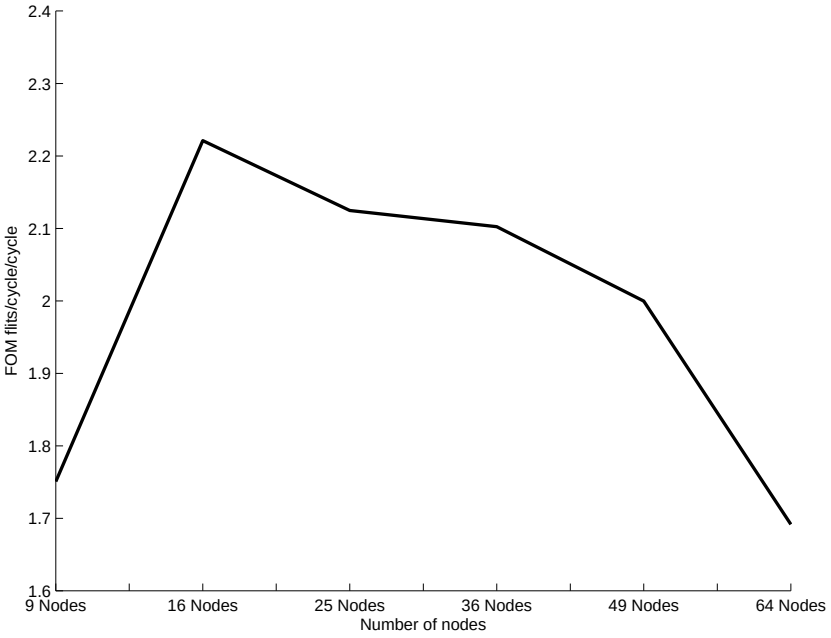
Figure 3.5 shows the FOM for different number of nodes.

4. Varying NoC topology: Some topologies are focused on enhancing the implementation properties, frequency and area, while others provide better network performance. Three promising topologies found in the literature that give the best performance are selected for comparison; MESH, TORUS, and Flattened butterfly. Flattened butterfly (FBFLY) takes three hops at maximum to deliver a flit in a 2-ary 4-flat network [22] that connects 16 nodes. Therefore, FBFLY provides the minimum latency and targets better network performance regardless the implementation. TORUS has a lower latency when compared with MESH because its diameter is smaller, the diameter is the maximum number of hops between any two nodes. The smaller diameter is achieved by connecting the end nodes, which causes the maximum wiring distance to increase to span the network dimension. Consequently, TORUS encounters a wiring problem that negatively affects the implementation and decreases the operating frequency. In order to limit the wiring problem of TORUS topology, a modified version of TORUS exists called FOLDED TORUS that limits the maximum wiring to twice the distance between two nodes. Figure 3.7 shows an example of 16-node TORUS and FOLDED TORUS networks. Although MESH faces the largest latency, it usually provides the highest operating frequency of the three topologies. To take into count the critical path induced by the wiring, the most accurate frequency of operation is measured after using Xilinx Floor-planning tool to distribute the network routers uniformly across the FPGA. Figure 3.6 shows the load-latency curves of different topologies with the operating frequency in consideration.

Tables 3.1, 3.2, 3.3 and 3.4 summarize the simulation results for SOTA and CONNECT when varying the four parameters.

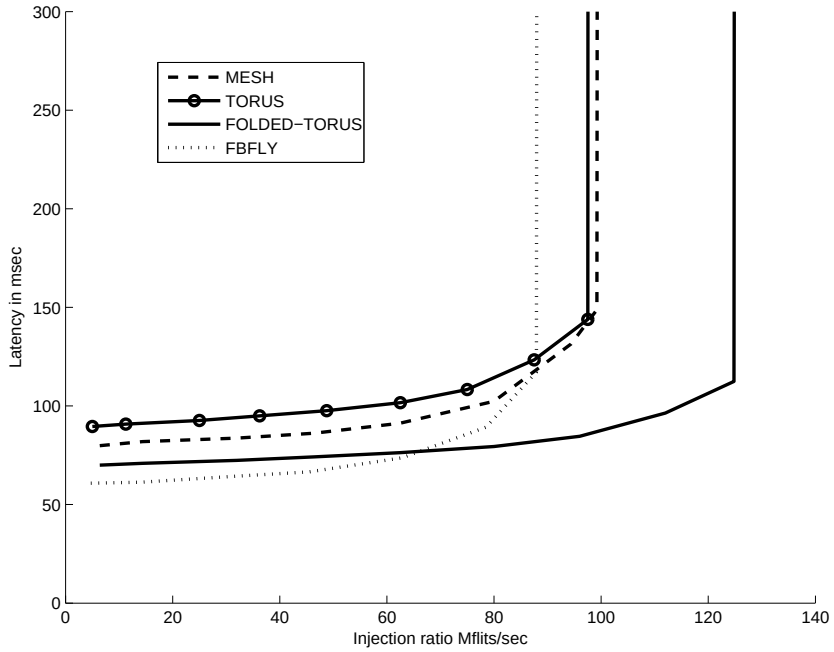


(a) SOTA MESH

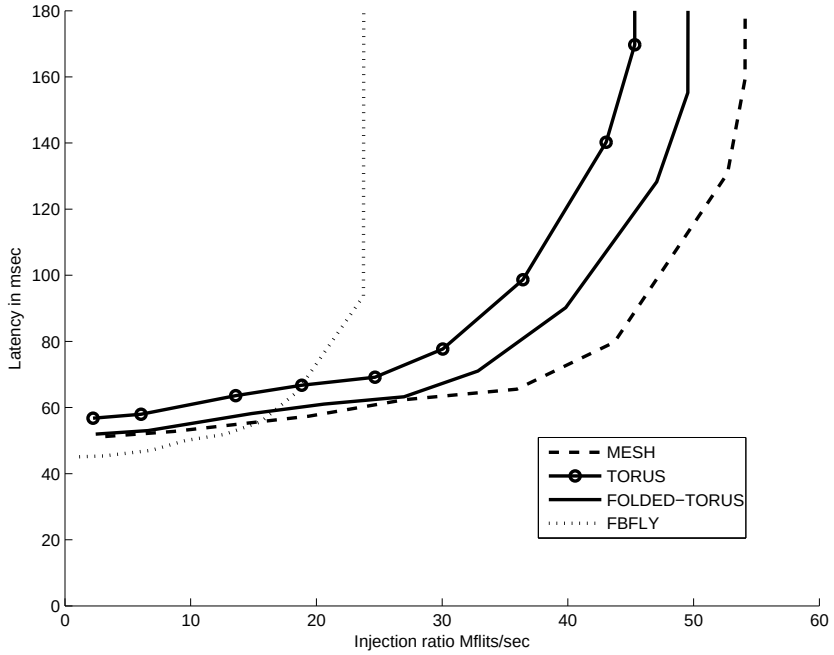


(b) CONNECT MESH

Figure 3.5: The FOM for different number of nodes

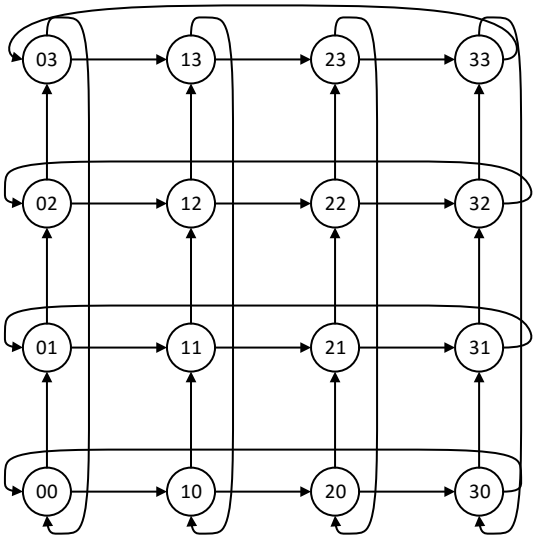


(a) SOTA MESH

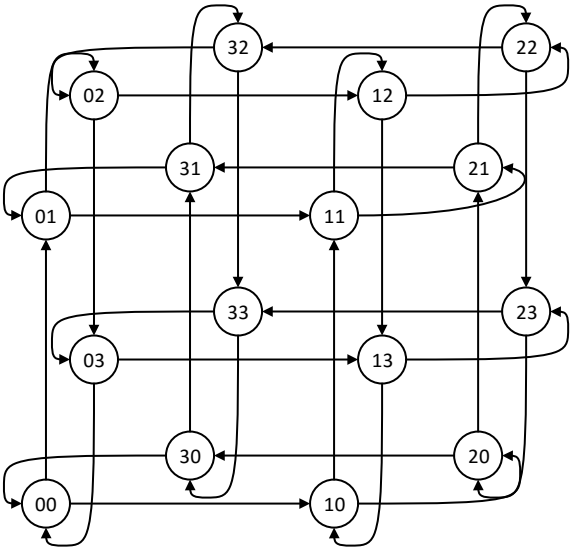


(b) CONNECT MESH

Figure 3.6: The load-latency curve for different topologies.



a) TORUS



b) FOLDED TORUS

Figure 3.7: An example of 16-node TORUS and FOLDED TORUS networks

	CONNECT					
Num. of Nodes	9	16	25	36	49	64
Throughput (Flits/Cycle/node)	0.71	0.60	0.50	0.43	0.38	0.33
Avg. Latency (Cycles)	3.63	4.36	5.87	7.38	9.23	12.56
	SOTA					
Num. of Nodes	9	16	25	36	49	64
Throughput (Flits/Cycle/node)	0.71	0.62	0.52	0.21	0.15	0.10
Avg. Latency (Cycles)	11.50	13.78	18.04	19.14	45.14	52.08

Table 3.2: Num. of Nodes Simulation Results

	CONNECT				SOTA			
Buffer Depth (Flits)	4	8	16	32	4	8	16	32
Throughput (Flits/Cycle/node)	0.61	0.66	0.69	0.70	0.62	0.69	0.72	0.73
Num. of LUTs	1107	1230	1346	1760	2112	2275	2498	2905

Table 3.3: Buffer Depth Simulation Results

	CONNECT				SOTA			
Topology	FBfly	Mesh	Torus	Folded	FBfly	Mesh	Torus	Folded
Throughput (Flits/Cycle/node)	0.37	0.61	0.73	0.73	0.56	0.62	0.78	0.78
Operating Freq. (MHz)	63	89	62	68	157	160	125	160

Table 3.4: Topology Simulation Results

3.5 Design Recommendations

After exploring the results of the preceding section, we propose design recommendations concerning the selection of ASIC-based and FPGA-based NoC parameters.

1. Virtual channels: Based on SOTA architecture results, a single VC is an optimal choice for applications that require low throughput, injection rate less than 60 Mflits/sec, as it provides the lowest delay, the smallest area, and the highest operating frequency because its low complexity. A 2-VC NoC gives the maximum throughput when measured in Mflits/sec for applications that require a high injection rate because it compromises between a high operating frequency and high network performance.For CONNECT, 1-VC NoC are treated differently from multiple VC NoCs. It's implemented using separate router, called IQ router, to avoid any overheads due to VC usage. Therefore, the performance of 1-VC and 2-VC NoC is almost the same. However, the 2-VC NoC saturates at higher throughput. Thus, a 2-VC NoC is suitable for both SOTA and CONNECT and will be used in all subsequent simulations. It is worth to note that, for larger network sizes, larger that 16-node network, it might be beneficial to use a greater number of VCs; however, it is completely useless in our case, 16-node network.

2. Buffer depth: The simulation results show that the effective buffer should be at least greater than one packet size. Thus, for CONNECT the optimum buffer size is 4 flits, can hold two packets, and for SOTA, the optimum buffer size is 8 flits, can hold four packets. The switching technique used is Wormhole, in which, if the head flit is temporally blocked, the remaining packet flit will follow it and accumulate in the blocking router buffer, therefore, the buffer should be able to hold more new coming flit and its depth should be greater than one packet. Using the recommended buffer sizes gives the best network performance without exaggerated area overheads.

3. Number of nodes: For the FPGA-based NoC CONNECT, a NoC of size 16 nodes gives the optimum performance. While for the ASIC-based NoC SOTA, both 16 node network and 25-node network gives the same performance, but with nearly the doubled area for 25 node network. Therefore, a 16 node network is suitable for both SOTA and CONNECT. Increasing the number of nodes more than 16 nodes will cause a congestion for each node traffic besides using an excessive area overhead.

4. Topology: FBFLY provides the lowest latency among the three topologies, FBFLY, MESH, and TORUS. However, it saturates at the lowest injection rate, 87.92 Mflits/sec for SOTA and 45.88 Mflits/sec for CONNECT, which make it more suitable for the applications that require low throughput. TORUS provides better network performance measured in clock cycles when compared with MESH; however, it operates at less frequency. FOLDED TORUS gives a similar frequency to MESH for SOTA router while it gives a lower frequency for CONNECT router, and lower

throughput measured in flits/sec. The reason behind this is CONNECT router is not pipelined, CONNECT recommended avoiding pipelining for FPGA-based NoC, that make the wire delay be added the router delay. That is not the case for SOTA as the router is pipelined and the dominant delay in MESH and FOLDED TORUS is the router itself, not the wire. Since we are targeting FPGA-based NoC, we will use MESH as the network topology for all the following simulations.

Chapter 4

Proposed Router

4.1 Introduction

NoCs that are designed and optimized to target ASIC do not give the optimum performance when mapped onto FPGAs. This is because of the differences in FPGA structure, such as the high register-to-logic ratio and the large delays of the configurable interconnect. A few research provides FPGA-oriented NoC design study, that take into consideration the FPGAs hardware characteristics [7]. Examples of the FPGA-oriented NoC designs that do not map ASIC-oriented NoC are CONNECT [21], Dual-crossbar router [23] and Split-Merge-based PS NoC [24].

In this chapter, a high-performance NoC router is proposed. The router is designed to target FPGAs. Furthermore, we provide a comparison between our proposed router performance and other high-performance routers presented in the literature.

4.2 Literature Review

In this section, different NoC routers present in the literature are reviewed. Starting with a generic router that targets ASIC, then three FPGA-oriented NoC routers are reviewed.

4.2.1 SOTA

SOTA (the State-Of-The-Art) is ASIC-oriented parameterized RTL router that is implemented in Verilog hardware description language based on a modular design that is easily extensible [**SOTAthesis**, 18]. SOTA designed to be a flexible and generic virtual channel (VC) router with the support of a wide range of configurations. The architecture is pipelined into three stages: VC allocation stage, switch allocation stage, and switch traversal stage. The internal structure of SOTA is shown in Figure 4.1.

SOTA buffer organization is input-queued router that holds that packets that cannot be forwarded immediately in a FIFO buffers at the input port. These FIFO buffers are divided logically into multiple FIFO, each for a VC, to avoid deadlock and Head-of-Line

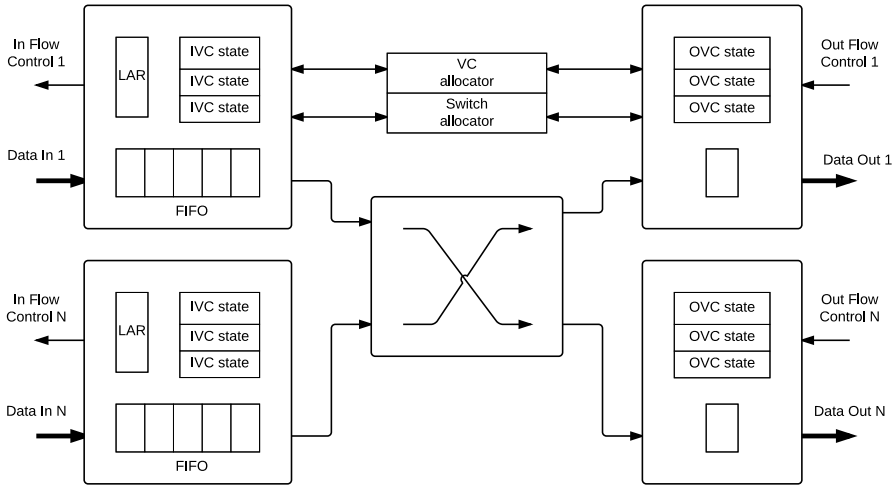


Figure 4.1: SOTA Architecture

(HoL) blocking. Since there are flits in different input VC need to be forwarded to the appropriate output VC, the router starts with route computation for the head flit using look-ahead routing (in order to minimize the pipeline delay). Thereafter, the router allocates the output VC and finally allocate the switch for flit to traverse through. In order to avoid the combinational loop the router uses wave-front allocators in VC and switch allocation. Also to improve the utilization of buffers, the router uses flexible buffer management schemes. Furthermore, it implements speculative switch allocation to parallelize VC allocation and switch allocation.

4.2.2 CONNECT

CONfigurable NEtwork Creation Tool (CONNECT) is an FPGA-based NoC generator [21]. It is implemented using Bluespec System Verilog (BSV) and generates a synthesizable RTL design. It supports various network topologies, various router parameters, and optional pipelining while minimizing the use of FPGA resources. The parameters that can vary are the number of input and output ports, the number of virtual channels (VCs), the flit width, the buffer depth, the flow control mechanisms, and the routing algorithm. The Flow Control supported is traditional credit-based flow control, and "peek" flow control. CONNECT routing algorithm is look-up tables based in which the look-up tables hold a record for each possible destination and its associated output port. Although Look-up based routing provides flexibility, look-up tables can grow for large networks. Therefore, CONNECT exploits the abundance of distributed RAMs to implement look-up tables. CONNECT implements flit buffers using distributed RAM, each VC FIFO is implemented as a circular buffer. Like SOTA, CONNECT organizes flit buffers as input-queued per virtual channel.

The internal structure of CONNECT is shown in Figure 4.2.

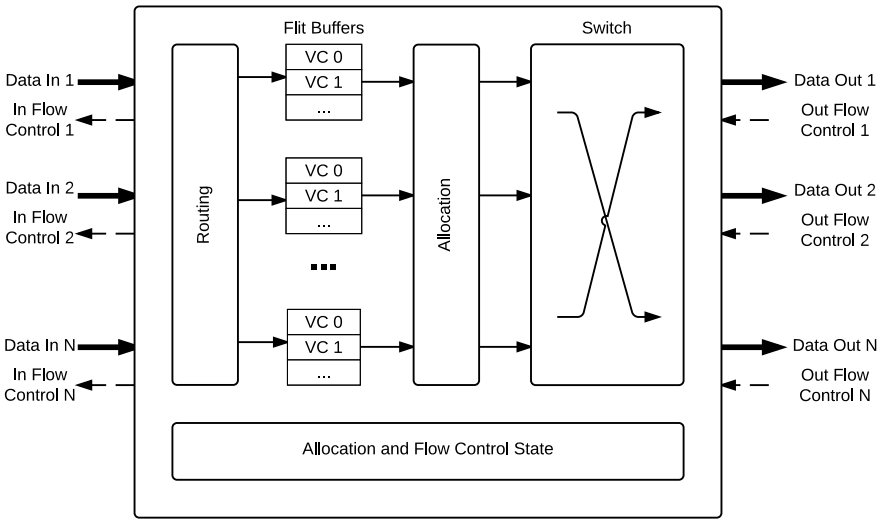


Figure 4.2: CONNECT Architecture

CONNECT considers the following in targeting FPGA:

1. Decrease the pipeline stages to a single stage to lower latency and hardware cost.
2. Using wider interfaces to tightly couple the routers, thus we can maximize wire utilization.
3. Exploit the abundance of distributed RAMs in building look-up tables and flit buffers.

4.2.3 Split-Merge

Huan proposes an FPGA optimized router in [24], which we will refer to as Split-Merge. It is based on the split and merge primitives [25]. It differs from CONNECT in two main aspects:

1. Split-Merge is pipelined which makes use of FPGA abundant flip-flops.
2. Split-Merge doesn't support virtual channels to decrease the hardware complexity.

Figure 4.3 shows the architecture of Split-Merge router. Bluespec language is used in the router implementation. The router uses backpressure flow control because of its simplicity that makes it suitable for non-virtual channel router. The buffers organization is input-queued and output-queued. Each input port has a single buffer to store flits that cannot be forwarded immediately. On the other hand, each output port has multiple buffers; a buffer corresponding to each other port's input.

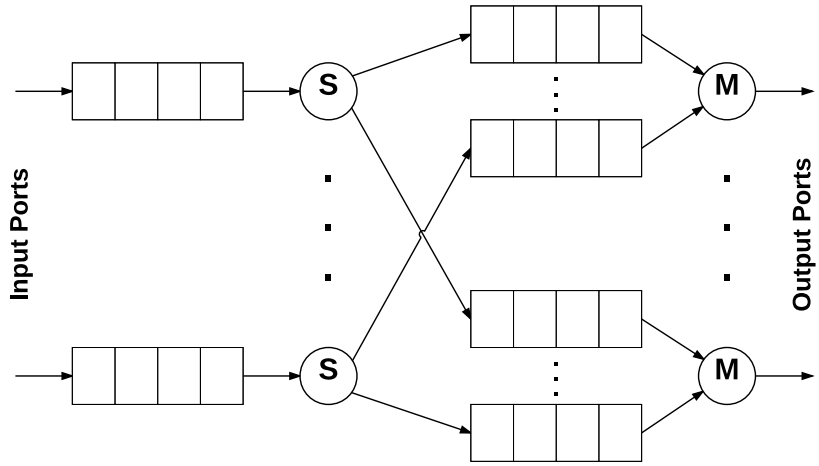


Figure 4.3: Split Merge Architecture

Basically, the router consists of two modules, split and merge. The split module reads the destination of incoming flits and splits them among the proper output buffers. Thereafter, the merge module uses an arbiter to select a flit, from the multiple buffers connected to the merge, to depart form the output port. The most advantageous feature of this organization is the elimination of the need for a crossbar.

4.2.4 Dual-crossbar

[23] proposes a configurable router that supports five network topologies and utilizes packet switching which will be refered as Dual-crossbar router for the rest of this thesis. The supported topologies are uni- and bi-directional ring, uni- and bi-directional octagon, and mesh.

Figure 4.4 shows Dual-crossbar router architecture which is based on a dual-crossbar arrangement. Dual-crossbar router consists of 5 bidirectional ports: Local, West, East, North and South and controlled by a control logic. The local port is responsible for establishing connections between the associated PE and the other four ports and providing a support for the different network topologies. Instead of using a full 5×5 crossbar to perform the switching, Dual-crossbar router uses two 3×3 crossbars to reduce power consumption and area [26]. The buffer organization used is output-queued router in which each port holds a buffered output channel with handshaking flow control. In output-queued router, the switching is performed prior to the buffering. Therefore, the inputs are isolated from the congestion at the output until the point that the output buffer is full. Dual-crossbar router uses a deterministic routing algorithm with all supported topologies. Figure 4.5

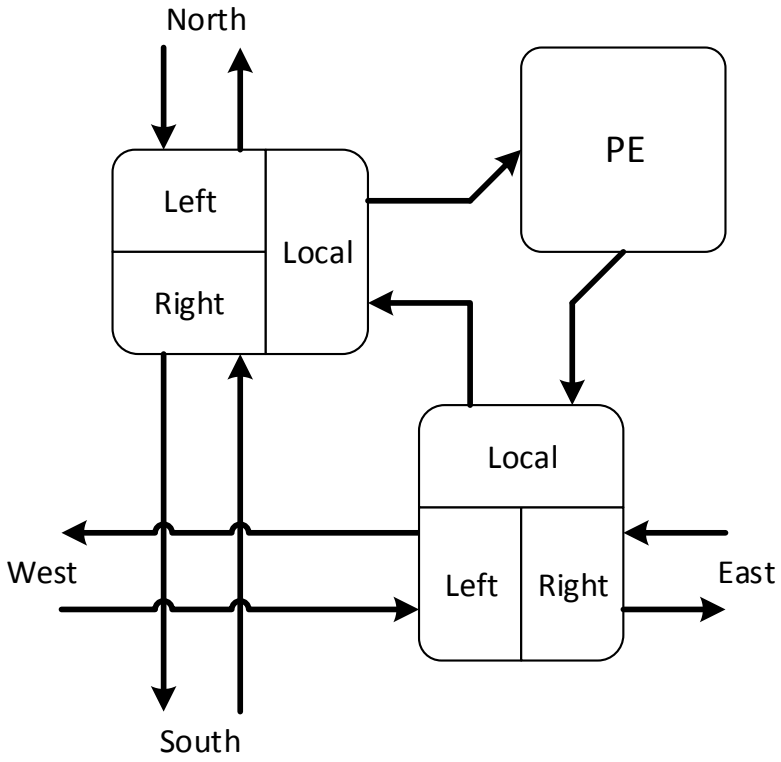


Figure 4.4: Dual-crossbar Architecture

shows the internal architecture of one 3× 3 crossbar. XY routing is suitable to the dual-crossbar arrangement; the horizontal routing is handled by the first crossbar while the vertical routing is handled by the second crossbar.

4.3 Proposed Architecture

One of the most important NoC performance measuring metrics is the maximum throughput. The throughput is usually measured in flits/s/node. In order to realize a high throughput NoC, two parts should be considered, the network part (i.e., the router network throughput measured in flits/cycle/node) and the chip part (i.e., the maximum operating frequency that will reflect on the cycle time). In this chapter, we propose a very high throughput router in which both the network part and the chip part are improved.

To achieve a high throughput NoC, first, we attempt to avoid the modules that negatively affect the router operating frequency, in other words, the significant time-consuming modules. One of these time-consuming modules is the crossbar [23]. The conventional implementation of the crossbar is using a set of multiplexers that grant connections between all possible input-output channels. One of the remarkable works to

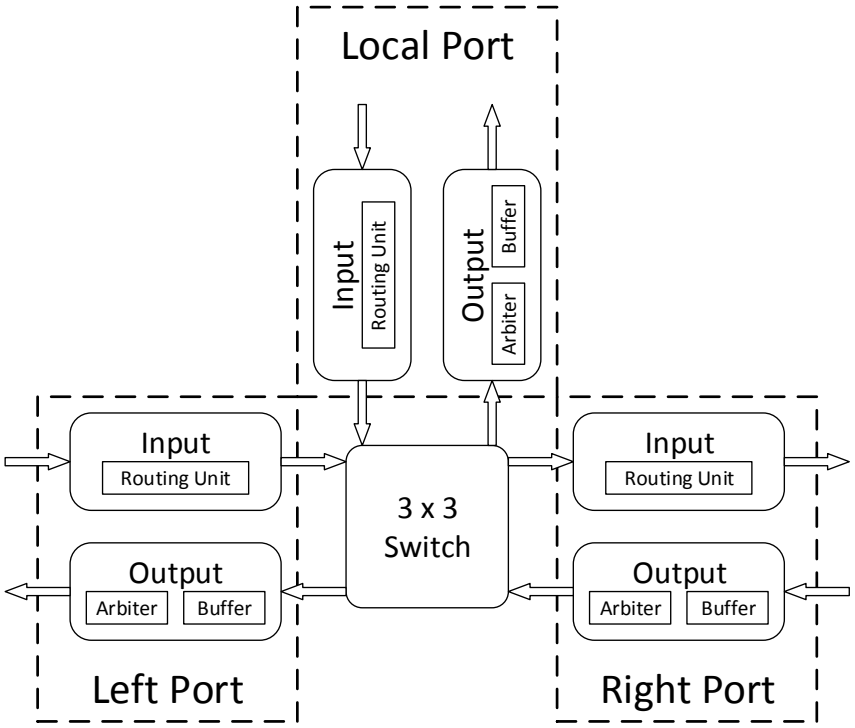


Figure 4.5: Dual-crossbar internal architecture of one 3× 3 crossbar

enhance the crossbars efficiency is the dual-crossbar router introduced in [23]. The dual-crossbar router has 5 ports in which, two 3 × 3 crossbars are used instead of using one 5 × 5 crossbar to perform the switching to decrease the area and power consumption. Although the dual-crossbar router successfully managed to reduce the average area by 22%, the operating frequency achieved is reduced to 123 MHz compared to 152 MHz of the conventional 5 × 5 crossbar router for the same platform.

However, the crossbar is completely removed in the Split-Merge router and is replaced with buffers, which enhances the operating frequency. Subsequently, Split-Merge router eliminates the need for the switch allocator. Besides being one of the frequency-limiting modules, the switch allocator also has a bad impact on the network throughput. The switch allocator poor matching quality in the separable allocator implementation causes that negative effect on the throughput [18]. Based on two previous architectures, we propose Dual-Split-Merge router (DSM) , a new router architecture that combines the advantages of both Split-Merge router and dual-crossbar router to realize a high throughput router.

The proposed router (DSM) is a 5-port router. One port is dedicated for the packets delivery from/to the processing element (PE) associated with the router (local port). The remaining four ports are used to deliver the packets to adjacent routers (East, West, North and South port). The router is designed mainly to support 2D-mesh topology and X-Y routing algorithm. It consists of two internal routers (each is a 3-port router) instead of

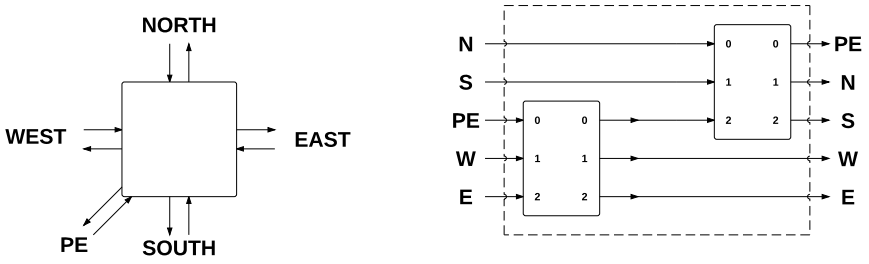


Figure 4.6: DSM router architecture

using a single 5-port router as shown in Figure 4.6. Each internal router is dedicated to handle the routing in one of the two directions, X and Y dimensions. This implies that the router handles each direction independently, which subsequently increase the network throughput. The same 5-port router functionality is done by the two internal routers, but with a higher operating frequency with only one added clock cycle at maximum for the whole routing path latency. As long as the packet travels along one direction, it uses only one of the two internal routers to path through unless a change in the direction is required, which occur one time at maximum in the pass from source to destination. Which means that the packet endures nearly half the complex logic, half the arbitration logic, and half the capacitance load when compared with the Split-Merge router.

Each internal router has three bidirectional ports; local, left, and right. The internal router function is to route the packets in only one dimension in either two ways. The local port input of the first internal router is connected to the PE output port. Moreover, in order to pass the packets from one dimension to another, the local port output of the first internal router is connected to the local port input of the second router. Furthermore, the local port output of the second internal router is connected to the PE input. The left and right ports of the first internal router are connected to West and East ports respectively. Therefore, the first internal router is responsible for routing along X direction. On the other hand, the left and right ports of the second internal router are connected to North and South ports respectively. Therefore, the second internal router is responsible for routing along Y direction. The packets to be sent from PE pass first through the local port input of the first internal router. Then the packet is routed to East/West ports until it switched eventually to the second internal router to be routed to North/South ports or to be transferred to destination PE that is associated with the second internal router.

Although the proposed router targets mainly Mesh topology, the most widely used topology, it also supports many other two-dimension topologies such as Torus, Bidir-ring, Uni-ring, Bidir-octagon and Uni-octagon. In order to manage input buffers and do handshaking with other routers, DSM router uses backpressure flow control.

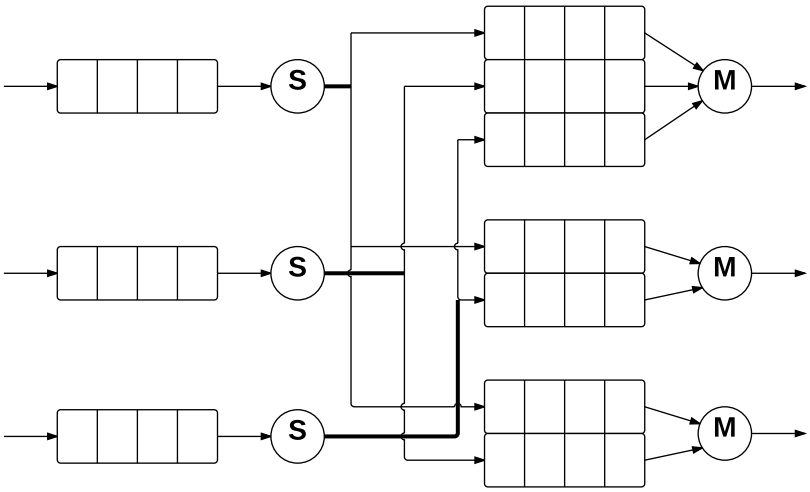


Figure 4.7: Internal router structure

Figure 4.7 shows the internal structure of the internal router which consists of two groups of modules, split modules and merge modules. The split/merge module of DSM has only two ports instead of four ports in the split/merge module of Split-Merge router. Subsequently, the performance is nearly doubled. However, there is one 3-port split/merge that can be pipelined to achieve the 2-port split/merge performance, but this would not be necessary as it is not in the critical path anymore. The internal router is pipelined into two stages; split stage and merge stage. However, the DSM router can be deeply pipelined for more increase in the maximum operating frequency by splitting each stage into two stages; a stage for the logic of split/merge and a stage for their associated buffers.

Besides increasing the network throughput and operating frequency, this architecture also uses fewer buffers (i.e., less area than the Split-Merge architecture). The conventional 5-port Split-Merge router has 5 input buffers, each for one input port, and 20 output buffers, 4 buffers for each output port, that is a total of 25 buffers. On the other hand, each internal router in the DSM has 3 ports with 3 input buffers and 7 output buffers, that is a total of 20 buffers for the DSM router. That means a 20 % reduction in the area when compared with the conventional Split-Merge.

For further increasing in the operating frequency, DSM router uses a look-ahead routing algorithm. In conventional routing, the split logic gets the required destination from the received header flit, then carries outs routing computation to determine which output port the packet should be forwarded to, thereafter the split logic splits the packet flits to the right output port. To decrease the critical path, a look-ahead routing logic in the previous router performs the routing computation for its next router so that split logic and routing computation can be performed in parallel. In DSM, the look-ahead routing is carried out

per internal router to decide for a packet what the output port in the next internal router is and either the next in internal router belongs to the same router or to the following router.

4.4 DSM router with Virtual Channels

Using virtual channels is one of the ways to boost network throughput. The idea of virtual channels is to create multiple virtual paths on the same physical channel, which requires including a buffer for each path. A closer look at the architecture of Split-Merge shows that it demonstrations the virtual output queuing architecture (VOQ) which is an implementation way of virtual channels [27]. In VOQ, each input port contains a dedicated virtual channel (i.e. a buffer) for each output port. In contrary, in Split-Merge, each output port accommodates a buffer for each input port. Which increases the network throughput; however, it does not remove the head-of-line (HOL) effect, as there is only one buffer in each input port. This single buffer implies that there are no multiple paths for the different packets, which prevents the proper implementation of virtual channels. For instance, if the head of the input buffer is occupied by a flit that is routed to a full output buffer, this filt will block all the subsequent flits in the same input buffer even if they are going to different output buffers.

In order to remove the HOL effect, we propose to support the input buffer with virtual channels (i.e., use multiple buffers per input port). Furthermore, we propose augmenting output port with more buffers for more increase in the throughput; use multiple dedicated buffers in the output port for each input port instead of using a single buffer. The number of these multiple buffers equals to the multiple buffers in the input port to implement dedicated multiple paths.

Figure 4.8 shows the internal structure of DSM router ports when the virtual channel mechanism is added. Each input port consists of multiple buffers (each for a virtual channel), a demultiplexer to direct the incoming flits to the appropriate buffer, and a modified split unit that supports virtual channels (SplitVC). SplitVC unit consists of an arbiter to select between the different VCs followed by a split unit to do the splitting function. In order to avoid any blocking, SplitVC can switch from any blocked packet to another while sending a label (one hot key) to indicate the VC of outgoing flits. The output port consists of a number of queuing buffers equal to the number of ports (three in this case) times the number of VCs, followed by a modified merge unit (MergeVC) that supports virtual channels. MergeVC consists of an arbiter (relatively large arbiter) followed by a merge unit. In order to eliminate the long delay induced by the required large arbitration process, the arbitration is done in two stages, first arbitration between the VCs followed by arbitration between the ports.

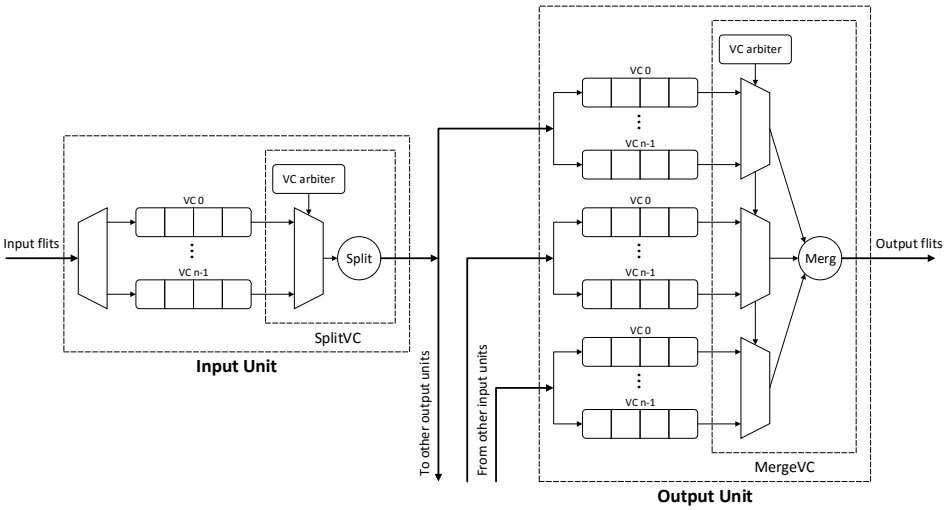


Figure 4.8: The internal structure of DSM router ports with virtual channels support

Implementing VCs in DSM router enhanced the throughput of the network significantly, as we will see in the simulation results, however, the straightforward implementation of VCs has two main drawbacks. First, the buffer area is almost double while doubling the number of VCs, i.e. the area is linearly increasing with the increment in VC number. Second, the operating frequency decreases with the increment in VC number because of the arbiter unit used by the VC control logic. These drawbacks are handled and discussed in subsequent chapters.

4.5 Network Interface

One of the most critical modules that directly impact the NoC performance is the network interface (NI) [28]. It is important to design generic, ready to run and yet highly efficient NI to transfer the data between the routers and different PEs.

We propose a simple NI architecture, that provides high throughput, simple design, and flexible interface. The proposed NI consists of two parts: Network to PE part and PE to network part as Figure 4.9 shows.

4.5.1 Network to PE part:

This part is responsible for delivering the data carried by network flits to the PE. This part consists of a buffer to hold the data received from the network until PE is ready to absorb the data, and a controller to handle received flit and manage PE requests. When a head flit arrives, the controller first checks the header information to make sure the flit is routed to the correct destination. Second, a counter is instantiated with the packet length,

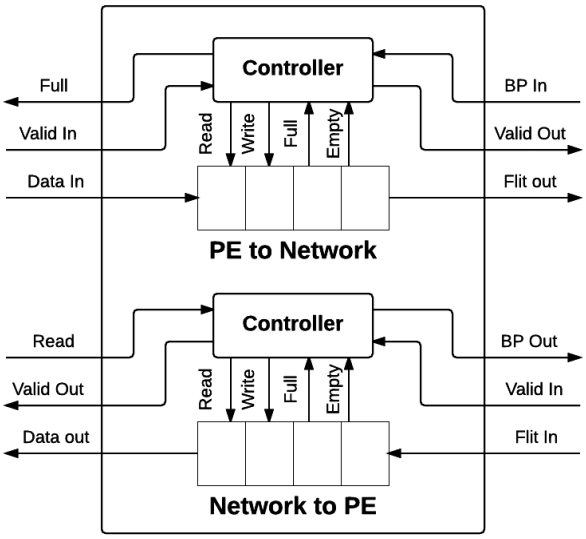


Figure 4.9: Network Interface Structure

then the data is directed immediately to PE if it is ready to receive or to the buffer. Here, the controller forward any received flit immediately without waiting for the whole packet to be received.

4.5.2 PE to Network part:

This part is responsible for encapsulating the data received from PE into packets and deliver it to the Network. This part also consists of a buffer and a controller. Whenever there is data available in the buffer or at the PE input port, the controller encapsulates the data into a group of flits that belong to a packet, taking into consideration the maximum number of flit per packets that NoC configured to work on. This does not mean that the controller has to wait until the maximum number of flit per packets are received, instead, the controller will try to send the maximum number of flit, to decrease the overhead, or it will send the available flits if PE temporarily stops sending, to increase the throughput. The proposed NI is fully parameterized and can handle any arbitrary data width on both sides; the network side and PE side.

4.6 Results

This section shows the DSM router simulation and implementation results.

4.6.1 Router Results

The results are compared with the other routers presented in literature; SOTA, CONNECT, and Split-Merge. Network performance results are obtained using MODELSIM

10.3c cycle-accurate RTL simulator. FPGA implementation results include FPGA resource usage percentage, the number of LUTs, and the maximum clock frequency. However, the implementation is performed over two FPGA platforms. First, Xilinx Virtex-6 LX760 FPGA (part xc6vlx75, speed grade -3) because it is close to the technology used in recent NoC researches, both CONNECT and Split-Merge used Xilinx Virtex-6 in their publication implementation results. Second, Virtex-5 LX330T FPGA (part xc5vlx330t, speed grade -2) which is used to compare FPGA results with ASIC results as it is close to the technology used in ASIC (65nm technology). ASIC results are based on synthesizing over a UMC 65nm Typical standard-cell fabrication technology. The implementation results are given by Xilinx ISE 14.6 tool and Synopsys Design Compiler for FPGA and ASIC results respectively. Our proposed router DSM and SOTA router are written in Verilog, however, CONNECT and Split-Merge are available in Bluespec language.

To ensure the fairness of comparison, the NoC parameters are unified for all the routers. The unified parameters are selected based on the design recommendations presented in [7]. These parameters are a 2D-Mesh network topology, 16 node network size, 32-bit flit width, and 32-flit buffer depth with 8-flit packet length. In addition, the packet generator for all routers is the same.

Two versions of the DSM router are implemented, 2-stage pipeline and 4-stage pipeline, with various numbers of virtual channels.

4.6.1.1 Network Performance Results

Figure 4.10 shows DSM network throughput in flits/cycle/node. It shows that with the increase of the number of pipeline stages the throughput almost does not change. However, with the increase in the number of VCs, the throughput increases until it saturates at a number of VCs larger than 8-VC.

Figure 4.11 and Figure 4.12 show the load-latency curves of DSM, with 2-stage pipeline and 4-stage pipeline respectively, for different configurations in which the simulation results of the average network latency (in cycles) at different injection loads are shown beside the maximum router throughput (in flits/cycle/node). The figure shows that DSM with 1-VC gives the lowest latency, however , it saturates at low throughput.

4.6.1.2 Virtex-6 FPGA Results

Figure 4.13 shows DSM operating frequency in GHz, while Figure 4.14 shows the router area in LUTs for different network configurations.

Figure 4.13 shows that with the increase of the number of pipeline stages the operating frequency increases and with the increase of the number of VCs it decreases. Figure 4.14 shows that with each increment in the number of VCs the area is almost double, however,

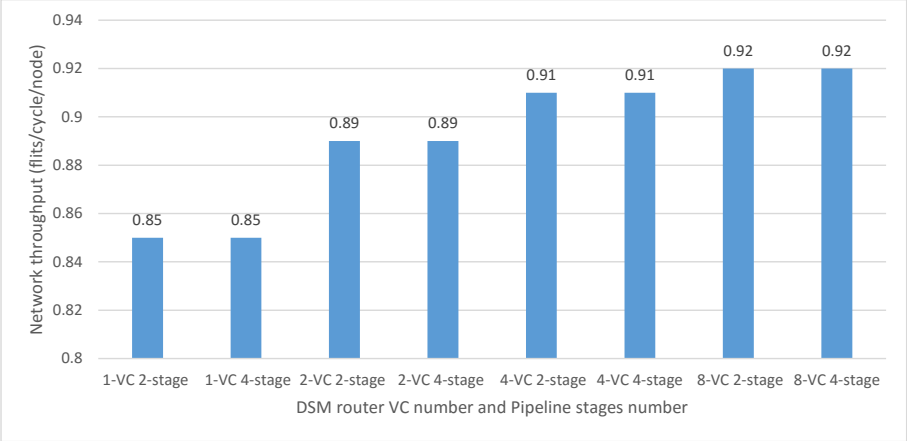


Figure 4.10: DSM network throughput in flits/cycle/node

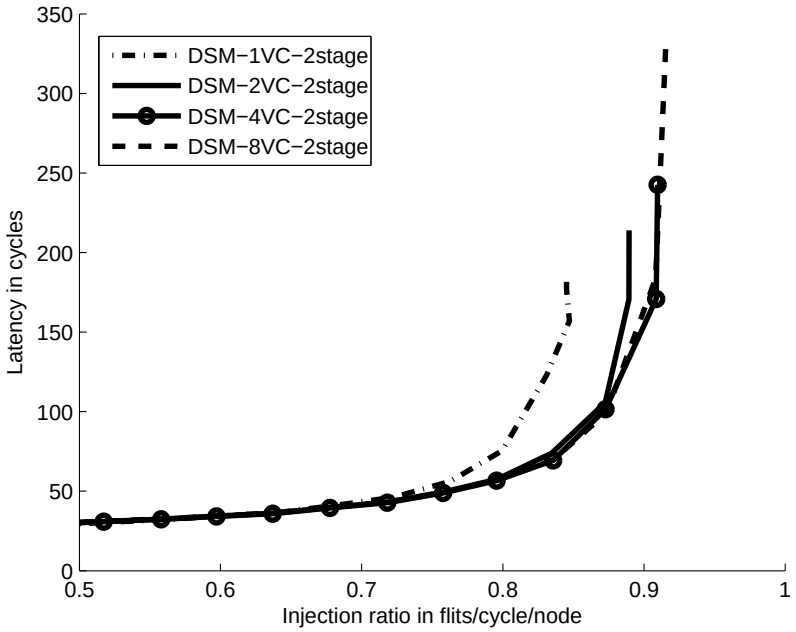


Figure 4.11: Load-Latency curves of different configurations of DSM with 2-stage pipeline

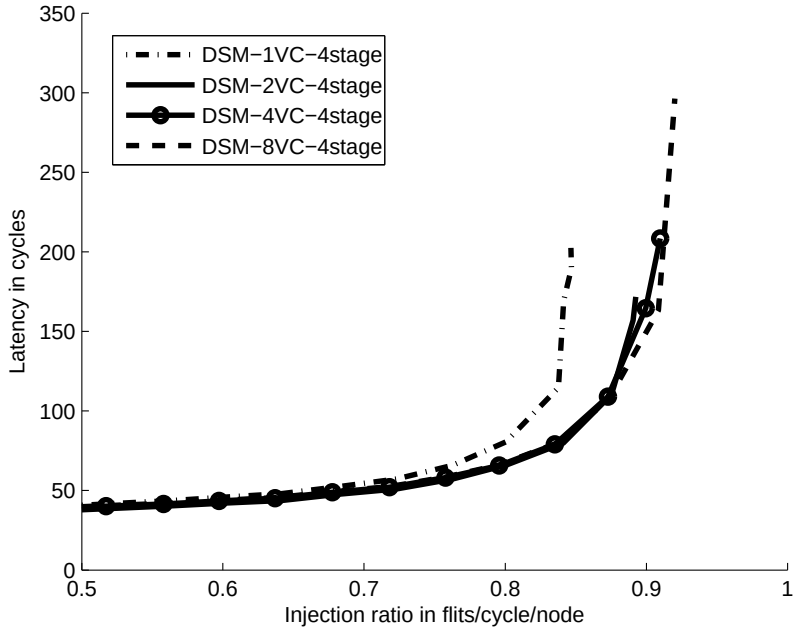


Figure 4.12: Load-Latency curves of different configurations of DSM with 4-stage pipeline

with the increase of pipeline stages it slightly increases. From the figures 4.10, 4.13 and 4.14 , we can conclude that increasing the number of VCs from 1-VC to 2-VC results in a high increase in the network throughput and small decrease in the operating frequency while for the other number of VCs there is an insignificant increase in the network throughput.

Figure 4.15 shows the load-latency curves of DSM in which the average network latency at different injection loads are shown beside the maximum router throughput after taking the operating frequency into consideration. DSM maximum throughput is 389 Mflits/s/node for 2-VC 4-stage pipeline configuration, that is corresponding to 12.4 Gbits/s/node for a flit width of 32-bit, which can be boosted to 50 Gbits/s/node by increasing the flit width to 128-bit flit which will not affect the maximum operating frequency. Also, the curves show that the number of virtual channels of one or two is optimal as discussed in [7].

The comparison between DSM router and the other routers presented in the literature is made with a number of VCs of 2-VC for all routers other than Split-Merge as the virtual channels is not supported in it. We also have selected the 4-stage pipeline 1-VC and 2-VC configurations of DSM for the comparison that is because although the 2-VC DSM gives the highest throughput, 1-VC DSM gives almost the same performance with a much smaller area. The maximum throughput, measured in Mflits/s/node, and the area,

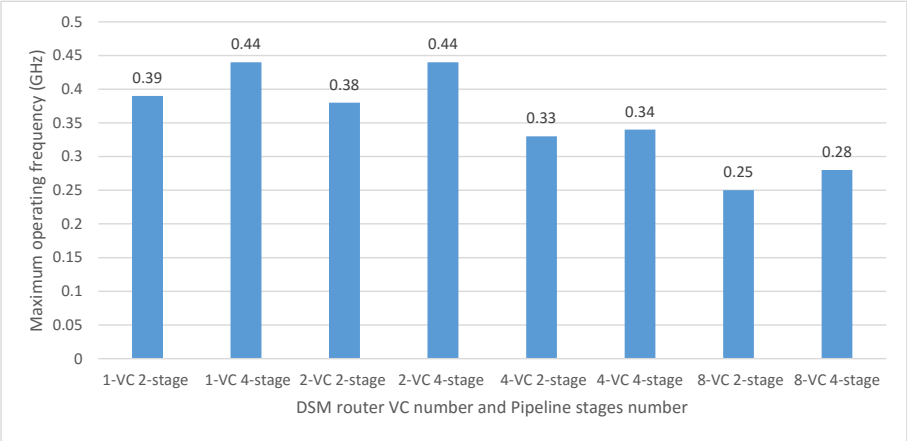


Figure 4.13: The operating frequency of DSM in GHz (based on Virtex-6 platform)

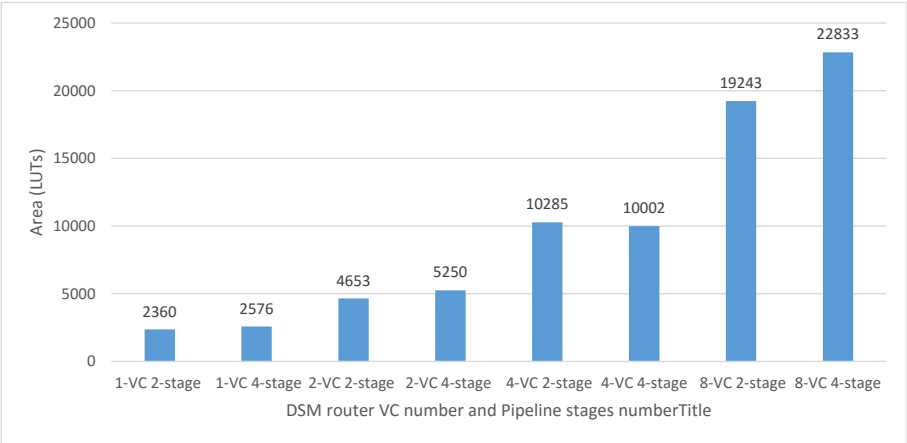


Figure 4.14: The area in LUTs of DSM (based on Virtex-6 platform)

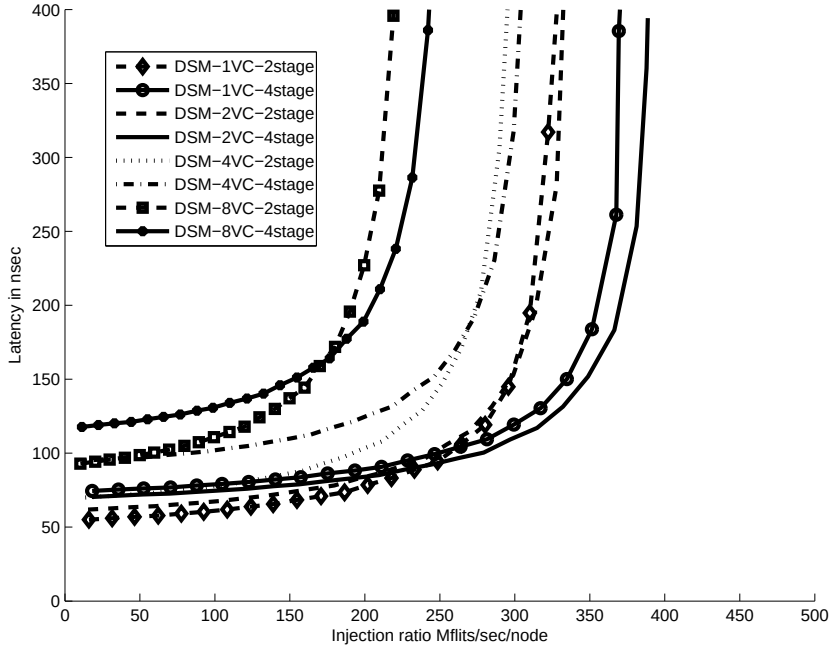


Figure 4.15: Load-Latency curves of different configurations of DSM (based on Virtex-6 platform)

measured in LUTs, for different routers are shown in Figure 4.16. Table 4.1 shows a comparison between DSM and the other routers in terms of network throughput, router maximum operating frequency, router throughput, and router area.

Fig.4.17 shows the different router load-latency curves. In order to take into consideration the operating frequency effect on the overall NoC performance, the latency is measured in nsec and the injection ratio is measured in MFlits/s/node. The comparison emphasizes the significant improvements of DSM router in term of throughput and latency. The DSM 1-VC throughput is 349%, 177%, and 67% higher than CONNECT, SOTA, and Split-Merge routers respectively while occupying almost the same area. The significant improvements of the throughput are due to the high maximum operating frequency achieved, 440 MHz, which is way beyond the other routers operating frequency. The network throughput enhanced, 0.85 flits/cycle/node, by handling each dimension separately and eliminating the switch allocation. The DSM 2-VC throughput is quite higher but needs more area because of the additional buffers.

4.6.1.3 Virtex-5 FPGA and UMC ASIC Results

Figure 4.18 shows DSM operating frequency in GHz for Virtex-5 FPGA and UMC ASIC platforms, while Figure 4.19 shows the router area (in LUTs for Vertix-5 and in μm^2) for different network configurations.

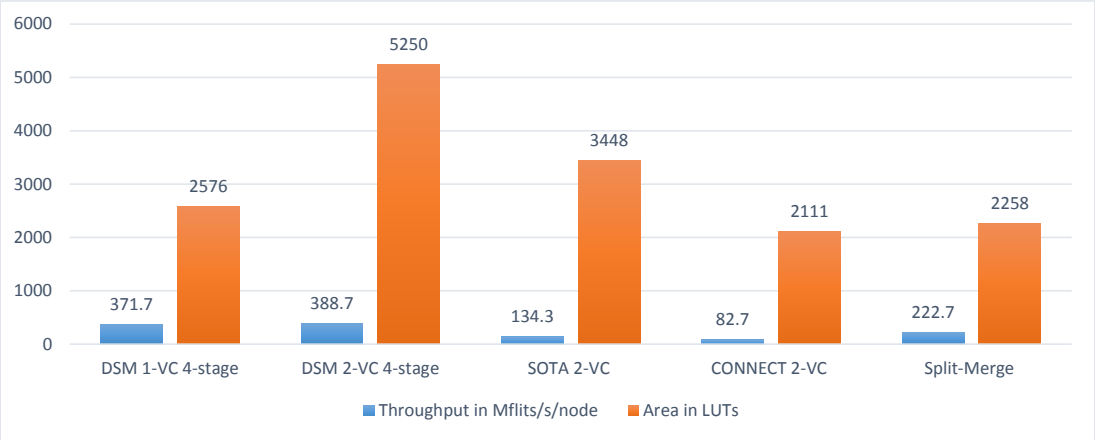


Figure 4.16: The maximum throughput and area of the different routers (based on Virtex-6 platform)

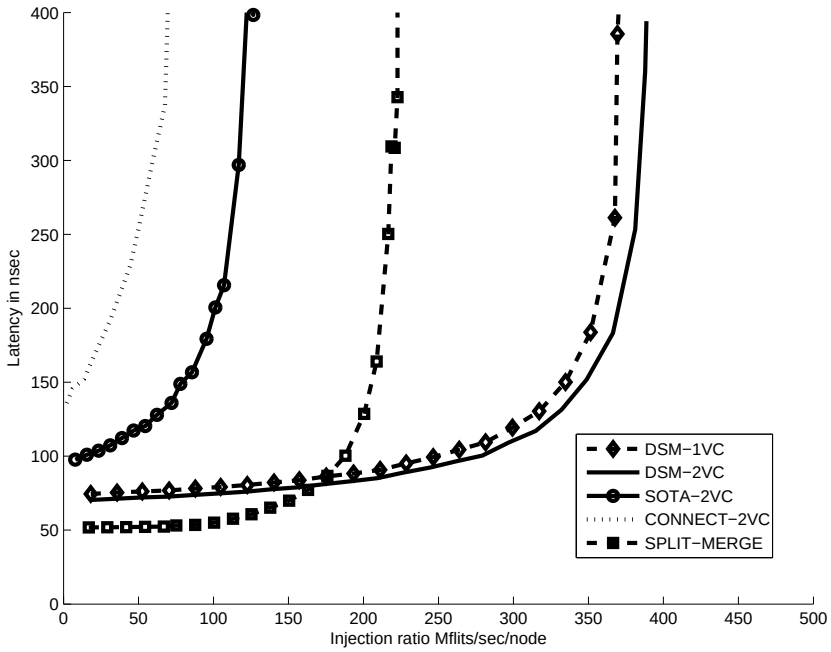
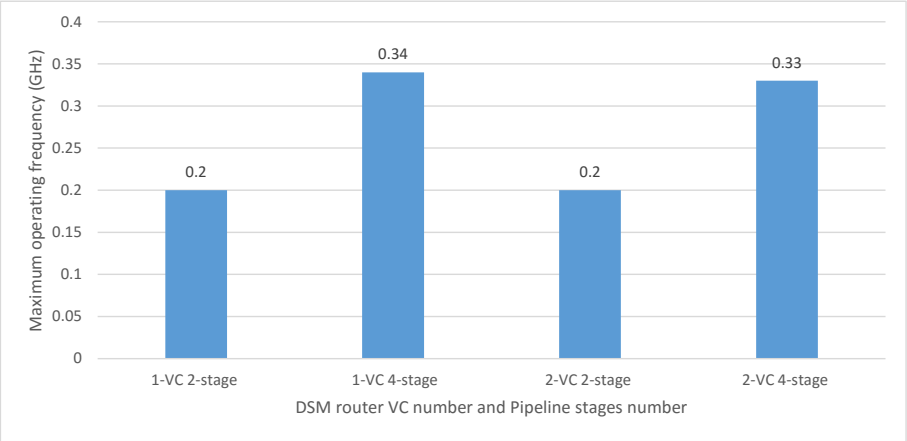


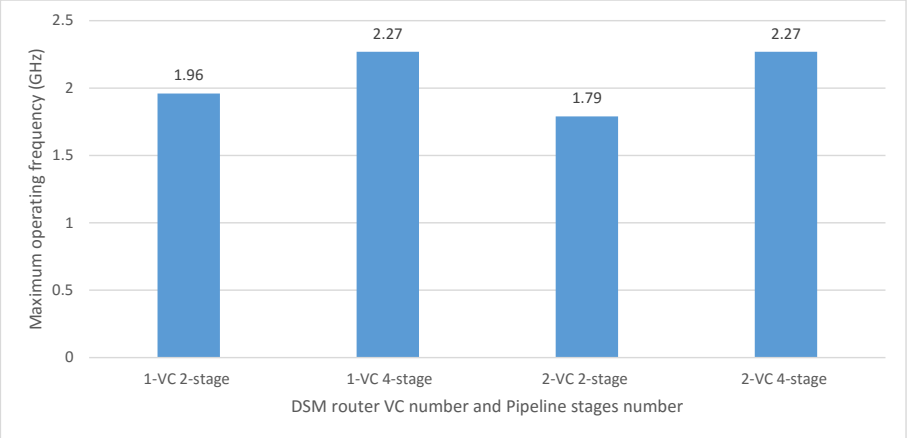
Figure 4.17: Load-Latency curves of DSM and other routers (based on Virtex-6 platform)

Table 4.1: A comparison between DSM, SOTA, CONNECT and Split-Merge (based on Virtex-6 platform)

	Net. Throughput (flits/cycle/node)	Max. Freq. (MHz)	Throughput (Mflits/s/node)	Area (LUT)
DSM $_{1VC}$	0.85	440	371.7	2576
DSM $_{2VC}$	0.89	436	388.7	5250
SOTA $_{2VC}$	0.69	195	134.3	3448
CONNECT $_{2VC}$	0.62	133	82.7	2111
Split-Merge	0.83	268	222.7	2258

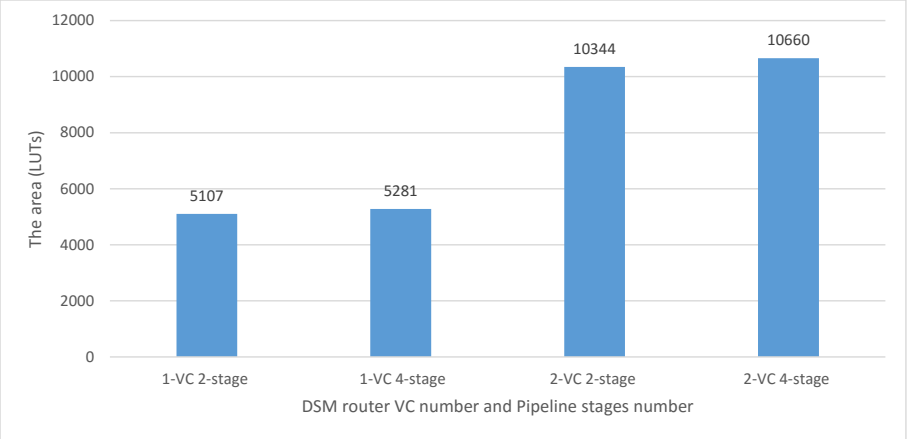


(a) Vertix-5 FPGA

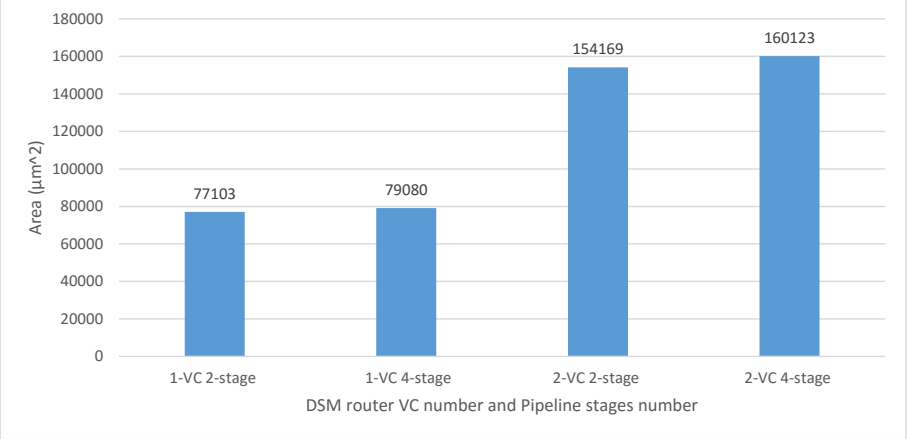


(b) UMC ASIC

Figure 4.18: The operating frequency of DSM in GHz (Virtex-5 and ASIC platfoms)

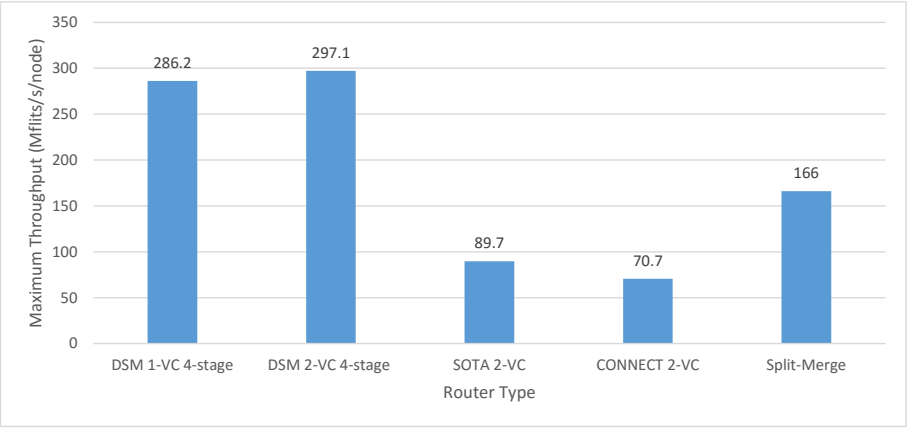


(a) Vertix-5 FPGA

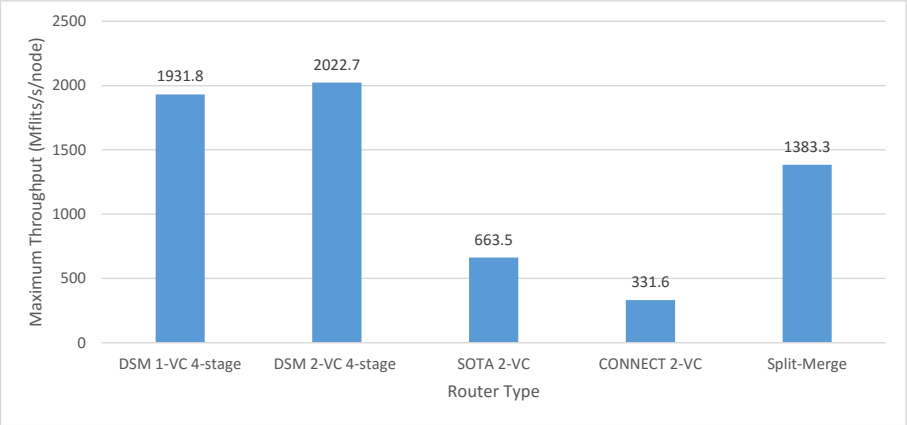


(b) UMC ASIC

Figure 4.19: The area of DSM (Virtex-5 and ASIC platfoms)



(a) Vertix-5 FPGA

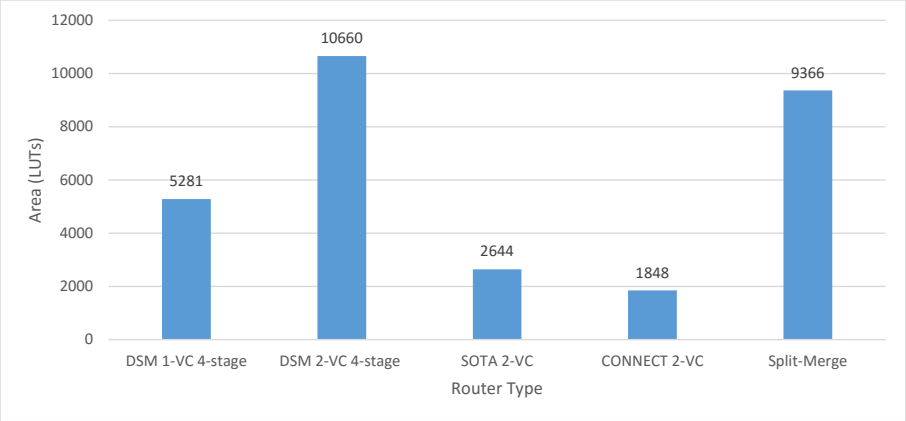


(b) UMC ASIC

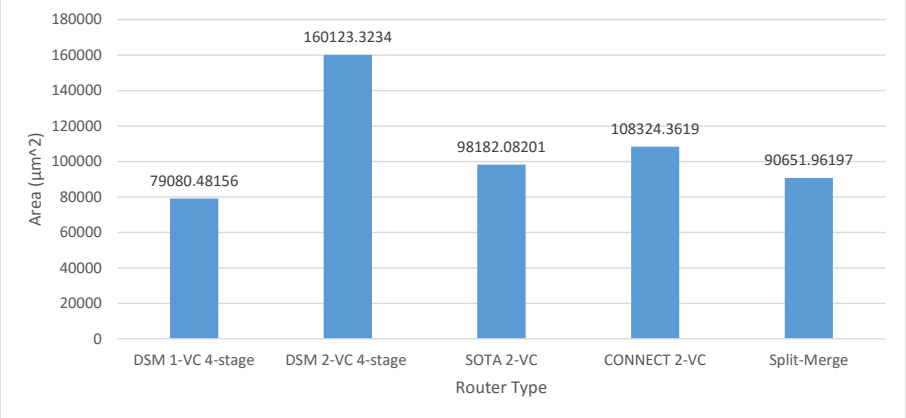
Figure 4.20: The maximum throughput of the different routers (Virtex-5 and ASIC platfoms)

The maximum throughput, measured in Mflits/s/node, and the area, measured in LUTs for Virtex-5 platform and in μm^2 for UMC ASIC pltfirm, for different routers are shown in Figure 4.20 and 4.21 respectively. Table 4.2 shows a comparison between DSM and the other routers in terms of network throughput, router maximum operating frequency, router throughput, and router area.

DSM is designed basically to target FPGAs with soft NoC to support high throughput communications without making any changes in the current FPGAs structure, which may be difficult to be done especially for the commercially available FPGAs. However, DSM shows great characteristics when implemented using ASIC which make it the appropriate router design for hard NoC, in case the change over the FPGA structure is targeted or applicable. The comparison shows the significant improvements of DSM router in term of throughput and latency for both soft NoC (Virtex-5 results) and hard NoC (UMC ASIC results). For soft NoC, DSM 1-VC throughput is 305%, 219%, and 73% higher



(a) Vertix-5 FPGA



(b) UMC ASIC

Figure 4.21: The area of the different routers (Virtex-5 and ASIC platfoms)

Table 4.2: A comparison between DSM, SOTA, CONNECT and Split-Merge (Virtex-5 and ASIC platfoms)

	Net. Throughput (flits/cycle/node)	Max. Freq. (MHz)	Throughput (Mflits/s/node)	Area (LUT)
DSM 1VC	0.85	336.7	286.45	5281
DSM 2VC	0.89	333.8	297.26	10660
SOTA 2VC	0.69	130	89.7	2644
CONNECT 2VC	0.62	114	70.68	1848
Split-Merge	0.83	200	166	9366

a) Vertix-5 FPGA.

	Net. Throughput (flits/cycle/node)	Max. Freq. (MHz)	Throughput (Mflits/s/node)	Area (μm^2)
DSM 1VC	0.85	2273	1932.05	79081
DSM 2VC	0.89	2273	2022.97	160123
SOTA 2VC	0.69	962	663.78	98182
CONNECT 2VC	0.62	535	331.7	108324
Split-Merge	0.83	1667	1383.61	90652

b) UMC ASIC.

Table 4.3: Network interface implementation results

Number of Slice Registers	22
Number of Slice LUTs	141
Maximum Frequency	600.166 MHz

than CONNECT, SOTA, and Split-Merge routers respectively while occupying almost the double area. For hard NoC, DSM 1-VC throughput is 482%, 191%, and 40% higher than CONNECT, SOTA, and Split-Merge routers respectively while occupying less area. The improvement DSM made for hard NoC is higher, in term of throughput and area, than the improvement it made for soft NoC. It gives a better throughput with less area even when compared with SOTA, the router that targets ASIC NoC.

4.6.2 Network Interface Results

In this section, we show the results of the proposed network interface when implemented over Virtex-6 FPGA. The implementation results of the network interface are summarized in Table 4.3.

The whole design parameters are chosen based on the design recommendations given in [7]. 2D-Mesh is chosen as our topology with 16 nodes. The buffer depth is 32-flit with an 8-flit packet length and a flit width of 32 bits.

Chapter 5

Priority-Select Arbiter: An Efficient Round-Robin Arbiter

5.1 Introduction

The arbiter is one of the widely used, yet critical, modules in various SoC applications. When multiple agents, such as processing elements, share one resource, such as a communication channel, a switch port, or a buffer, an arbiter is used to allocate the shared resource to one agent at a time. Fairness is one of the substantial features in an arbiter, which represent the equality of the resource assignment to the different requests. Thus, the fairness can be classified into three categories:

- Strong fairness: Each requester will have an equal average number of grants when calculated over an adequate number of arbitrations.
- Weak fairness: In which eventually each request will end up with a grant regardless when.
- FIFO fairness: The grants will be given to the requesters in the order they request it.

There are many types of arbiters that differ in their fairness. The first arbiter type is the fixed priority arbiter in which the priority given to each requester does not change over time; instead, the priority is linearly assigned to the requester. Fixed priority arbiter does not grantee any fairness; the low priority requester could starve while blocked by the higher priority requester. The second type is the variable priority arbiter. In the variable priority arbiter, the priority is continuously changing from cycle to cycle in order to provide fairness. Variable priority arbiters are categorized into oblivious arbiters and round-robin arbiters. Oblivious arbiters generate priority that is independent of the last grant; instead, the priority is based on the last cycle priority. Therefore, the next priority is generated using simple circuits, such as shift registers, however, it gives a weak fairness. Round-Robin Arbiters (RRAs) assign the lowest priority to the last granted requester to provide strong fairness. Hence, the priority vector generated is a shifted version of the grant vector [8].

RRAs are widely used in the routers of the networks. Routers basic function is to redirect packets from the source node to the destination node. They consist of buffers, virtual channels, and crossbar switches. The crossbar switch function is to connect the input ports to the output ports. Since many input ports may request sending packets simultaneously to the same output port, an arbiter is required by the scheduler of the crossbar switch to determine which input port will be connected to the output port at this cycle. Round-robin arbiter usually used to implement the crossbar scheduler to avoid input ports starvation **[29]**. In DSM, the arbiter is used in merge unit in order to determine which input port buffer will be connected to the output port.

In this chapter, we study different RRAs architectures and their performance in term of speed and area over FPGA and ASIC, thereafter, we propose a new RRA architecture that provides significant performance improvements over previous architectures. The chapter is organized as follows. A literature review of various RRAs found in the literature is presented in section 5.2. Section 5.3 introduces the proposed arbiter whereas the results are presented in section 5.4. Finally, the conclusion is in Section 5.5.

5.2 Literature Review

In this section, we will present different architectures found in the literature for implementing the round-robin arbiter (RRA). The RRA generally consists of an arbitration logic and priority update circuit. The arbitration logic is responsible for generating the grant signals while the priority update circuit is responsible for promoting the next input in line to the highest priority after the grant of the last input.

5.2.1 Baseline arbiter

Baseline arbiter uses a group of cells, one for each request-grant pair, to implement RRA as shown in Figure 5.1 **[8]**. Each cell contains a priority flag that indicates the cell has the highest priority. Therefore, the arbiter should initialize only one cell with an active priority flag. If the request signal corresponding to the highest priority cell is active, the cell generates a grant signal. However, if there is no request at the highest priority cell, the priority transfers to the next cell using Carry-out signal (Cout). In a similar way the adder works, each cell's Carry-out signal is connected to the next cell's Carry-in (Cin) signal, in addition, the last cell's Cout is connected to the first cell's Cin, therefore, the priority can propagate from a cell to another. If there are no requests, the priority will start to propagate from the highest priority cell through all cells until it returns back again to the old highest priority cell and the priority will remain the same. Figure 5.2 shows the implementation of a cell.

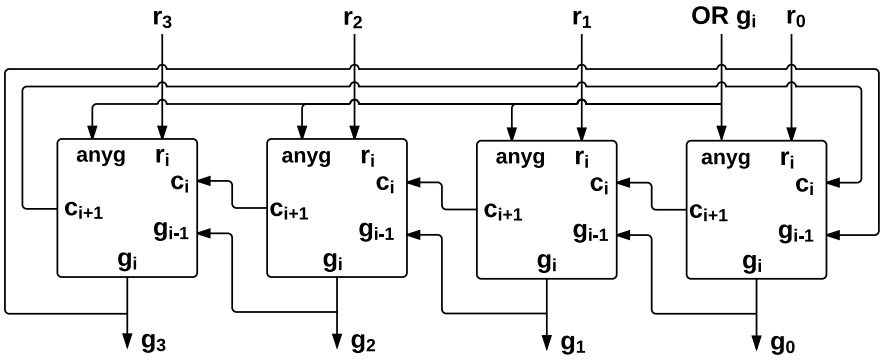


Figure 5.1: Baseline arbiter

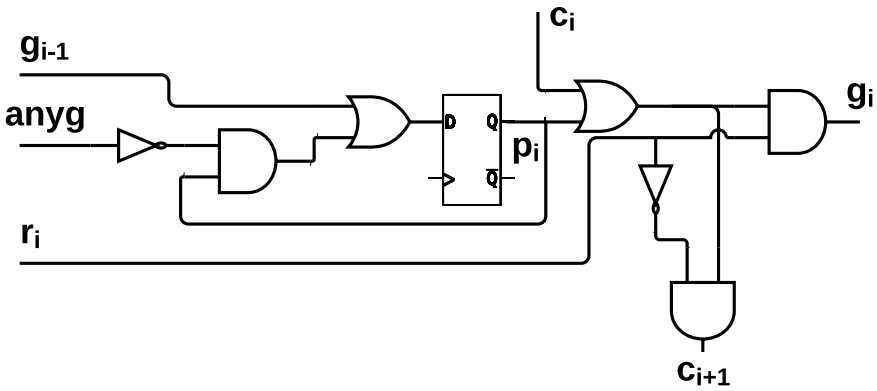


Figure 5.2: Baseline arbiter cell

The wrap-around of carry signal results in a combinational loop, that prevent most static timing analyzers from properly analyze the design. Therefore, synthesis tools cannot optimize these designs.

5.2.2 Timing speculative arbiter

In order to avoid the long propagation delay of the baseline arbiter, timing speculative arbiter limits the propagation of priority to a fixed number of cells (s) **[30]**. The arbitration is pipelined if the priority needs to propagate over a number of cell's larger than the (s) cells. The propagation limitation is implemented by connecting each cell to the next (s) cell, i.e. connecting the cell i with the cell i+s. If there is no request in the highest priority cell i or in the cells from i to i+s, the priority will be transferred to cell i+s in this cycle so the priority starts to propagate from it in the next cycle. If there are no requests for all cell's, the priority should remain at the cell i. To differentiate between the idle case (where no requests) and the pipeline case (the requests are beyond i+s), the arbiter adds

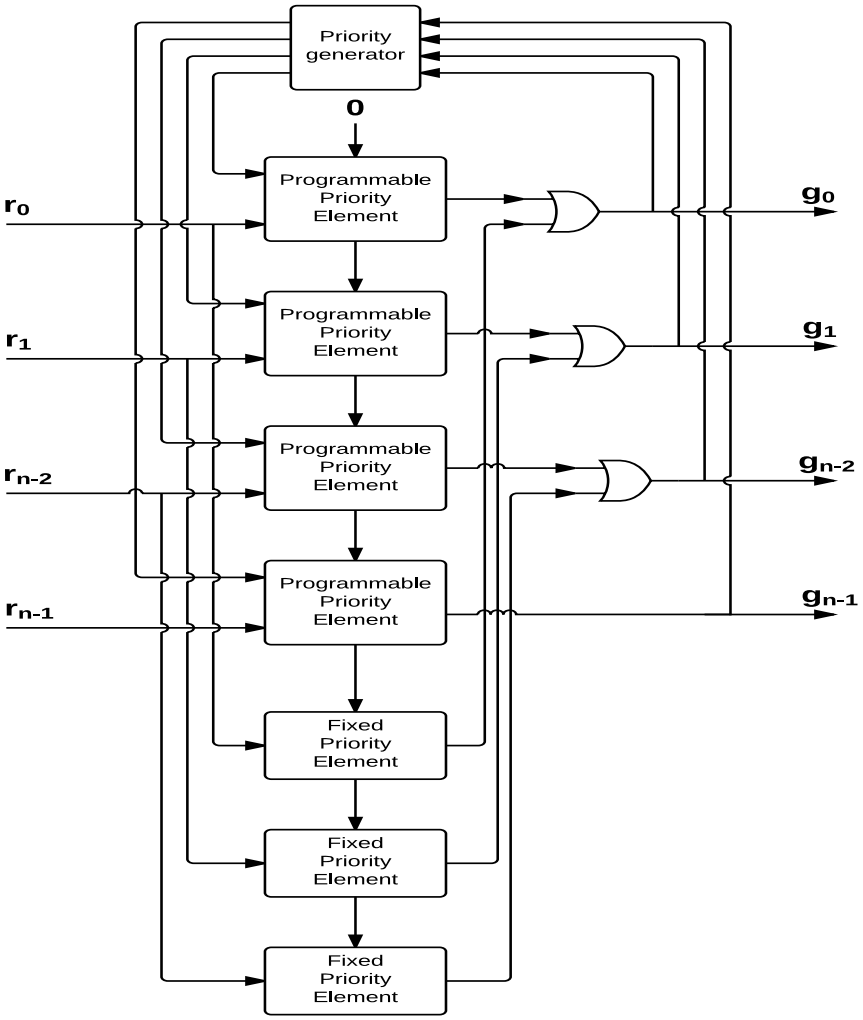


Figure 5.3: Acyclic Arbiter

an extra signal (I). (I) indicates an idle arbitration case by NOR all input requests. Even though this architecture still has a combinational loop, it is faster than the baseline RRA.

5.2.3 Acyclic arbiter

To overcome the cyclic implementation, **[18]** proposed an acyclic RRA. In this RRA, a fixed priority cells chain replaces the connection between the last and the first cell as shown in Figure 5.3. The priority propagates through the carry signals to the next cells if the higher priority cells receive no requests. In case, the last cell has the priority while no request has received, it propagates the carry to the fixed priority cells instead of directly connect it to the first cell. Because of the carry propagation through fixed and variable priority cells, this implementation has a long critical path. However, it avoids the combinational loop.

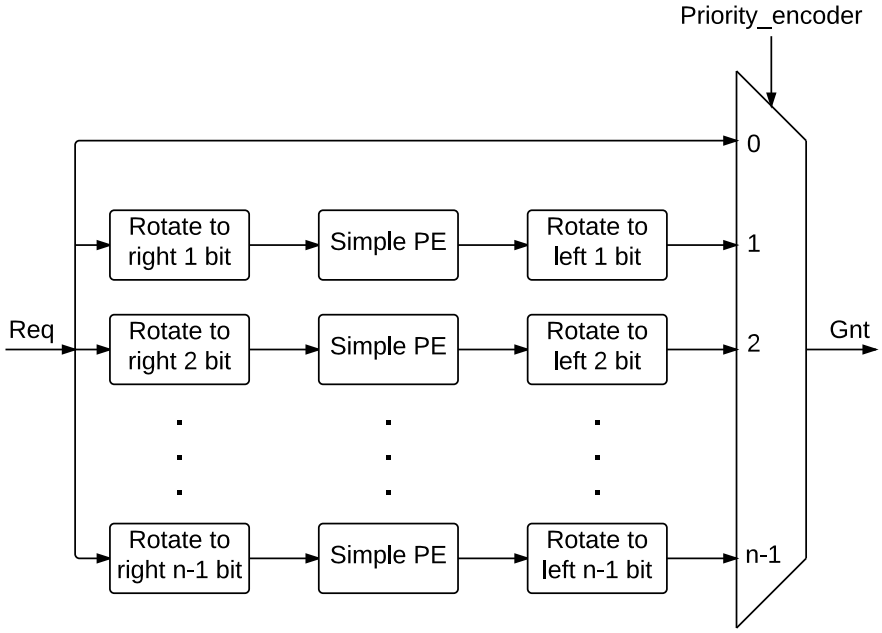


Figure 5.4: Exhaustive PE arbiter

5.2.4 Priority-encoder based arbiter

This arbiter type consists of a group of priority encoders (PEs) **[31]**. A priority encoder is an arbiter with fixed priority in which the highest priority belongs to the first input.

5.2.4.1 Exhaustive PE arbiter

Exhaustive PE arbiter uses N PEs to make an N-bit RRA. In order to input different request line on each PE highest priority position, first input, the circuit rotates and inputs the request vector to each PE. The rotation is implemented by wiring connections. Therefore, according to the priority vector, a multiplexer selects one of the PEs. For large values of N, this RRA requires a large area. Figure 5.4 shows the internal architecture of Exhaustive PE arbiter.

5.2.4.2 Dual-path PE arbiter

Dual-path PE arbiter is introduced in **[31]**. It uses two parallel PEs to implement round-robin algorithm. The first PE starts searching for an active request from the highest priority bit to the lowest priority bit, last bit, without cycling back to the first bit. Since PE has a fixed priority, the requesters to PE that lead the highest priority request has to be disabled. This is achieved by using a thermometer code to encode the priority vector and then use the request vector to mask it. For the second PE, the request vector is entered directly without masking. While both PEs work in parallel, a multiplexer is used to select between the two outputs. If the request exists only before the highest priority bit, there

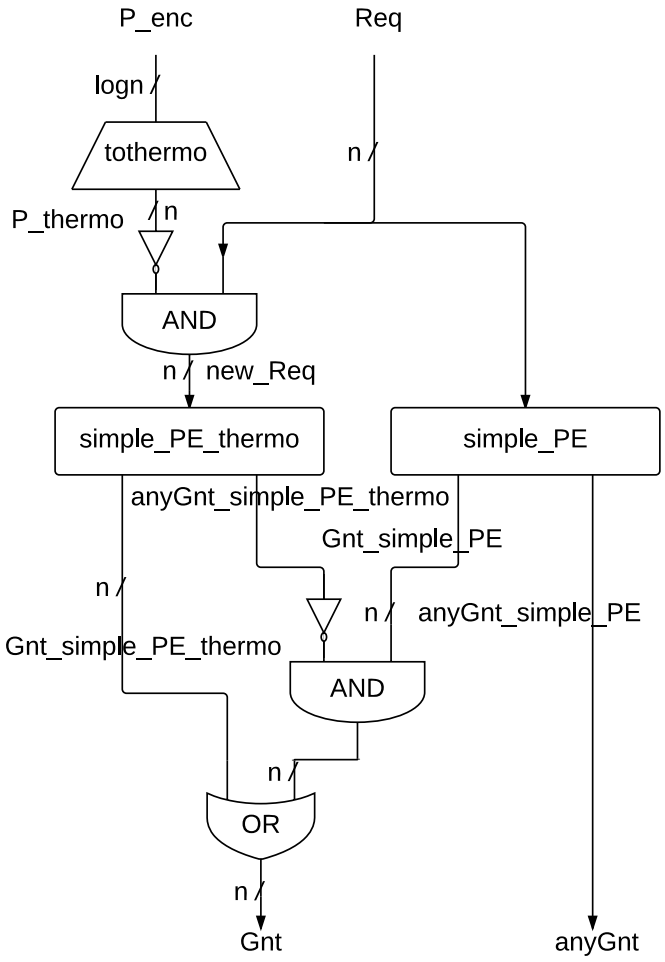


Figure 5.5: Dual-path PE Arbiter

will be no active requests after the request vector being masking, therefore, the multiplexer will select the second PE. Consequently, the multiplexer can be reduced as in Figure 5.5 shows. The priority is thermometer coded and the grant vector is one-hot coded that make the priority update circuit more complex, thus, before updating the priority the grant vector needs to be a thermometer encoded.

5.2.5 Parallel prefix arbiter

This architecture is introduced in **[32]**. It generates each grant signal as a combinational function of the priority vectors bits and the request bits instead of using multiple units of fixed priority arbiter. Despite the high speed of this architecture, it does not scalable well for a higher number of requests.

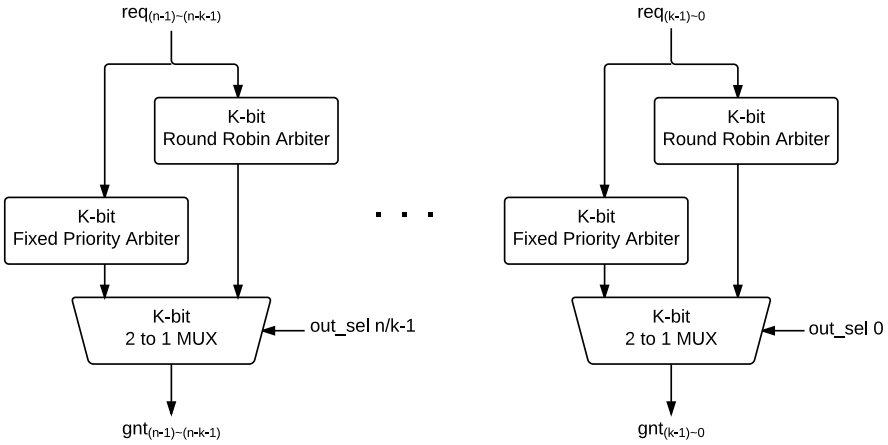


Figure 5.6: Priority-Select Arbiter

5.3 Proposed Arbiter

Selecting the arbiter type is a compromise between fairness and implementation characteristics such as area and operating frequency. Fixed priority arbiter does not guarantee fairness between requests, but has the highest frequency and the smallest area among other arbiters. On the contrary, round-robin arbiters grantee fairness between requests, but they consume more area and operate at low frequency because the priority changes over the time and has to propagate from location to another.

We are proposing a new round-robin arbiter (Priority-Select arbiter) with advantageous implementation characteristics [33]. In our arbiter, to boost the operating frequency we subdivide the n-bit arbiter to n/k units, each of which consists of a k-bit round-robin arbiter, implemented using either of the literature architectures, a k-bit fixed priority arbiter, and a multiplexer to select between the round-robin and fixed arbiters as Figure 5.6 shows. Let us assume that the priority propagates from the least bit to the most bit (from right to left) and wrap-around to the least bit again. When n-bit requests divided into n/k groups, only the group that contains the priority bit needs to have a programmable priority, which we will refer to as the priority group. The other groups need to have a fixed priority with the highest priority located at the least bit. Thus, only one multiplexer selects the priority group round-robin arbiter while the remaining multiplexers select the fixed priority arbiters. Finally, to prevent multiple grants from different groups, each group will block the following group's grants, starting from the priority group, if it has at least one grant or one request.

For the k-bit round-robin arbiter of the priority group, if the priority was at bit m, where $0 \leq m \leq k-1$, any bit preceding m, located at i where $0 \leq i \leq m-1$, should have a priority lower

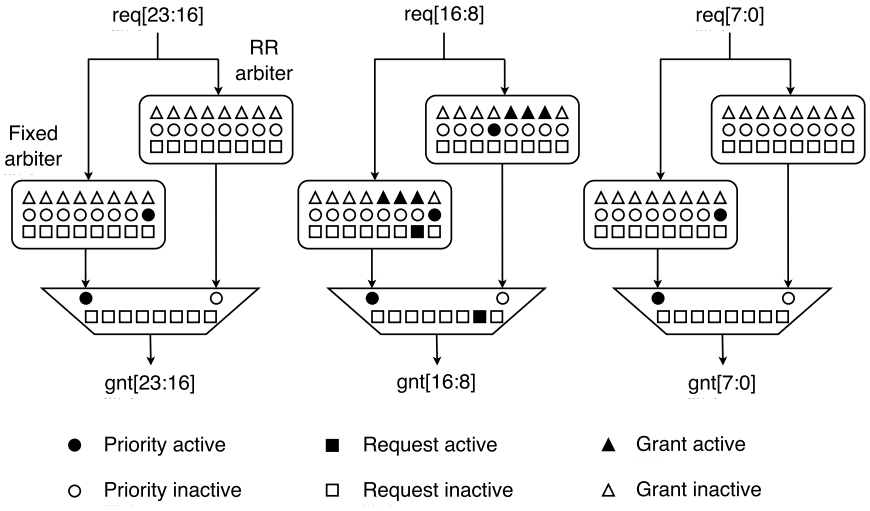


Figure 5.7: Priority-Select Arbiter Example

than all remaining group's bits. Therefore, the priority of round-robin arbiter should be prevented from propagation from the most bit to the least bit that also help in enhancing the frequency. Subsequently, the Round-Robin arbiter will respond only to requests starting from the priority bit to the most bit. If there are any requests in the bits preceding the priority bit, and no requests in all the group's bits, including priority group round-robin arbiter, in that particular case, the output of the priority group fixed arbiter will be selected.

For example, suppose we have 24 bits arbiter (from bit 0 to 23) subdivided into three groups (from group 0 to 2), each group of 8 bits as shown in Figure 5.7. In addition, assume that the priority was at bit 12, and requests 9, 10, and 11 were active. The priority group will be Group 1 and the multiplexer of this group will select round-robin arbiter while the other remaining groups multiplexer will select fixed arbiter. At this point, the round-robin arbiter of priority group will grant no request because the priority cannot propagate from bit 15 to bit 8. In addition, the other groups also will generate no grants because they have no requests. However, the multiplexer of the priority group will change the selection to the fixed arbiter which will grant bit 9.

The k-bit round-robin arbiter group can be implemented using any RRA architecture after being simplified to remove the cyclic search, as we have mentioned earlier that is not allowed for the carry to propagate from the most bit to the least bit. We selected the exhaustive PE architecture as the k-bit RRA. The k-bit exhaustive PE RRA has a high operating frequency but occupies a large area because it consists of k k-bit PEs. However, after the cyclic search being removed, the k-bit PEs are simplified to [k, k-1, k-2... 2, 1]-bit PEs, which result in significant reduction in the area used.

5.4 Embed PS Arbiter in DSM Router

DSM with VCs was introduced in the last chapter. It provided advantageous network performance by enhancing the network throughput and decreasing the HoL effect. However, adding VCs to DSM has a negative impact on the implementation characteristics, specifically, the frequency and the area. In this section, we will focus on elevating the frequency problem.

The limitation on the maximum operating frequency is imposed by the usage of the arbiters in implementing VCs for DSM. Both the split and merge units use an arbiter to support VCs. The results showed in the last chapter were based on implementing VCs using the conventional arbiters. Therefore, we replaced these conventional arbiters with PS arbiter to move the critical path from the arbiter module to another module. The size of the arbiter is based on the number of VCs used in DSM. However, PS arbiter is characterized by two parameters; the arbiter size, which is the number of requests the arbiter can support, and the group size. While the arbiter size is selected based on the number of VCs, the group size is selected based on the results of implementing the arbiter standalone. The implementation results of PS arbiter, stand alone, and DSM, when implemented using PS arbiter, are showed in the next section.

5.5 Results

In this section, our proposed architecture (Priority-Select arbiter) and the previous RRA architectures implementation results are discussed. All the different architectures reviewed in Section II, except that contain a combinational loop, besides the proposed arbiter, are implemented using Verilog HDL. PS arbiter basically targets ASIC, however, it used in FPGA-based router implementation, DSM router. Therefore, both ASIC and FPGA implementation results will be shown for the standalone arbiter, while only FPGA results will be introduced for DSM router. Virtex-5 FPGA (part xc5vlx50t-3ff665, speed grade -3) is used as the implementation FPGA target platform. Xilinx ISE 14.6 tool is used to obtain the FPGA implementation results. ASIC results are based on synthesizing over a UMC 65nm Typical standard-cell fabrication technology. The ASIC implementation results are given by Synopsys Design Compiler.

5.5.1 PS Arbiter Results

Table 5.1, 5.2 and 5.3 present a summary of all implementation results. The area, in LUTs, and the maximum operating frequency, in MHz, of the previous architectures implementation for different number of requesters are shown in Table 5.1 and 5.2. The results emphasize that the exhaustive PE arbiter and the parallel prefix arbiter have the highest operating frequency between all the others, however, are not scalable because their areas grow exponentially.

	Area in LUTs				
	n=4	n=8	n=16	n=32	n=64
Parallel Prefix	20	63	163	406	1031
Exhaustive PE	20	63	171	1453	3767
Acyclic	21	45	104	242	506
Dual-Path	7	49	115	262	500

a) FPGA platform results

	Area in μm^2				
	n=4	n=8	n=16	n=32	n=64
Parallel Prefix	106	389	1575	7573	26673
Exhaustive PE	156	724	2554	9294	35657
Acyclic	194	399	798	1806	3529
Dual-Path	101	279	633	1256	3397

b) ASIC platform results

Table 5.1: Area Implementation Results of previous architectures for different no. of requesters (n)

	Frequency in MHz				
	n=4	n=8	n=16	n=32	n=64
Parallel Prefix	505.8	449.8	316.1	292.1	242.8
Exhaustive PE	505.8	449.8	334.3	256.2	221.4
Acyclic	500.4	403.2	236.9	180.3	166.2
Dual-Path	740.1	256.9	184.3	143.0	137.3

a) FPGA platform results

	Frequency in MHz				
	n=4	n=8	n=16	n=32	n=64
Parallel Prefix	3225.8	2777.7	2272.7	1960.8	1612.9
Exhaustive PE	3225.8	2631.6	2173.9	1818.1	1612.9
Acyclic	2381	1470.6	793.7	426.7	218.3
Dual-Path	2500	1960.7	1515.1	1190.4	990.1

b) ASIC platform results

Table 5.2: Frequency Implementation Results of previous architectures for dierent no. of requesters (n)

	n=4	n=8		n=16		
	k=2	k=2	k=4	k=2	k=4	k=8
Area (LUTs)	19	48	47	123	102	147
Frequency (MHz)	493.4	378.0	329.9	268.1	244.3	286.2

	n=32			
	k=2	k=4	k=8	k=16
Area (LUTs)	280	305	228	257
Frequency (MHz)	209.8	223.9	284.8	248.7

	n=64				
	k=2	k=4	k=8	k=16	k=32
Area (LUTs)	515	548	419	471	398
Frequency (MHz)	193.5	200.0	251.4	215.1	216.6

a) FPGA platform results

	n=4	n=8		n=16		
	k=2	k=2	k=4	k=2	k=4	k=8
Area (LUTs)	152	568	386	1084	984	910
Frequency (MHz)	3030.3	2272.7	2381	2000	2040.8	1923.1

	n=32			
	k=2	k=4	k=8	k=16
Area (LUTs)	3387	2466	1818.2	1138
Frequency (MHz)	1754.4	1754.4	2010	1694.9

	n=64				
	k=2	k=4	k=8	k=16	k=32
Area (LUTs)	9896	10208	4148	1945	1380
Frequency (MHz)	1639.3	1724.1	1587.3	1818.1	1923.1

b) ASIC platform results

Table 5.3: Implementation Results of the proposed arbiter for different no. of requesters (n) and different bits per unit (k)

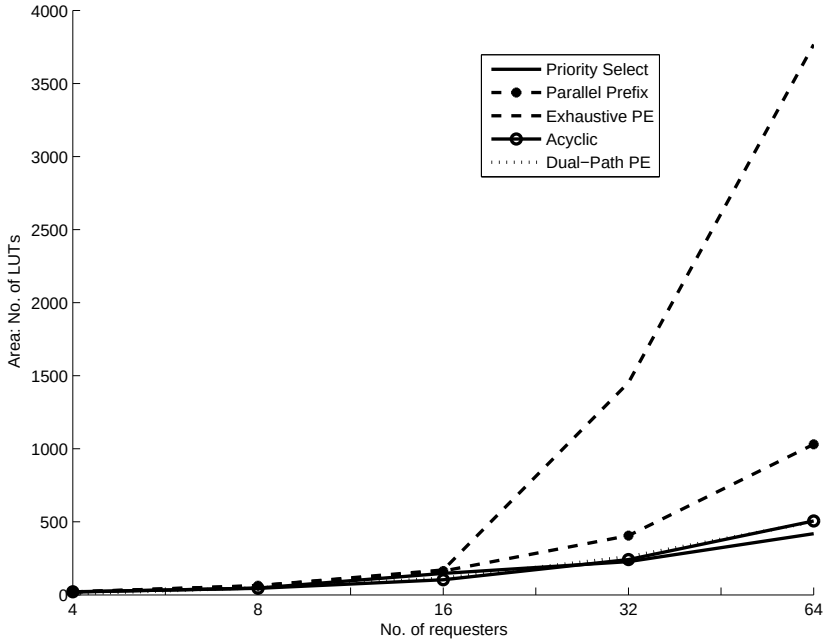
Table 5.3 shows the results of the implementation of the proposed arbiter for different values of the number of requesters and different numbers of group bits. The table asserts that the area gradually increases with the number of requesters increase, and provides the smallest area compared with the other architectures. Furthermore, the arbiter operates at a high frequency that exceeds all arbiters found in the literature for a large number of requesters. The critical path delay in the proposed arbiter is the unit group delay; however, increasing the number of group unit's decreases the number of requesters in each group which intern decreases its delay and move the critical delay to the controller that controls the unit group multiplexer selection. This leads to an optimization problem that introduces a degree of freedom to choose, for a given number of requesters, the proper group unit size.

Figure 5.8 and Figure 5.9 show a comparison between the priority-select arbiter, the proposed arbiter, and the other presented arbiters. Figure 5.8 shows, for a different number of requesters, the area of the implemented arbiters. For a large number of requesters, the priority-select arbiter provides the lowest area among all other arbiters. The configuration of the proposed arbiter shown in the figure is k=2 for [4, 8]-bit arbiter and k=8 for [16, 32, 64]-bit arbiter (where k is the group unit size).

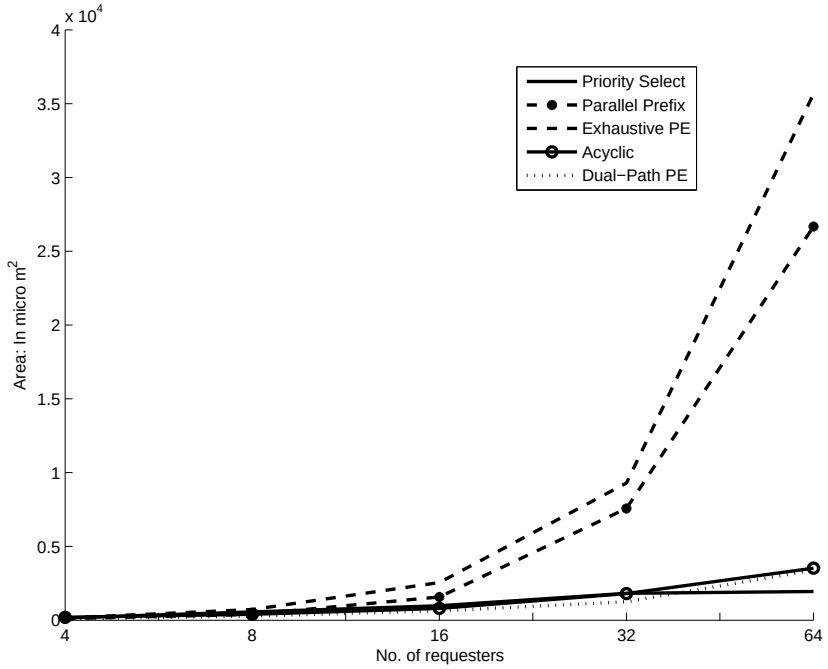
Figure 5.9 shows the implemented arbiters maximum operating frequencies. The priority-select arbiter provides the highest maximum frequency. Both parallel prefix and exhaustive PE give almost the same operating frequency, but at the expense of area as Figure 5.8 shows. The configuration of the priority-select arbiter is as follows: k=2 for [4, 8]-bit arbiter and k=8 for [16, 32, 64]-bit arbiter.

5.5.2 DSM with PS Arbiter Results

Table 5.4 shows the implementation results, maximum operating frequency and area, of DSM router VCs with and without PS arbiter over the FPGA platform. The results show a great enhancement in the operating frequency starting from 27.8% to increasing 107.5% increasing in the frequency for DSM router with 4-VC and 32-VC respectively. Moreover, the results show area saving but with less percentage.

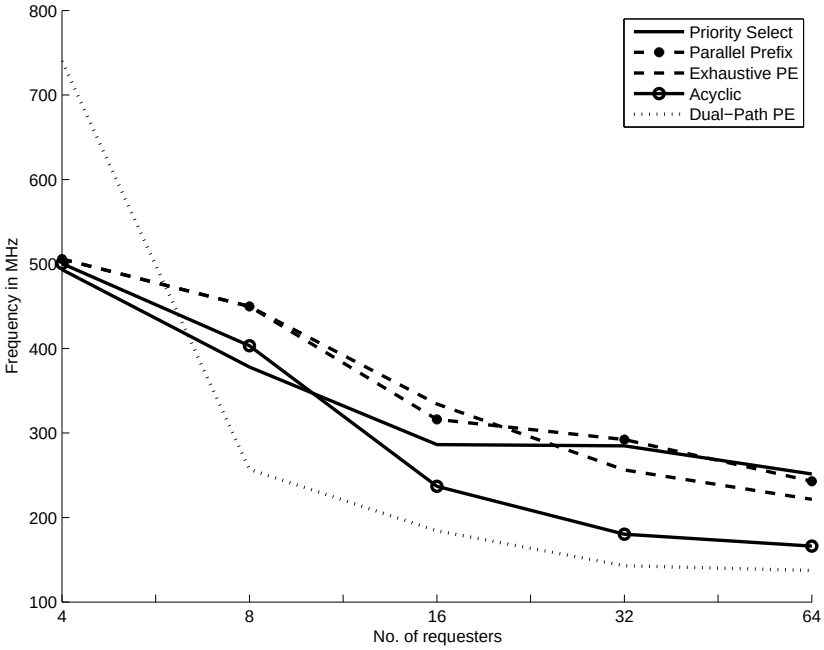


(a) FPGA platform results

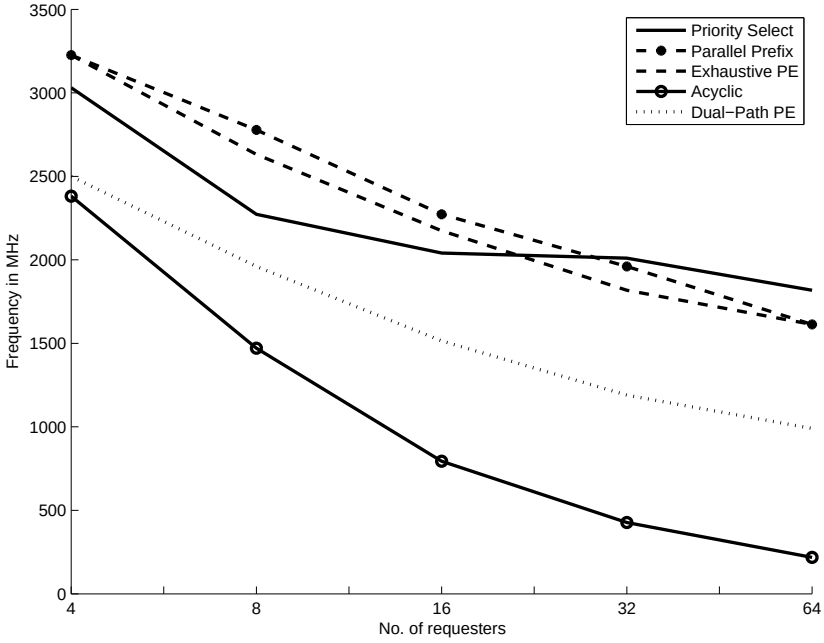


(b) ASIC platform results

Figure 5.8: Area of different arbiters



(a) FPGA platform results



(b) ASIC platform results

Figure 5.9: Frequency of different arbiters

	n=4	n=8	n=16	n=32
Frequency in MHz	195.427	150.060	96.441	62.477
Area in LUTs	18414	36750	74063	145929

a) The frequency and area of DSM with VCs using conventional arbiter

	n=4	n=8	n=16	n=32
Frequency in MHz	249.688	235.460	156.299	129.634
Area in LUTs	17892	35961	72844	144996

b) The frequency and area of DSM with VCs using PS arbiter

Table 5.4: Frequecy and Area Implementation Results of DSM for different no. of VCs (n) using conventional arbiter and PS arbiter

Chapter 6

Dynamic Virtual Channels

6.1 Introduction

Router buffers have an effective role in the overall operation of NoC. However, among the various components comprising the SoC interconnection, buffers are the components that consume the largest area and power, about 64% of the total static power are consumed by buffers [34]. Also, the dynamic power consumed by the buffer significantly increase with the packet throughput increment [35]. Similarly, the chip area occupied by buffers is the biggest compared to other on-chip router components [36–38].

Furthermore, virtual channels have a critical role in NoC performance. However, in its implementation, it depends fundamentally on buffers. Studies show that for low traffic, a low number of VCs can enhance the network throughput with low latency, while for high traffic intensity, increasing the number of VCs is more effective in boosting the performance than increasing the buffer depth [39].

VCs are implemented using separate buffer slots per VC, each slot holds a flit, when a header flit enters a VC, the VC allocator reserve this VC to the incoming packet until the tail flit comes. Therefore the VC allocator can assign the same VC to the new coming packet. This means that two or more packets can exist in the same VC which may lead to HOL blocking if the first packet blocked, however, VCs can reduce the effect of HOL blocking by assigning different packets to different VCs as possible. Static Virtual Channel (SVC) mechanism is the convention, and the most commonly used, in implementing VCs. In SVC, the buffer slots are allocated statically to different VCs. The implementation is simple and characterizes by high operating frequency, however, the buffers size almost double with the VCs number increment. On the other hand, Dynamic Virtual Channel (DVC) mechanism allocates the buffer slots dynamically to different VCs, upon VC request, on a buffer slot basis. DVC buffers are characterized as their VCs are adapted with the variations in the traffic condition. In other words, the depth of each VC queue varies, in the extreme case, from zero to the whole buffer size, according to the traffic condition. As a result, the size of buffers greatly decreased for the same network performance at certain SVC number. This enhancement in the area is at the expense of operating frequency as the DVS has a lower operating frequency, when compared to SVC, due to the relative complexity of the implementation.

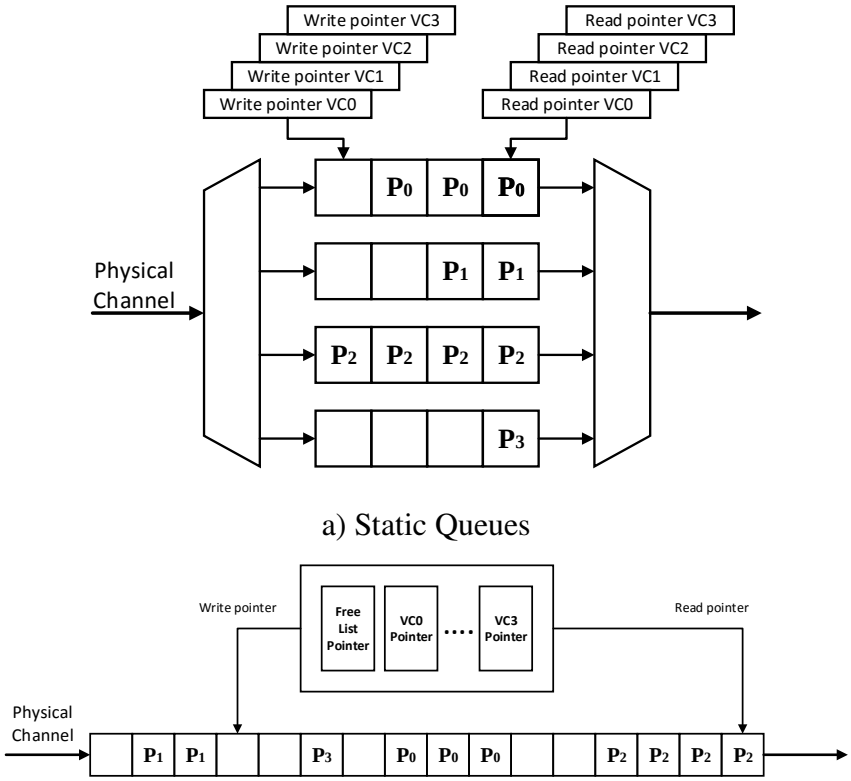


Figure 6.1: The architecture of conventional static and dynamic virtual channel buffers.

Figure 6.1 shows the architecture of conventional static and dynamic virtual channel buffers.

6.2 Related Work in Buffer Design

In this section, different dynamic virtual channel that presented in the literature are reviewed.

6.2.1 DAMQ

Dynamically Allocated Multi-Queue (DAMQ) is a linked list based dynamic buffer presented in [40]. In DAMQ, an output port multi-queue is used to store the packets into queues.

In order to dynamically hold multiple queues of packets in a DAMQ buffer, linked list is used in the implementation. DAMQ preserve (k+1) linked lists per buffer, where k is the number of ports; one list for each output port, including the local port, and one list

for empty slots in the buffer. For each linked list there are head and tail registers. The head register points to the first slot in the linked list, which also the first flit to leave the linked list. Similarly, the tail register point to the last slot in the linked list, which hold the recent entered flit. Besides the head and the tail registers, each DAMQ buffer contains a pointer register that holds the address of the next slot to maintain the order of the FIFO. The logic that controls the DAMQ buffer is quite complex. The process of receiving or transmitting a flit involves moving a slot from a linked list to another with its all associated registers manipulation. In receiving, a slot from the empty slots linked list is moved to the appropriate output port linked list, while in transmitting process, a slot from one of the lists is moved to the empty slots linked list.

Moving a slot from a linked list to another logically requires three cycles:

- Cycle 1: Starting with the linked list that contains the slot that needs to be moved, the slot to be moved is pointed by the head register. The value of the head register is copied into a temporary storage, while the pointer register is used to locate the next slot to the head register; therefore, we copy the next slot address to the head register.
- Cycle 2: The temporary register content, the old source head register, is then used to update the pointer register by adding it to the destination tail next slot. In addition, the temporary register is copied to the destination linked list tail pointer.
- Cycle 3: Again the pointer register is updated by adding the NULL value to the next slot of the new tail address of the destination linked list.

Figure 6.2 shows an example of DAMQ buffer that supports four VCs.

There are three main drawbacks in DAMQ. First, the high latency where reading or writing takes at least three cycles. Moreover, the implementation is complex and that can significantly decrease the operating frequency. Finally, as the depth of each queue can vary according to the traffic situation, if one of the VCs is blocked it will continue to reserve slots from the DAMQ buffer for its incoming flit till it consumes the most of the buffer and causes the other VCs to block. The detailed implementation of DAMQ is explained in a few researches, we will present them in the next sections.

6.2.2 SCB

In Self-Compacting Buffers (SCB) scheme [41], the buffer is dynamically divided into regions, each region corresponds to a linked list or output channel in DAMQ. However, the slots of each list are kept contiguous and the order of the lists is preserved, i.e. the starting location of VC n is always less than VC n+1. As a result of making the lists contiguous, there will be no need for a next slot pointer, because all list slots are stored

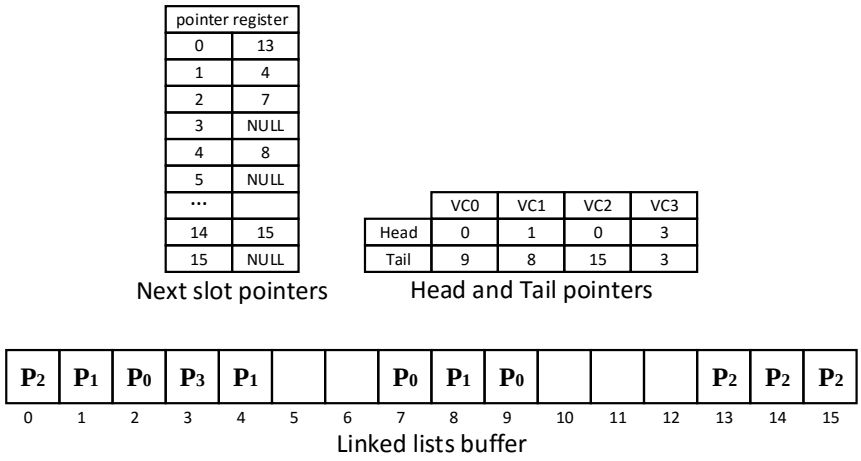


Figure 6.2: An example of DAMQ buffer that support four VCs: The packet that belongs to VC i is doneted by P_i.

adjacently in order. In addition, an empty slot list will not be needed. In order to keep the slots of a list contiguous, the buffer supports slots push in two directions. When a flit required to be inserted in the middle of the buffer, all subsequent slots will move down to create a space for the incoming flit. Similarly, when a flit required to be removed from the buffer, all the subsequent slots mush move up to keep the buffer compact.

Figure 6.3 shows an example of SCB storage buffer that supports four VCs compared to a DAMQ buffer when storing the same packets.

6.2.3 FC-CB

The Fully Connected Circular Buffered (FC-CB) is based on the circular buffer scheme. The buffer has a free space counter that counts the number of available slots in all VCs, while for each VC there is a head register, a tail register, a residing counter, a passing counter, a state register and linkage registers. The head register and the tail register point to the first and the last slot in the VC queue respectively. The residing counter tracks the number of flits that reside in the VC. The passing counter specifies the number of flits to be transmitted for the currently transmitting packet. The state register shows the availability of the VC. The linkage register stores information about the output port and the virtual channel number corresponding to the VC. However, the state register and the linkage register are more related to the routing mechanism. Similar to self-compacting DAMQ [41], each VC queue slots is kept adjacent while allowing for bubbles of empty slots to exist between the VCs, besides that, the relative order of VC does not change through the operation. A head register and a tail register are kept for each VC to keep tracking the queue position. However, unlike the self-compacting DAMQ, the VC queues move in one direction as the circular buffer allows for the slots to wrap around. To avoid

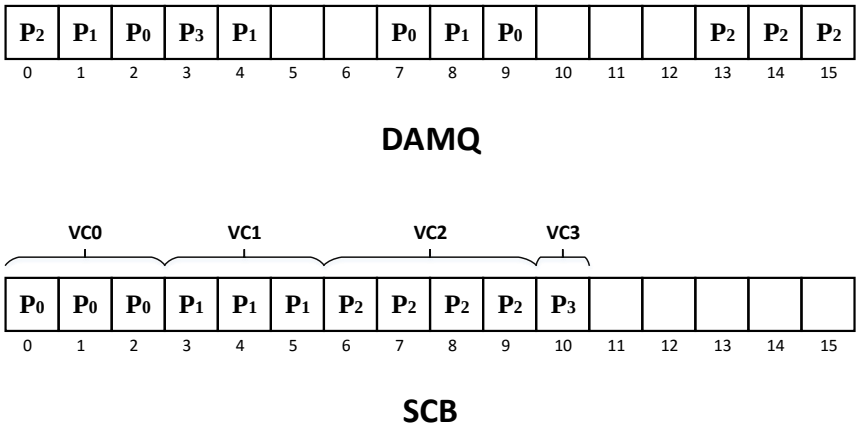


Figure 6.3: An example of SCB storage buffer that supports four VCs compared to a DAMQ buffer.

the case where the blocked VC consume all the slots and block the other VCs, a flit is kept reserved for each channel to ensure that there is a flow path for each VC.

Figure 6.4 shows an example of FC-CB storage buffer that supports four VCs compared to a DAMQ buffer when storing the same packets.

6.2.4 DAMQ-all

This scheme is based on self-compacting DAMQ [41] with modifications that allow the lists to be separated by a bubble of slots [42]. When a flit required to be inserted in the middle of the buffer, all slots that contain a valid data will be moved down until an empty slot is encountered to make a space for the incoming flit. However, when a flit required to be removed from the buffer, there will be no need for moving any slots allowing for empty slots to exist between lists. Similar to FC-CB, the buffer is implemented with a reserved space for the VCs. However, the reserved space could be shared between all VCs or it could be reserved per VC. The number of reserved slots per VC can be adjusted; however, the number of two flits is selected as it gives a satisfactory performance, based on their simulations, with more available free space to be shared. The drawback here, which does not exist in FC-CB, is that the buffer should support the shift in two directions, up and down, as there no wrap around here, which may complicate the implementation.

Figure 6.5 shows an example of FC-CB storage buffer that supports four VCs compared to a DAMQ buffer when storing the same packets.

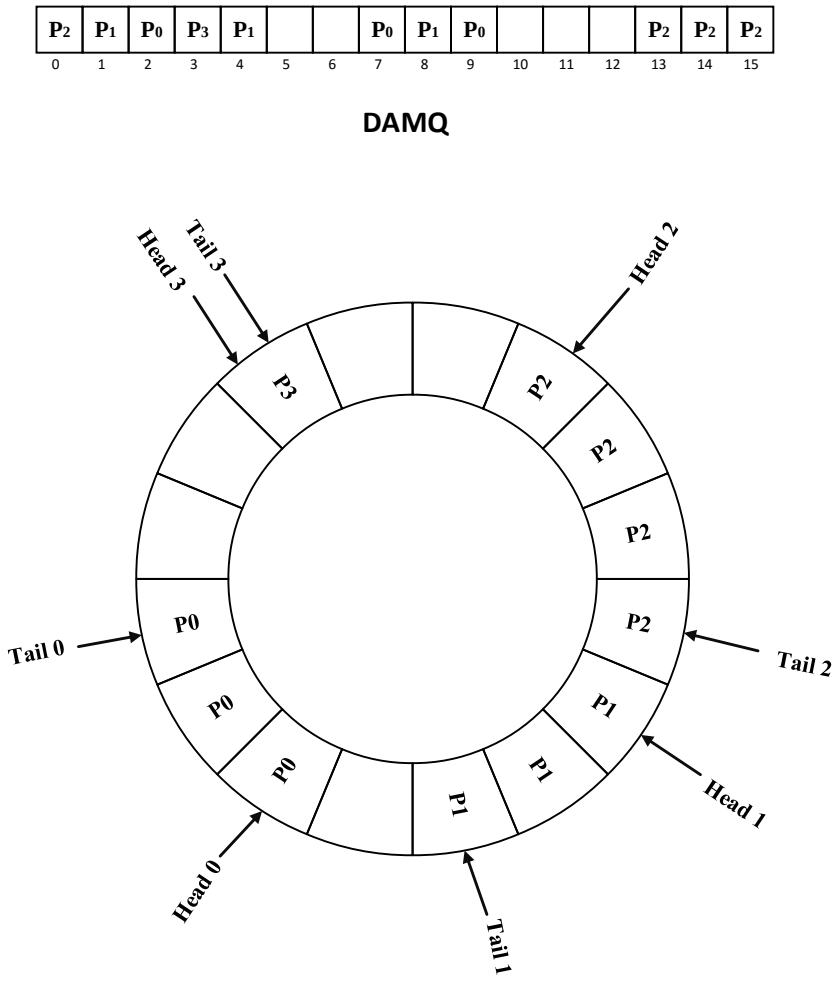


Figure 6.4: An example of FC-CB storage buffer that supports four VCs compared to a DAMQ buffer.

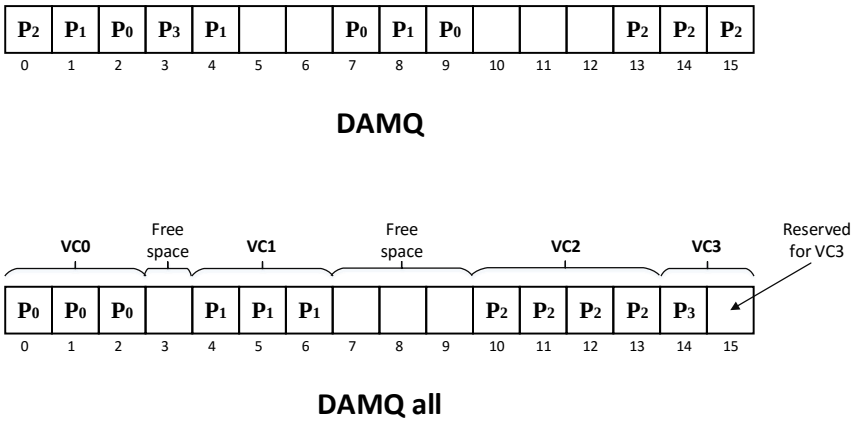


Figure 6.5: An example of DAMQ-all storage buffer that supports four VCs compared to a DAMQ buffer.

6.2.5 ViChaR

Nicopoulos et al [43] introduced a shared dynamic buffer with a variable VC number called Virtual Channel Regulator (ViChaR). Vicar supports a variable number of VC from one to the number of slots in the buffer. The arbiters of ViChaR are implemented to support the extreme case of the number of VCs, that would represent a long critical path that reduces the maximum operating frequency. ViChaR consists of two main components; a storage unit called Unified Buffer Structure (UBS), and a control unit called Unified Control Logic (UCL). UCL module is further subdivided into five sub-modules:

- The Arriving/Departing Flit Pointers Logic: that point to the available slot for the incoming flit and the leaving flit respectively.
- The Slot Availability Tracker: that indicates the available slots.
- The VC Availability Tracker: that indicates the available VCs.
- The VC Control Table: that represent the ViChaR central hub that store information about all in-use VCs and UBS status.
- The Token (VC) Dispenser: that grants the available VCs to the new packets.

ViChaR architecture allows to the number of VCs to vary based on network conditions. This variable number of VCs allows ViChaR to assign a new VC for each incoming packet. Therefore, no multiple packets can share the same VC, which eliminates the case where the blocked packet can block the progress of another packet, HoL blocking.

6.2.6 DVOQR

Dynamic Virtual Output Queues Router (DVOQR) buffer represents the simplest implementation of DAMQ [44]. It consists of three main components; a single Unified Dynamic Buffer (UDB), a Unified Dynamic Buffer Allocator (UDBA), and multiple Virtual Output Address Queues (VOAQ) each for a single VC as shown if figure 6.6. UDB has one write port and n read ports for n VCs. It is implemented using Registers to avoid the delay introduced by SRAM address decoding while the size of buffers needed is relatively small. UDBA consists of a state vector and a fixed priority arbiter. The state vector is used to trace the availability of the slots in UDB. The fixed priority arbiter is used in allocation logic. The fixed priority will simplify the allocation by giving the highest priority to the lowest available slot. VOAQ is a FIFO buffer that holds the addresses of the used slot in UDB. It is used to allocate or deallocate a slot from the UDB. On allocation, the state vector is fed into the fixed priority arbiter to select one of the available slots in UDB. On deallocation, the corresponding bit of the removed slot address, stored in VOAQ, is simply cleared. Figure show the structure of a single queue of VOAQ.

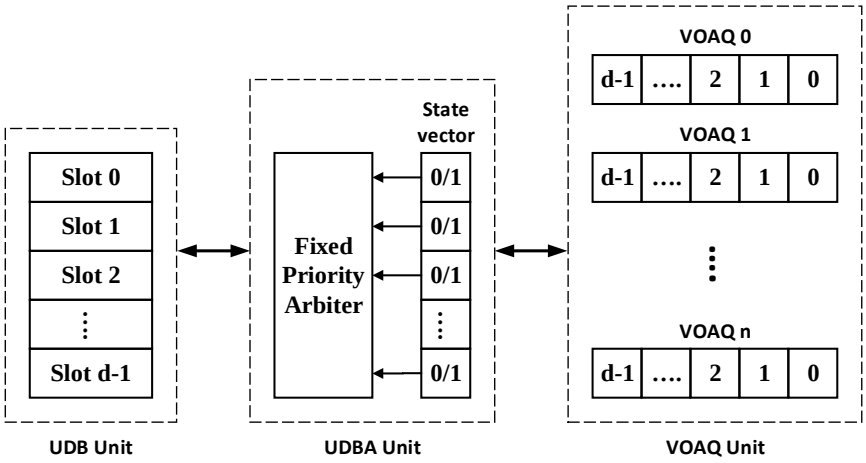


Figure 6.6: DVOQR internal structure

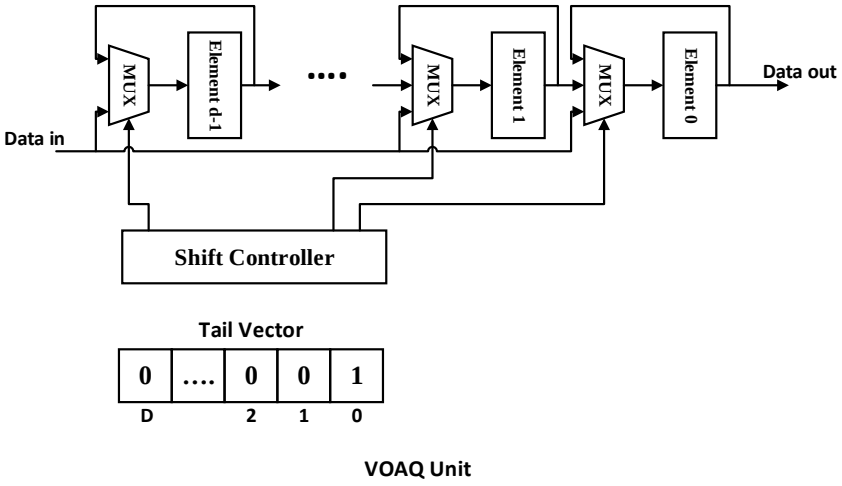


Figure 6.7: The structure of a single queue of VOAQ

When a new flit is required to be added to the buffer, first, the flit is stored in the first available slot in UDB determined by UDBA. Therefore, the address of the new flit is pushed to its corresponding VOAQ. When a new flit is required to be read out of the buffer, its address should be first popped from VOAQ. To reduce the latency of the sequential events, VOAQ is implemented as a shift FIFO where the to be read address is on the top. VOAQ uses a one-hot tail vector to determine the state of the FIFO. The tail vector width is D+1, where D is the depth of VOAQ FIFO which also the number of the slots in DBU. When FIFO is empty only the first bit of the tail vector will be 1. Upon insertion of the new address to the FIFO, the tail vector will be shifted toward the last bit and vice versa.

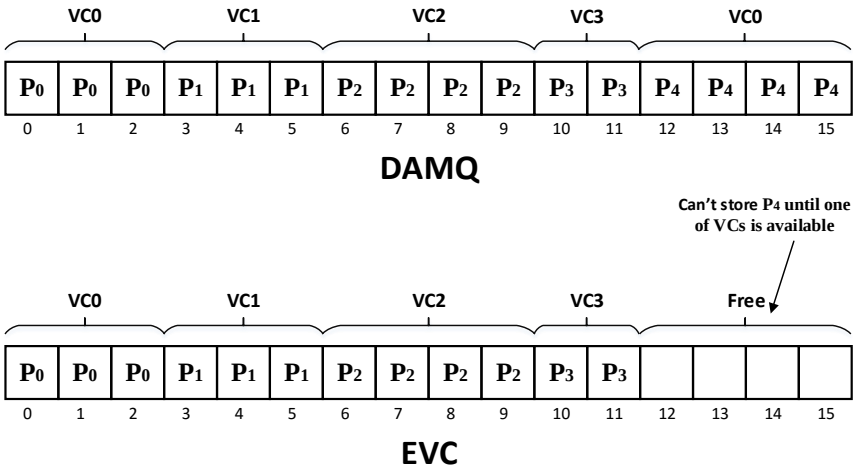


Figure 6.8: An example of EVC storage buffer that supports four VCs compared to a DAMQ buffer.

6.2.7 EVC

The Efficient Virtual Channel organization (EVC) uses the same concept of DAMQ of sharing the buffer, but it eliminates the HOL effect by not allowing for multiple packets to share the same VC [45]. Instead, the first packet will enter the VC queue will prevent the subsequent packets from using the same VC queue. This means that at a certain point, the buffer may have available slots but not used because all VCs are busy serving a single packet. Therefore, the buffer utilization of EVC is lower than the conventional DAMQ.

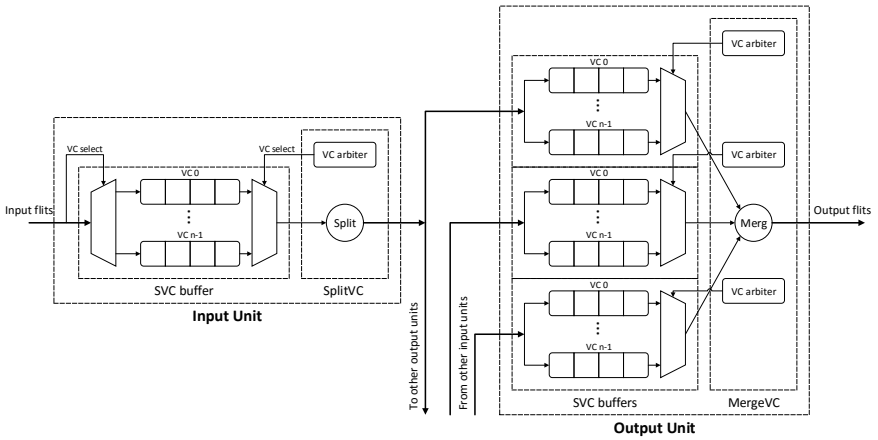
Figure 6.8 shows an example of EVC storage buffer that supports four VCs compared to a DAMQ buffer when storing the same packets.

6.3 Embed Dynamic Virtual Channel in DSM Router

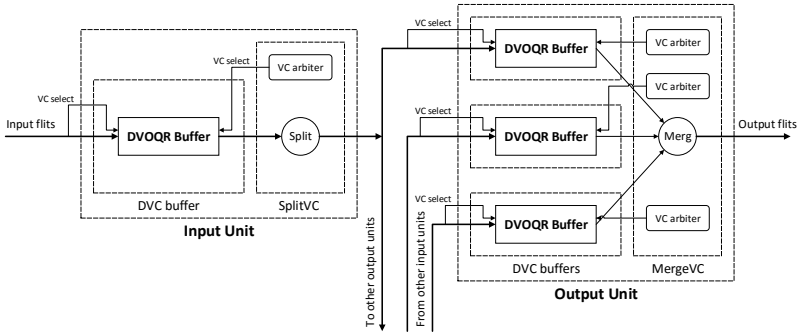
We discussed earlier that adding VCs to DSM has a negative impact on the implementation characteristics, the frequency and the area. In addition, we discussed how to enhance the operating frequency by using PS arbiter instead of the conventional arbiter. In this section, we will focus on elevating the area problem by replacing the static virtual channel buffers with dynamic virtual channel buffers.

6.3.0.1 Architecture

There are several DVC discussed in last section that differ in its performance and ease of implementation. The one we select for implementing DSM buffer is DVOQR. DVOQR is based in primitive linked list scheme in which we should keep track of each slot in the buffer beside the long sequential process of insertion and removal of a buffer slot. DVOQR



a) DSM router with SVC buffers



b) DSM router with DVC buffers

Figure 6.9: The input port and the output port of DSM router with SVC and with DVC.

is expected to has the lowest operating frequency and the largest area, because the memory needed to trace each slot for each VC, among the other DVC found in literature. The only justification of using DVOQR is the ease of the implementation. The implementation of other DVC schemes are future work.

The implementation of DVOQR is discussed in the last section. Figure 6.9 shows the input port and the output port of DSM router with SVC and with DVC.

6.3.0.2 Dynamic Buffers Area Overhead

Beside the control logic of the buffer, the storage memory required for the buffer represent the largest portion of its area. For a static buffer the storage required in bits is given using the following equation:

the storage capacity= $v * d * w$ bits,

where v is the number of virtual channels, d is the depth of the individual VC queue, and w is the flit width.

While for DVOQR the required storage memory in bits is:

the storage capacity = UDB size + the overhead memory size

the storage capacity = UDB size + state vector size + VOAQ size

the storage capacity = $D * w + (D + 1) + v * (D * log_2(D))$

where D is the total dynamic buffer depth.

For the static buffer, the required memory size scales linearly with the flit width. Therefore, the selected flit width doesn't affect much on the comparison with others buffers types. However, for the dynamic memory, increasing the flit width decreases the percentage of the memory overhead. Therefore, selecting a small flit width (8, 16, or 32 bits) the static buffers will be more suitable and for larger flit width the dynamic buffers will give a lower area.

6.3.0.3 Dynamic Buffers Frequency

The dynamic buffer suffers from the long critical path due to the more complicated control logic, compared to static buffer, and the sequential of the events. For example, in DVOQR buffer, the read operation run through fetching the head pointer from VOAQ then reading it out from UDB. This sequential occur of events causes the dynamic buffers to operate at lower frequency relative to static buffers. Therefore, selecting between dynamic and static buffer is a compromise between the frequency and the area.

6.4 Results

This section shows the DVOQR and DSM router with DVC simulation and implementation results.

The implementation is performed over Virtex-5 LX330T FPGA (part xc5vlx330t, speed grade -2) . The implementation results are given by Xilinx ISE 14.6 tool .

6.4.1 DVOQR Implementation Results

Tables 6.1 and 6.2 show the implementation results, the maximum operating frequency and the area, of DVOQR buffer for different depth and number of VCs at 32-bit and 128-bit flit width respectively.

VC	2-VC			
Buffer Depth	8	16	32	64
Frequency (MHz)	334.8	253.8	200.3	204.8
Area (LUTs)	438	1121	2052	4030

VC	4-VC			8-VC	
Buffer Depth	16	32	64	32	64
Frequency (MHz)	232	195.9	196.3	142.6	138.2
Area (LUTs)	1507	2948	11695	5637	11934

Table 6.1: The maximum frequency and area of DVOQR at 32-bit flit width

VC	2-VC			
Buffer Depth	8	16	32	64
Frequency (MHz)	327.9	277.3	225.4	266.1
Area (LUTs)	1K	3K	6K	11K

VC	4-VC			8-VC	
Buffer Depth	16	32	64	32	64
Frequency (MHz)	231.9	195.9	195.4	146.4	137.9
Area (LUTs)	3K	7K	22K	15K	28K

Table 6.2: The maximum frequency and area of DVOQR at 128-bit flit width

6.4.2 DSM Router with DVC Results

In this section, the DSM 2-stage router with 32-bit and 128-bit flit width is used. Although the 16-bit width is used as the default filt width through this thesis, we decided to change it in this section because increasing filt width reduces the percentage of the area overhead of DVC buffer. It is expected that the router with 32-bit flit width and static buffers will give a lower area than the router with dynamic buffers for the same network performance. Similarly, for the router with 128-bit flit width and dynamic buffers will give a lower area than the router with static buffers for the same network performance.

6.4.2.1 32-bit Flit Width Results

This part shows the implementation results for the 32-bit flit width. Table 6.3 shows the implementation results, the maximum operating frequency and the area, of DSM router with DVC over the FPGA platform. Table 6.4 shows the maximum throughput and the ports buffer depth of DSM router when using SVC. The SVC results are based on a constant buffer depth of 16 flits per VC queue.

The results show that the SVC gives a lower area when compared with DVC. Therefore, the recommended buffer scheme for a low flit width (32-bit or less) is the static buffer to avoid the excess area overhead that results from using the dynamic buffer scheme.

VC	2-VC			4-VC			8-VC	
Buffer Depth	16	32	64	16	32	64	32	64
Throughput (flits/cycle/node)	0.67	0.85	0.89	0.72	0.83	0.9	0.84	0.89
Frequency (MHz)	188.4	154.4	155.7	121.6	124.1	84.7	86.5	77.9
Area (LUTs)	12K	29K	44K	22K	44K	83K	86K	170K

Table 6.3: The maximum throughput, maximum frequency and area of DSM router with DVC and 32-bit flit width

VC	2-VC	4-VC	8-VC
Buffer Depth	32	64	128
Throughput (flits/cycle/node)	0.89	0.91	0.92
Frequency (MHz)	170.6	148.8	102.554
Area (LUTs)	19K	38K	78K

Table 6.4: The maximum throughput , maximum frequency and area of DSM router with SVCand 32-bit flit width

6.4.2.2 128-bit Flit Width Results

This part shows the implementation results for the 128-bit flit width. Table 6.5 shows the implementation results, the maximum operating frequency and the area, of DSM router with DVC over the FPGA platforms. Table 6.6 shows the maximum throughput and the ports buffer depth of DSM router when using SVC. The SVC results are based on a constant buffer depth of 16 flits per VC queue. Because of the limited resources we have, limited time and processing power, the FPGA results included in this part, specifically, the operating frequency is taken from the synthesize stage, not after place and route stage, which gives a good approximation of the timing delay but not the best estimation.

The results show that the performance of an SVC based network can be achieved using a DVC based network with significantly less buffer depth and without much degrade in the throughput. For example; the typical port buffer depth for 8-VC static buffers router

VC	2-VC			4-VC			8-VC	
Buffer Depth	16	32	64	16	32	64	32	64
Throughput (flits/cycle/node)	0.67	0.85	0.89	0.72	0.83	0.9	0.84	0.89
Frequency (MHz)	229	187.7	205.8	158.8	150.9	141	102.8	98.9
Area (LUTs)	26K	64K	95K	51K	98K	183K	195K	373K

Table 6.5: The maximum throughput, maximum frequency and area of DSM router with DVC and 128-bit flit width

VC	2-VC	4-VC	8-VC
Buffer Depth	32	64	128
Throughput (flits/cycle/node)	0.89	0.91	0.92
Frequency (MHz)	222.9	202.2	160.2
Area (LUTs)	70K	139K	279K

Table 6.6: The maximum throughput , maximum frequency and area of DSM router with SVC and 128-bit flit width

is 128 flits (16x8) with a throughput equal to 0.92 with area equal to 279K LUTs, while 8-VC dynamic buffers router can give a throughput of 0.84 with a buffer depth of 32 flits and area equals to 195K LUTs which reduces the area used by 30%. Another example is using 4-VC dynamic buffer that gives a throughput of 0.84 with a buffer depth of 32 flits and area equals to 98K LUTs which reduces the area used by 30% when compared to the static buffer which has area equals to 139K, or using a buffer depth of 16 flits when the targeted application has a relatively lower traffic as it gives a throughput of 0.72 flits/cycle/node but yet reduces the are by 63%, however, its is not recommended as we might get the same performance using a static buffer with lower depth per VC.

6.4.3 DSM Router with DVC and PS-arbiter Results

The results of adding PS-arbiter to DSM router to enhance the operating frequency is showed in Priority-Select Arbiter chapter. The enhancement is achieved by replacing the conventional arbiters with PS-arbiters.

We also showed the results of adding dynamic buffers to DSM router in order to reduce the occupied area. However, the reduction of the area was associated with a slight reduction in the operating frequency introduced by the dynamic buffers control logic. This reduction in the operating frequency is unremovable by adding PS-arbiter as it targets only the critical path added by the conventional arbiter. Therefore, the enhancement of the operating frequency of DSM router when equipped by both dynamic buffers and PS-arbiter is limited by the operating frequency of the dynamic buffers. In this part, we show the results of adding both the dynamic buffers and PS-arbiter to DSM router to try to provide some alternative and increasing the operating frequency as much as possible.

Table 6.7 show the implementation results, the maximum operating frequency and the area, of DSM router for different buffer depth and number of VCs at 128-bit flit width.

To avoid confusion, let us call the DSM router with static buffers DSM_SVC, the DSM router with dynamic buffers DSM_DVC, and the DSM router with dynamic buffers and PS-arbiter DSM_DVC_PS.

VC	2-VC			4-VC			8-VC	
Buffer Depth	16	32	64	16	32	64	32	64
Throughput (flits/cycle/node)	0.67	0.85	0.89	0.72	0.83	0.9	0.84	0.89
Frequency (MHz)	248.4	200	211.7	224.4	194.6	199.2	137.5	130.8
Area (LUTs)	38K	64K	95K	53K	105K	183K	196K	372K

Table 6.7: The maximum throughput , maximum frequency and area of DSM router with DVC and PS-arbiter

The results show an improvement in the operating frequency of DSM_DVC_PS while keeping the area reduction we achieved from adding dynamic buffers. For example, DSM_SVC_PS with 8-VC and 32 flits buffer depth keeps the area reduction of 30% when compared with DSM_SVC while showing a frequency increasing by 33.8% when compared with DSM_DVC to achieve a frequency closer to DSM_SVC. The same goes for 4-VC with 32 flits buffer depth were DSM_DVC_PS has area 24.4% less than DSM_SVC and operating frequency 39% more than DSM_DVC to achieve an operating frequency almost equal to DSM_SVC operating frequency.

Moreover, in a lower buffer depth, DSM_DVC_PS shows an improvement in both frequency and area over DSM_DVC and DSM_SVC. For 4-VC with 16 flits buffer depth, DSM_DVC_PS has 61.9% less area than DSM_SVC, and 41.3% and 11% higher frequency than DSM_DVC and DSM_SVC respectively. Similarly, for 2-VC with 16 flits buffer depth, DSM_DVC_PS has 45.7% less area than DSM_SVC, and 8.5% and 11.4% higher frequency than DSM_DVC and DSM_SVC respectively.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this research, we analyzed the NoC parameters and used two NoC implementation for comparisons; ASIC-based NoC called SOTA and FPGA-oriented NoC called CONNECT. For CONNECT, the preferred parameters are two virtual channels, four flits buffer depth for a packet of 2 flits length, 16 node network size, and MESH network topology. For SOTA, the preferred parameters are two virtual channels; eight flits buffer depth for a packet of 2 flits length, 16-node network size, and FOLDED TORUS network topology. CONNECT provides lower latency when compared to SOTA because of the three pipelining stages used in SOTA. In addition, CONNECT gives higher throughput, maximum injection rate, with lower latency in the most cases. That is all beside the area occupied by CONNECT is lower than that occupied by SOTA. All that because CONNECT is designed to target and exploit the FPGA platforms, while SOTA is designed to target ASIC. Which demonstration that there is a big difference between design an FPGA-oriented NoC and just mapping an ASIC-oriented NoC to the FPGA.

Next, we propose a new router architecture, called Dual Spit-Merge, that provides a high performance. DSM basically targets FPGAs and represent the highest throughput among all the existing routers presented in literature without area increment. This all besides its small network latency. The implementation on Virtex-6 FPGA achieves a throughput of 389 MFlits/s/node, that is nearly three to four times the throughput of SOTA router and the FPGA-oriented CONNECT router. DSM also showed a great performance when hard NoC is targeted.

In addition, a comparative review between various round-robin arbiter architectures found in the literature is delivered. A novel efficient RRA is proposed that provides significant improvements over existing RRA architectures. The proposed arbiter provides the highest maximum frequency while occupying nearly half the area of existing arbiter with similar performance for a large number of requesters, e.g. 64. In addition, the results of implementing DSM router VCs using PS arbiter show a boosting in the operating frequency, nearly doubled, with less area.

Moreover, we have augmented DSM router with dynamic virtual channel buffer to reduce the area occupied by the router when implementing virtual channels using static buffers. Although the selected dynamic virtual channel buffer scheme gives the lowest frequency and highest area among the other schemes when implemented stand alone, it showed a significant saving in the area while keeping the same throughput when embedded in DSM router. However, this solution is effective with wide flit width router deside showing a decrease in the operating frequency due to the more complex buffer control.

Finally, we achived a decrease in the area and increase in the operating frequency when using both the proposed RRA and the dynamic buffer at the time.

7.2 Future Work

As an extension to this work, we recommend the following points for the future work:

- Studying other network parameters that effectively contribute to the NoC performance.
- As DSM showed a great performance when implemented in ASIC, we can investigate its performance over a higher dimensions network. This can be achieved by dedicating an internal router for each dimension.
- Supporting adaptive routing algorithms in DSM to make it more suitable for a wide verity of applications that target the FPGA.
- Studying the impact of using the different RRA architectures as the internal RRA of PS arbiter. Furthermore, formulating the relation between the size of internal RRA and the optimum frequency and area.
- Implementing more efficient dynamic virtual channels buffer, other than DVOQR, and studying its impact on DSM performance.

References

[1] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, 2001.

[2] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.

[3] B. A. Abderazek and M. Sowa, "Basic network-on-chip interconnection for future gigascale MCSoCs applications: communication and computation orthogonalization," *TJASSST2006, Dec*, pp. 4–9, 2006.

[4] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *INTEGRATION, the VLSI journal*, vol. 38, no. 1, pp. 69–93, 2004.

[5] F. A. Samman, T. Hollstein, and M. Glesner, "Multicast parallel pipeline router architecture for network-on-chip," in *PROCEEDINGS OF THE CONFERENCE ON DESIGN, AUTOMATION AND TEST IN EUROPE*, ACM, 2008, pp. 1396–1401.

[6] A. Janarthanan, V. Swaminathan, and K. A. Tomko, "MoCReS: an area-efficient multi-clock on-chip network for reconfigurable systems," in *IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI (ISVLSI'07)*, IEEE, 2007, pp. 455–456.

[7] K. A. Helal, S. Attia, T. Ismail, and H. Mostafa, "Comparative review of NoCs in the context of ASICs and FPGAs," in *2015 IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS)*, IEEE, 2015, pp. 1866–1869.

[8] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[9] D. Jayasimha, B. Zafar, and Y. Hoskote, "On-chip interconnection networks: Why they are different and how to compare them," *Platform Architecture Research, Intel Corporation*, 2006.

[10] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[11] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed systems*, vol. 3, no. 2, pp. 194–205, 1992.

[12] R. Mullins, A. West, and S. Moore, “Low-latency virtual-channel routers for on-chip networks,” in *ACM SIGARCH COMPUTER ARCHITECTURE NEWS*, IEEE Computer Society, vol. 32, 2004, p. 188.

[13] C. J. Glass and L. M. Ni, “The turn model for adaptive routing,” *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 278–287, 1992.

[14] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 26, no. 2, pp. 203–215, 2007.

[15] M. S. Abdelfattah and V. Betz, *Augmenting FPGAs with Embedded Networks-on-chip - Semantic Scholar*. [Online]. Available: https://www.semanticscholar.org/paper/Augmenting-Fpgas-with-Embedded-Networks-on-chip-Abdelfattah-Betz/b36a637d456656777655eb85b1f45464078529d3.

[16] D. Lewis and J. Chromczak, “Process technology implications for FPGAs,” in *2012 INTERNATIONAL ELECTRON DEVICES MEETING*, 2012.

[17] C. Chiasson and V. Betz, “Should FPGAs abandon the pass-gate?” In *2013 23RD INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS*, IEEE, 2013, pp. 1–8.

[18] D. U. Becker, “Efficient microarchitecture for network-on-chip routers,” PhD thesis, Stanford University, 2012.

[19] P. P. 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.

[20] É. Cota, A. de Morais Amory, and M. Soares Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-Chip*. Boston, MA: Springer US, 2012, pp. 11–25.

[21] M. K. Papamichael and J. C. Hoe, “CONNECT: re-examining conventional wisdom for designing NoCs in the context of FPGAs,” in *PROCEEDINGS OF THE ACM/SIGDA INTERNATIONAL SYMPOSIUM ON FIELD PROGRAMMABLE GATE ARRAYS*, ACM, 2012, pp. 37–46.

[22] J. Kim, J. Balfour, and W. Dally, “Flattened butterfly topology for on-chip networks,” pp. 172–182, 2007.

[23] R. Pau and N. Manjikian, “Implementation of a configurable router for embedded network-on-chip support in FPGAs,” in *CIRCUITS AND SYSTEMS AND TAISA CONFERENCE, 2008. NEWCAS-TAISA 2008. 2008 JOINT 6TH INTERNATIONAL IEEE NORTHEAST WORKSHOP ON*, IEEE, 2008, pp. 25–28.

[24] Y. Huan and A. DeHon, “FPGA optimized packet-switched NoC using split and merge primitives,” in *FIELD-PROGRAMMABLE TECHNOLOGY (FPT), 2012 INTERNATIONAL CONFERENCE ON*, IEEE, 2012, pp. 47–52.

[25] N. Kapre, N. Mehta, R. Rubin, H. Barnor, M. J. Wilson, M. Wrighton, A. DeHon, *et al.*, "Packet switched vs. time multiplexed FPGA overlay networks," in *2006 14TH ANNUAL IEEE SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES*, IEEE, 2006, pp. 205–216.

[26] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif, and C. R. Das, "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 2, pp. 4–15, 2006.

[27] S. T. Nguyen and S. Oyanagi, "A low cost single-cycle router based on virtual output queuing for on-chip networks," in *DIGITAL SYSTEM DESIGN: ARCHITECTURES, METHODS AND TOOLS (DSD), 2010 13TH EUROMICRO CONFERENCE ON*, IEEE, 2010, pp. 60–67.

[28] S. Yoo, G. Nicolescu, D. Lyonnard, A. Baghdadi, and A. A. Jerraya, "A generic wrapper architecture for multi-processor SoC cosimulation and design," in *PROCEEDINGS OF THE NINTH INTERNATIONAL SYMPOSIUM ON HARDWARE/-SOFTWARE CODESIGN*, ACM, 2001, pp. 195–200.

[29] M. Abdelrasoul, M. Ragab, and V. Goulart, "Impact of Round Robin Arbiters on router's performance for NoCs on FPGAs," in *CIRCUITS AND SYSTEMS (ICCAS), 2013 IEEE INTERNATIONAL CONFERENCE ON*, IEEE, 2013, pp. 59–64.

[30] B. Zhao, Y. Zhang, and J. Yang, "A speculative arbiter design to enable high-frequency many-VC router in NoCs," in *NETWORKS ON CHIP (NOCS), 2013 SEVENTH IEEE/ACM INTERNATIONAL SYMPOSIUM ON*, Apr. 2013, pp. 1–8.

[31] P. Gupta and N. McKeown, "Designing and Implementing a Fast Crossbar Scheduler," *IEEE Micro*, vol. 19, no. 1, pp. 20–28, Jan. 1999.

[32] G. Dimitrakopoulos, N. Chrysos, and K. Galanopoulos, "Fast arbiters for on-chip network switches," in *COMPUTER DESIGN, 2008. ICCD 2008. IEEE INTERNATIONAL CONFERENCE ON*, Oct. 2008, pp. 664–670.

[33] K. A. Helal, S. Attia, T. Ismail, and H. Mostafa, "Priority-select arbiter: An efficient round-robin arbiter," in *NEW CIRCUITS AND SYSTEMS CONFERENCE (NEWCAS), 2015 IEEE 13TH INTERNATIONAL*, IEEE, 2015, pp. 1–4.

[34] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks," in *PROCEEDINGS OF THE 2003 INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN*, ACM, 2003, pp. 90–95.

[35] T. T. Ye, G. D. Micheli, and L. Benini, "Analysis of power consumption on switch fabrics in network routers," in *PROCEEDINGS OF THE 39TH ANNUAL DESIGN AUTOMATION CONFERENCE*, ACM, 2002, pp. 524–529.

[36] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *DESIGN AUTOMATION CONFERENCE, 2001. PROCEEDINGS*, IEEE, 2001, pp. 684–689.

[37] G. Varatkar and R. Marculescu, "Traffic analysis for on-chip networks design of multimedia applications," in *DESIGN AUTOMATION CONFERENCE, 2002. PROCEEDINGS. 39TH*, IEEE, 2002, pp. 795–800.

[38] J. Hu and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in *PROCEEDINGS OF THE 2004 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN*, IEEE Computer Society, 2004, pp. 354–361.

[39] M. Rezazad and H. Sarbazi-Azad, "The effect of virtual channel organization on the performance of interconnection networks," in *19TH IEEE INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM*, IEEE, 2005, 8–pp.

[40] G. L. Frazier and Y. Tamir, "The design and implementation of a multiqueue buffer for VLSI communication switches," in *COMPUTER DESIGN: VLSI IN COMPUTERS AND PROCESSORS, 1989. ICCD'89. PROCEEDINGS., 1989 IEEE INTERNATIONAL CONFERENCE ON*, IEEE, 1989, pp. 466–471.

[41] J. Park, B. W. O'Krafka, S. Vassiliadis, and J. Delgado-Frias, "Design and evaluation of a DAMQ multiprocessor network with self-compacting buffers," in *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, IEEE Computer Society Press, 1994, pp. 713–722.

[42] J. Liu and J. G. Delgado-Frias, "DAMQ Self-Compacting Buffer Schemes for Systems with Network-On-Chip.," in *CDES*, 2005, pp. 97–103.

[43] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A dynamic virtual channel regulator for network-on-chip routers," in *2006 39TH ANNUAL IEEE/ACM INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE (MICRO'06)*, IEEE, 2006, pp. 333–346.

[44] S. Qi, M. Zhang, J. Li, T. Zhao, C. Zhang, and S. Li, "A high performance router with dynamic buffer allocation for on-chip interconnect networks," in *COMPUTER DESIGN (ICCD), 2010 IEEE INTERNATIONAL CONFERENCE ON*, IEEE, 2010, pp. 462–467.

[45] M. O. Gharan and G. N. Khan, "Dynamic Virtual Channel Configuration for Efficient Multicore Systems," in *COMPLEX, INTELLIGENT AND SOFTWARE INTENSIVE SYSTEMS (CISIS), 2014 EIGHTH INTERNATIONAL CONFERENCE ON*, IEEE, 2014, pp. 445–450.

ملخص الرسالة

أدى التقدم في تكنولوجيا أشباه الموصلات إلى رقاقة ذات كثافة عالية والذي بدروه يحرك عنق الزجاجة من أنظمة الحساب على الرقائق الإلكترونية الى أنظمة الاتصالات على الرقائق الإلكترونية. هذا التقدم يعطي مصفوفات البوابات المنطقية القابلة للبرمجة فرصة للتنافس مع الدوائر المتكاملة محددة التطبيق، وذلك من خلال زيادة عناصر المعالجة المدمجة المختلفة بالإضافة الى بنية ربط مرنة. ومع ذلك، فشلت نماذج الاتصال التقليدية في تلبية احتاج نظم الرقائق الإلكترونية لسرعة عالية، ونطاق اتصال عالي، وموارد مهدرة منخفضة، ومعيارية في التصميم. تعتبر شبكات الرقائق الإلكترونية حلاً واعداً لمواجهة تحديات الاتصالات على الرقائق الإلكترونية المرتبطة مع التقدم التكنولوجي. في هذه الرسالة سوف ندرس تطوير شبكات الرقائق الإلكترونية عالية الأداء التي تستهدف البوابات المنطقية القابلة للبرمجة.

هذه الرسالة تقارن وتناقش عوامل البوابات المنطقية القابلة للبرمجة التي تعطي الأداء العالي والكفاءة لشبكات الرقائق الإلكترونية الخاصة بالبوابات المنطقية القابلة للبرمجة ولشبكات الرقائق الإلكترونية الخاصة بالدوائر المتكاملة محددة التطبيق. وبالإضافة إلى ذلك، فإن الرسالة تقدم بينية جديدة لجهاز توجيه شبكات الرقائق الإلكترونية الخاصة بالبوابات المنطقية القابلة للبرمجة والتي تتفوق على البنيات السابقة. وعلاوة على ذلك، تقترح هذه الرسالة بنية تنفيذية جديدة لمحكم راوند–روبن والتي تقلل بشكل كبير المساحة وتزيد من تردد التشغيل بالمقارنة مع سابقاتها. هذا المحكم يعزز إلى حد كبير تردد التشغيل عند ادماجه مع بنية جهاز توجيه شبكات الرقائق الإلكترونية المقترح. وأخيراً، فإن الرسالة تجهز جهاز توجيه شبكات الرقائق الإلكترونية المقترح بواحدة من أنظمة الذاكرة الديناميكية للحد من الزيادة في المساحة المستخدمة.

مهندس: خالد عبدالله هلال
تاريخ الميلاد: ١٩٨٩/٠٦/١٦
الجنسية: مصري
تاريخ التسجيل: ٢٠١٣/١٠/٠١
تاريخ المنح: ٢٠١٦/.../....
الدرجة: ماجستير العلوم
القسم: هندسة الإلكترونيات والاتصالات الكهربية

المشرفون:

أ.د. حسام علي حسن فهمي
د. حسن مصطفى

الممتحنون:

أ.د. حسام علي حسن فهمي (المشرف الرئيسي)
أ.د. أمين محمد نصار (الممتحن الداخلي)
أ.د. مهاب حسين أنيس، كلية الهندسة – الجامعة الأمريكية بالقاهرة (الممتحن الخارجي)

عنوان الرسالة:

جهاز توجيه ومحكم لشبكات الرقائق الالكترونية الخاصة بمصفوفة البوابات المنطقية القابلة للبرمجة

الكلمات الدالة:
جهاز توجيه، محكم، شبكات الرقائق الالكترونية، مصفوفة البوابات المنطقية القابلة للبرمجة

ملخص الرسالة:
أدى التقدم في تكنولوجيا أشباه الموصلات إلى رقاقة ذات كثافة عالية والذي بدروه يحرك عنق الزجاجة من أنظمة الحساب على الرقائق الإلكترونية الى أنظمة الاتصالات عليها. هذا التقدم يعطي مصفوفات البوابات المنطقية القابلة للبرمجة فرصة للتنافس مع الدوائر المتكاملة محددة التطبيق. ومع ذلك، فشلت نماذج الاتصال التقليدية في تلبية احتياجات نظم الرقائق الإلكترونية والتى تعتبر شبكات الرقائق الإلكترونية حلاً واعداً لها. في هذه الرسالة تدرس تطوير شبكات الرقائق الإلكترونية عالية الأداء التي تستهدف البوابات المنطقية القابلة للبرمجة.

جهاز توجيه ومحكم لشبكات الرقائق الالكترونية الخاصة بمصفوفة البوابات المنطقية القابلة للبرمجة

اعداد

خالد عبدالله هلال

رسالة مقدمة الي
كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول علي درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

يعتمد من لجنة الممتحنين:

أ.د. حسام علي حسن فهمي – المشرف الرئيسي

أ.د. أمين محمد نصار – الممتحن الداخلي

أ.د. مهاب حسين أنيس – الممتحن الخارجي
كلية الهندسة – الجامعة الأمريكية بالقاهرة

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٦

جهاز توجيه ومحكم لشبكات الرقائق الالكترونية الخاصة بمصفوفة البوابات المنطقية القابلة للبرمجة

اعداد

خالد عبدالله هلال

رسالة مقدمة الي
كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول علي درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

تحت إشراف

أ.د. حسام علي حسن فهمي
أستاذ
قسم هندسة الإلكترونيات والاتصالات الكهربية
كلية الهندسة – جامعة القاهرة

د. حسن مصطفى
مدرس
قسم هندسة الإلكترونيات والاتصالات الكهربية
كلية الهندسة – جامعة القاهرة

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٦

جهاز توجيه ومحكم لشبكات الرقائق الالكترونية الخاصة بمصفوفة البوابات المنطقية القابلة للبرمجة

اعداد

خالد عبدالله هلال

رسالة مقدمة الي
كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول علي درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٦