Cairo University

OPTIMIZING FPGA-BASED HARD NOCS

By

Sameh Attia Ahmed Attia

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

OPTIMIZING FPGA-BASED HARD NOCS

By

Sameh Attia Ahmed Attia

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

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

OPTIMIZING FPGA-BASED HARD NOCS

By

Sameh Attia Ahmed Attia

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
(The American University in Cairo)

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

Engineer's Name:	Sameh Attia Ahmed Attia
Date of Birth:	01/10/1991
Nationality:	Egyptian
E-mail:	sameh.a.attia@ieee.org
Phone:	+201002920361
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, The American University in Cairo
(External examiner)

Title of Thesis:

OPTIMIZING FPGA-BASED HARD NOCS

Key Words:
NoC; FPGA; Router; BlockRAM

Summary:
In this thesis, we evaluate various NoC design parameters to find the best-fit parameters that can be used in the nonconfigurable hard NoC. We also evaluate different router architectures to select the optimum one for hard NoCs. The designed router reduces the wasted area by using minimum and shareable resources. Moreover, we propose an efficient novel way for embedding the hard NoC inside the FPGA. The proposed NoC provides the FPGA with a high performance communications infrastructure at a negligible cost.

Acknowledgments

First and foremost, I am thankful to ALLAH for giving me the strength and perseverance to complete this work.

I would like to thank my advisors, Dr. Hossam Fahmy, and Dr. Hassan Mostafa for their guidance, help, encouragement and support. I am very grateful to them for helping me in my first steps in my academic career.

Moreover, I must express my profound gratitude to my parents and to my sister for providing me with endless support and continuous encouragement throughout my years of study. I cannot thank them enough.

I am deeply indebted to my wife, Marwa, for her unfailing support and unconditional care and love throughout the process of researching and writing this thesis. This accomplishment would not have been possible without her.

Finally, I would like to thank my friends and my colleagues at Cairo University for making life easier and more enjoyable.

Dedication

To

My Father,

My Mother,

Shereen,

Marwa,

and

Laila

Table of Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

Abbreviations	Description
ASIC	Application-Specific Integrated Circuit.
BLE	Basic Logic Element.
BP	Back Pressure.
CMOS	Complementary Metal Oxide Semiconductor.
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.
NRE	Non Recurring Expenses.
PDR	Partial Dynamic Reconfiguration.
PE	Processing Element.
RAM	Random Access Memory.
RTL	Register Transfer Level.
SoC	System on Chip.
VC	Virtual Channel.
VOQ	Virtual Output Queueing.

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,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1866–1869, May 2015.

[2] K. Helal, S. Attia, T. Ismail, and H. Mostafa, “Priority-select arbiter: An efficient round-robin arbiter,” *IEEE International Conference on NEW Circuits and Systems (NEWCAS)*, pp. 1–4, Jun. 2015.

Abstract

FPGA has become a favorable platform for Systems-on-Chip (SoC). As SoC gets larger, a Network-on-Chip (NoC) emerges as a promising solution for communications problems between SoC's modules. Thus, the importance of NoC on FPGAs has increased, not only to solve SoC's communications problems but also as a solution to FPGA's slow interconnects and to simplify Partial Dynamic Reconfiguration (PDR). Hard NoCs have better performance and consume less area and power than Soft NoCs. However, they are not configurable and they lead to a wasted area when the network is not in use. This makes the design of hard NoCs more critical in order to get an optimum performance while minimizing the wasted area as much as possible. In this thesis, we evaluate various NoC design parameters to find the best-fit parameters that can be used in the nonconfigurable hard NoC. We also evaluate different router architectures to select the optimum one for hard NoCs. The designed router reduces the wasted area by using minimum and shareable resources. Moreover, we propose an efficient novel way for embedding the hard NoC inside the FPGA. The proposed NoC provides the FPGA with a high performance communications infrastructure at a negligible cost.

Chapter 1

Introduction

1.1 Motivation

The extensive increase in number of transistors that can fit in one chip has enabled large scale Systems-on-Chip (SoC). In SoC, a single chip can contain a large number of IP modules, memories, and embedded processors. With technology scaling, SoC systems become more complex and they can accommodate more and more modules. As the number of modules increases, the commonly used shared-bus interconnect becomes a source of performance congestion. This is due to the increase in parasitic capacitance and resistance on the bus and the complexity of arbitration.

Moreover, the interconnect delays do not scale down with technology scaling by the same rate the gate delays do, especially global wiring delays which may increase exponentially. In ultra-deep sub-micron technology, eighty percent of the critical path delay comes from the interconnect delays [1]. All of these move the delay bottleneck from the computation to the communication. Therefore, a lot of research has been done to solve the interconnects problem in large scale SoCs and find an alternative for the shared-bus interconnect.

The idea of NoC emerged in 2001 by William Dally when he proposed replacing the dedicated global wires of chips with a general interconnection network. Long wires are replaced by short links connecting between routers. The various blocks on the chip communicate with each other by "routing packets not wires". This organization is characterized by high performance and high regularity that decreases the communication complexity of SoCs [2].

Due to their increasing capacity and their configurability, FPGAs have become an attractive platform for wide range of SoCs [3]. This requires also a high performance interconnection network between the tens/hundreds SoC modules. Moreover, the current FPGA's interconnection architecture has many shortcomings that adversely affect the performance, limit the speed of the FPGA, and increase the need of higher level communication protocol within the FPGA.

The first drawback of the current FPGA programmable interconnects is that they are getting worse with technology scaling due to the decreasing performance of pass transistors, used in interconnects multiplexers, and the increasing resistance of metal wires [4, 5]. Thus, the delay is dominated by the interconnects. This leads to a second problem that you can not predict the critical paths using only the functional description as they are mostly dependable on the wiring delays, leading to a more complex design cycle. Moreover, hard blocks and I/O interfaces, such as DDR, Ethernet and PCI, work at a much higher speed than the FPGA fabric, so a very wide datapath is needed to accommodate this bandwidth which uses a lot of interconnect and logic resources. Finally, the current low level abstraction of FPGA interconnects prevents the division of a design into modules that can be independently optimized and compiled.

Consequently, a NoC as an overlay network on the top of the FPGA interconnects emerges as a promising solution. NoCs improve wire utilization and simplifies timing closure. The data of fast I/O interfaces can be distributed among the FPGA without the need of excessive wires [6]. Also, the high level abstraction of NoC eases the division of the design into modules [2]. This simplifies partial dynamic reconfiguration (PDR) [7]. In PDR, a module is replaced by another module during run-time. When using NoC, the newly configured module needs to be connected only to an NoC interface. This eliminates the need to route its connections within the whole design.

FPGA-based NoCs have two types: Soft NoCs [8–10] and Hard NoCs [7, 11–15]. Soft NoCs are implemented by end users using the conventional FPGA programmable resources. On the other hand, hard NoCs are embedded blocks which are implemented on silicon like other FPGA dedicated blocks such as embedded RAMs, multipliers and embedded processors. Hard NoCs have several advantages over soft NoCs [6], discussed in Chapter 2.

When firstly the idea of hard NoCs emerged, the main proposal was to divide the FPGA into regions and connect these regions together with a hard NoC. This idea completely alters the conventional routing architecture of FPGA. It does not suit all FPGA applications and it requires great modifications in CAD tools. Consequently, this idea was not implemented commercially, and until now, most commercial FPGAs utilize the conventional island-style architecture.

The idea of hard NoCs has recently emerged again but this time without any modifications to the conventional FPGA routing architecture. The main proposal is to keep the original routing resources “the roads”, but add a fast high-level communications layer “highways” [16]. The most recent research on embedding hard NoCs is carried out in University of Toronto [6, 15–19].

However, hard NoCs have some drawbacks. They are not configurable like soft NoCs. Moreover, when the NoC is not used, the area consumed by hard routers are wasted area.

In this thesis, we try to overcome these problems and optimize FPGA-based hard NoCs. This involves finding the best-fit NoC parameters and router architecture. We try also to reduce the wasted area by minimizing and sharing resources. We also propose an efficient novel way of embedding hard NoCs inside the FPGA fabric.

1.2 Thesis Contributions

This thesis makes the following contributions:

- Provides design recommendations for selecting NoC parameters.
- Finds, optimizes and implements best-fit router architecture for FPGA-based hard NoCs.
- Proposes an efficient novel way for embedding hard NoCs.

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

- Chapter 2 presents background information about FPGAs and NoCs as well as a survey of the previous soft and hard NoCs.
- Chapter 3 studies the effect of various NoC parameters on NoC performance. It provides design recommendations for the best-fit NoC parameters. This work is published in ISCAS 2015.
- Chapter 4 selects the router architecture and optimizes it according to the selected NoC parameters.
- Chapter 5 describes the design of the network interface and proposes some ideas about embedding the hard NoC inside the FPGA fabric.
- Chapter 6 evaluates the performance of the proposed hard NoC and compares it with previous NoCs presented in literature.
- A conclusion and future work are discussed in Chapter 7.

Chapter 2

Background and Literature Review

2.1 FPGA

2.1.1 Definition

A Field Programmable Gate Array (FPGA) is a special type of integrated circuits (ICs) that can be electrically programmed several times to implement different digital systems. This is in contrast with Application Specific Integrated Circuits (ASIC) technologies such as Standard-Cells or Full-Custom that are fabricated for a specific function only. To allow this programmability, FPGA comprises logic elements and routing paths that are highly configurable to do any function.

A logic element mainly consists of an N-input Lookup-table (LUT) and a Flip-Flop as shown in Fig. 2.1. An N-input LUT is capable of implementing any combinational function that has N inputs. A number of basic logic elements (BLEs) are grouped together to form a logic cluster [20]. FPGAs nowadays contain thousands of logic clusters.

Routing channels and switch boxes are needed to provide the connections between logic clusters. All the routing resources are configurable to create any possible connection between the logic clusters. For speed and area issues, routing channels vary in length. Some routing channels are short connecting adjacent logic clusters, and they are called local routing channels. Global routing channels are long, spanning multiple logic blocks. The switch box provide the connection between the horizontal and vertical routing channels to form the signal path as shown in Fig. 2.2. To reduce its area and its complexity, the switch box does not allow all the connections between the routing channels; only subset of these connections is supported [21]. Fig. 2.3 shows three different switch box topologies.

The configurablity of logic elements and routing resources are provided through a programming technology such as static memory (SRAM), EEPROM, flash and anti-fuse [22]. SRAM is the most widely used technology as it utilizes the standard CMOS technology. SRAM cells are distributed across the whole FPGA. For example, they are used as the selection lines of the multiplexers that drive the routing channels. They are also used to store the data in LUTs. Upon FPGA startup, all SRAM cells are loaded with a bitstream that program the FPGA to implement a certain function. The bitstream loading

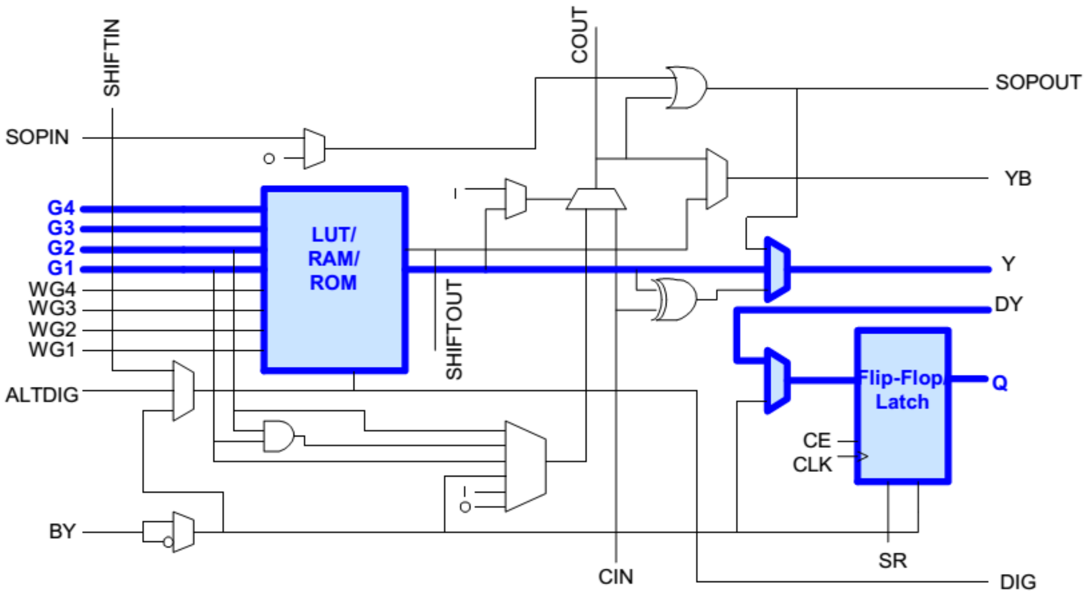


Figure 2.1: An FPGA logic element

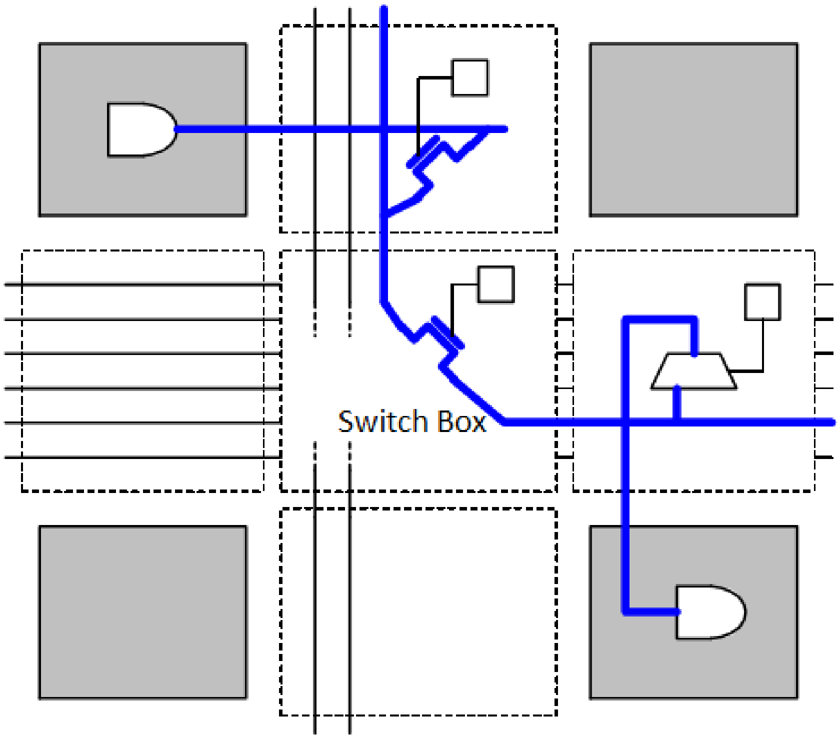


Figure 2.2: A signal path between logic clusters

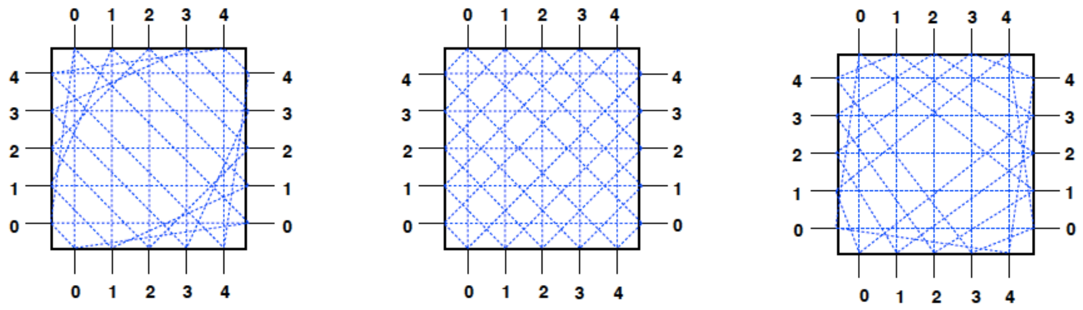


Figure 2.3: Switch box topologies: Disjoint, Universal and Wilton

process is called FPGA configuration. A bitstream is generated by a CAD tool that takes the Hardware Description Language (HDL) netlist and converts it to a bitstream targeting a specific FPGA.

Beside logic blocks and routing resources, modern FPGAs contain also hard (non-configurable) blocks that provide high performance functionality such as multipliers, embedded memories (Block RAMs), transceivers, clock managers, I/O blocks and processors. Block RAMs are high density memory that are used for efficient and high performance data storage or buffering.

In modern FPGAs, all the aforementioned blocks, soft and hard blocks, are arranged in a 2-D array as shown in Fig. 2.4. This arrangement utilizes the island-style routing architecture which provides high regularity. The island-style architecture is widely used in commercial FPGAs. For example, Fig. 2.5 shows the floorplan of Altera Stratix-5 FPGA showing the 2-D array of logic clusters along with hard blocks such as memories and DSP blocks. Altera and Xilinx are the two leading company in FPGA manufacturing.

2.1.2 FPGA advantages over ASIC:

FPGA has many advantages over ASIC. We can summarize these advantages in three main points: reconfigurability, time-to-market and cost.

2.1.2.1 Reconfigurability

The most important difference between FPGA and ASIC is the reconfigurability. That means that it can be reprogrammed indefinite times to implement various digital circuits. While ASIC is manufactured to do a specific function only. This makes FPGA a flexible platform that can be easily used for prototyping and embedded systems. Moreover, upgrading a FPGA-based product to a newer version can easily be done by reprogramming the FPGA, while the upgrading is impossible if it is an ASIC-based product. Also,

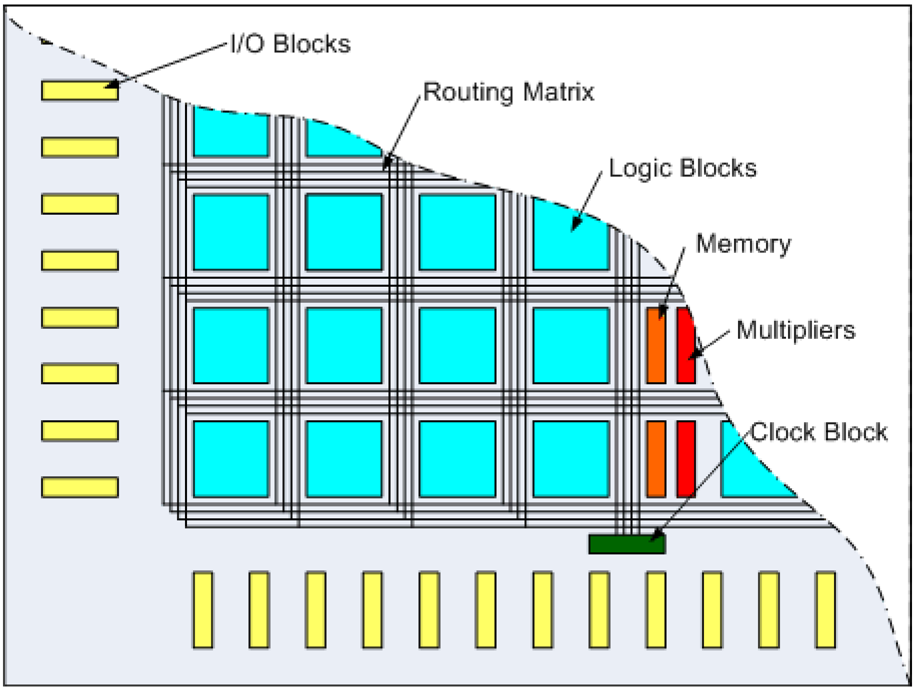


Figure 2.4: Island-style FPGA architecture

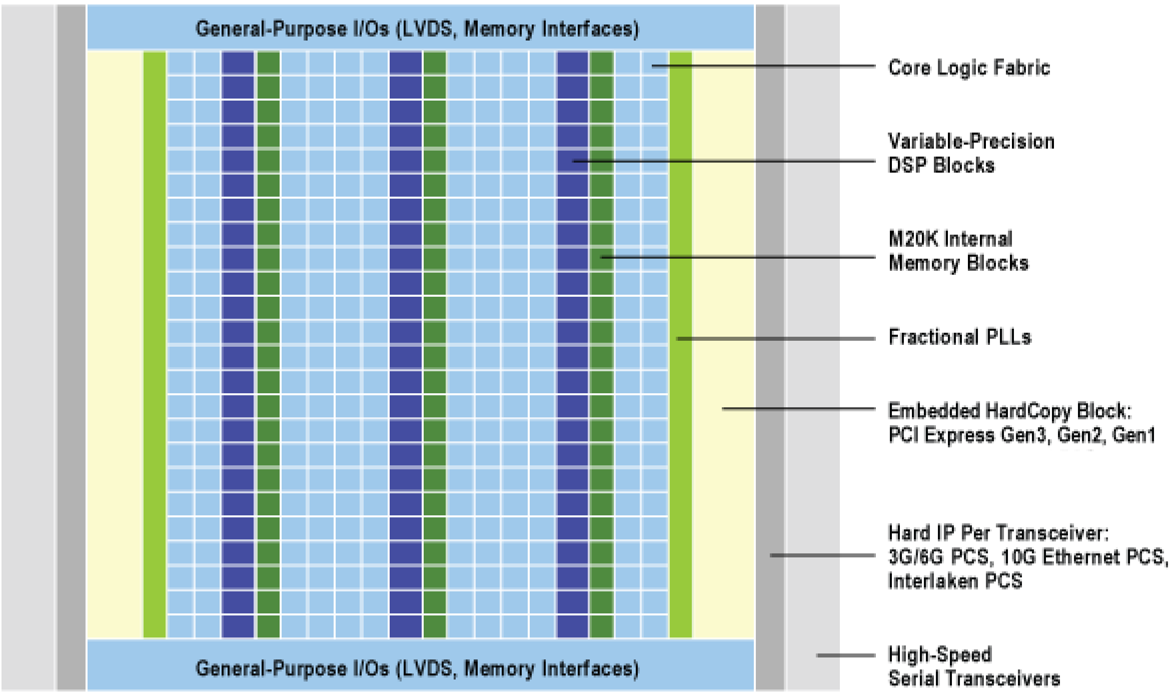

General-Purpose I/Os (LVDS, Memory Interfaces)
Core Logic Fabric
Variable-Precision DSP Blocks
M20K Internal Memory Blocks
Fractional PLLs
Embedded HardCopy Block: PCI Express Gen3, Gen2, Gen1
Hard IP Per Transceiver: 3G/6G PCS, 10G Ethernet PCS, Interlaken PCS
High-Speed Serial Transceivers
General-Purpose I/Os (LVDS, Memory Interfaces)

Figure 2.5: Altera Stratix V floorplan

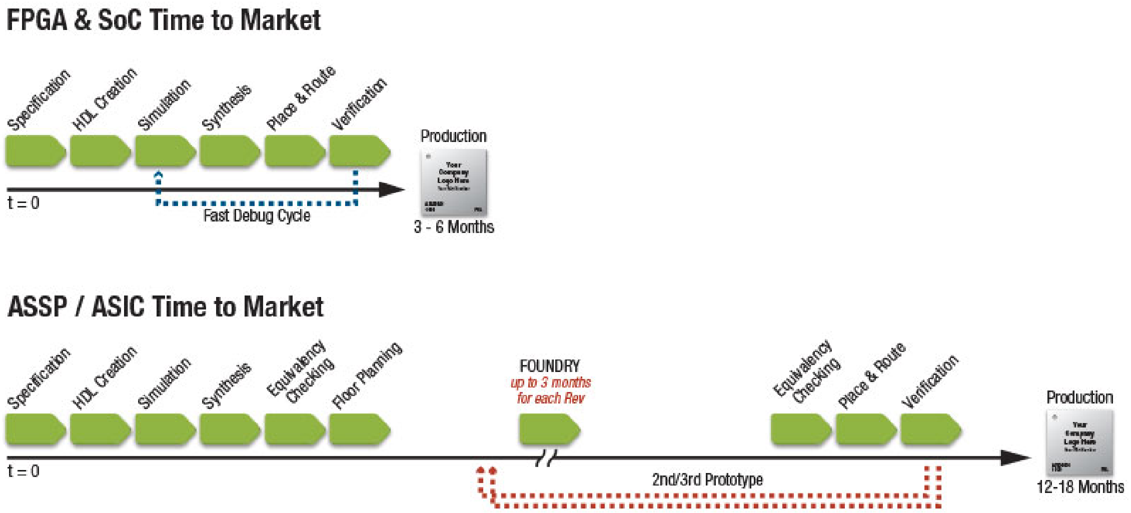


Figure 2.6: FPGA and ASIC time-to-market [23]

design's errors are unrecoverable in ASIC. This is in contrast with FPGA, due to its reconfigurability, which permits late design changes.

2.1.2.2 Time to market

FPGA has faster time-to-market than ASIC as shown in Fig. 2.6 for many reasons. ASIC fabrication requires masks generation and many fabrication steps that may take more than three months. While FPGA programming is an instant process. Also, FPGA has a shorter and simpler design cycle. This is because of CAD tools that handle much of the design flow such as routing, placement and timing. Moreover, ASIC design cycle involves a long period of validation and verification as it is very unforgiving process. Besides shorter design cycle, FPGA provides more predictable project cycle because of the elimination of ASIC re-spins. ASIC re-spins can delay the project by about two months for each re-spin.

2.1.2.3 Cost

One of the most crucial issues in deciding whether to implement the design on FPGA or ASIC is the cost. ASIC has a high non-recurring expenses (NRE). NRE costs such as mask costs have increased exponentially with technology nodes [24]. This is one of the greatest barrier to a profitable ASIC-based business especially for a low volume production. Fig. 2.7 shows the dramatic increase in ASIC process costs with each technology advance. Besides high NRE, any error in an ASIC design is considered as a disaster that costs a lot of money. Also, ASIC re-spins not only take a lot of time but also consume money. On the contrary, FPGA has almost no NRE. An error can be fixed at zero cost. ASIC may have a lower unit cost at high volume production. But the question is where is the break-even point? As technology advances, this point is moving away because of

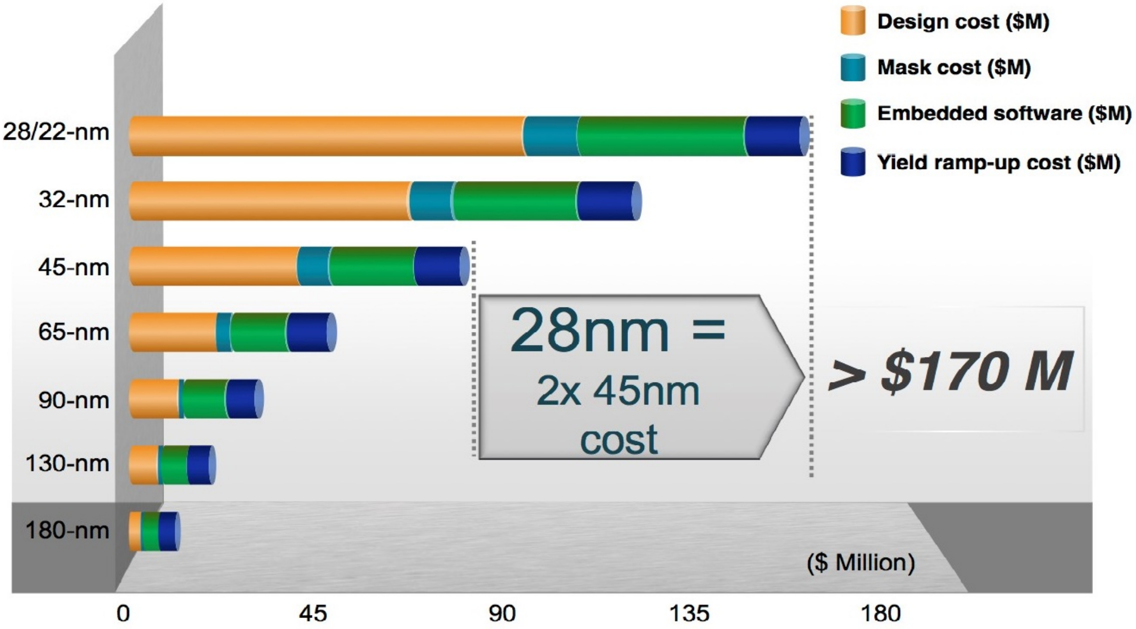


Figure 2.7: ASIC process cost [25]

the exponential increase in ASIC NRE [24]. This means that FPGA has become a very promising solution economically except for extremely high volume production. Thus, FPGA offers a higher technology at a lower cost.

2.2 Network-on-Chip

2.2.1 Definition

The NoC architecture consists of processing elements (PEs), routers, links and network interfaces (NIs) as shown in Fig. 2.8. Each PE is connected to the network through the NI that divides the data into packets and forwards them to the router. The router then routes the packets through the network to their destination.

2.2.2 Network Parameters

2.2.2.1 Network Topology

The topology is the way the routers are physically connected to each other. Choosing a topology is the first step in designing a network. It affects the selection of the routing algorithm and flow-control. It can be regular or irregular. Irregular topologies are formed of mixed types of regular ones. Examples of regular topologies, shown in Fig. 2.9, are mesh, torus, butterfly, tree, polygon and star.

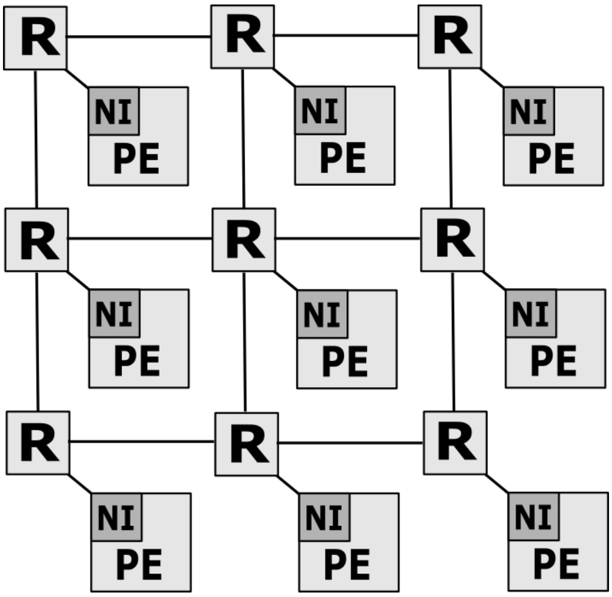


Figure 2.8: NoC architecture

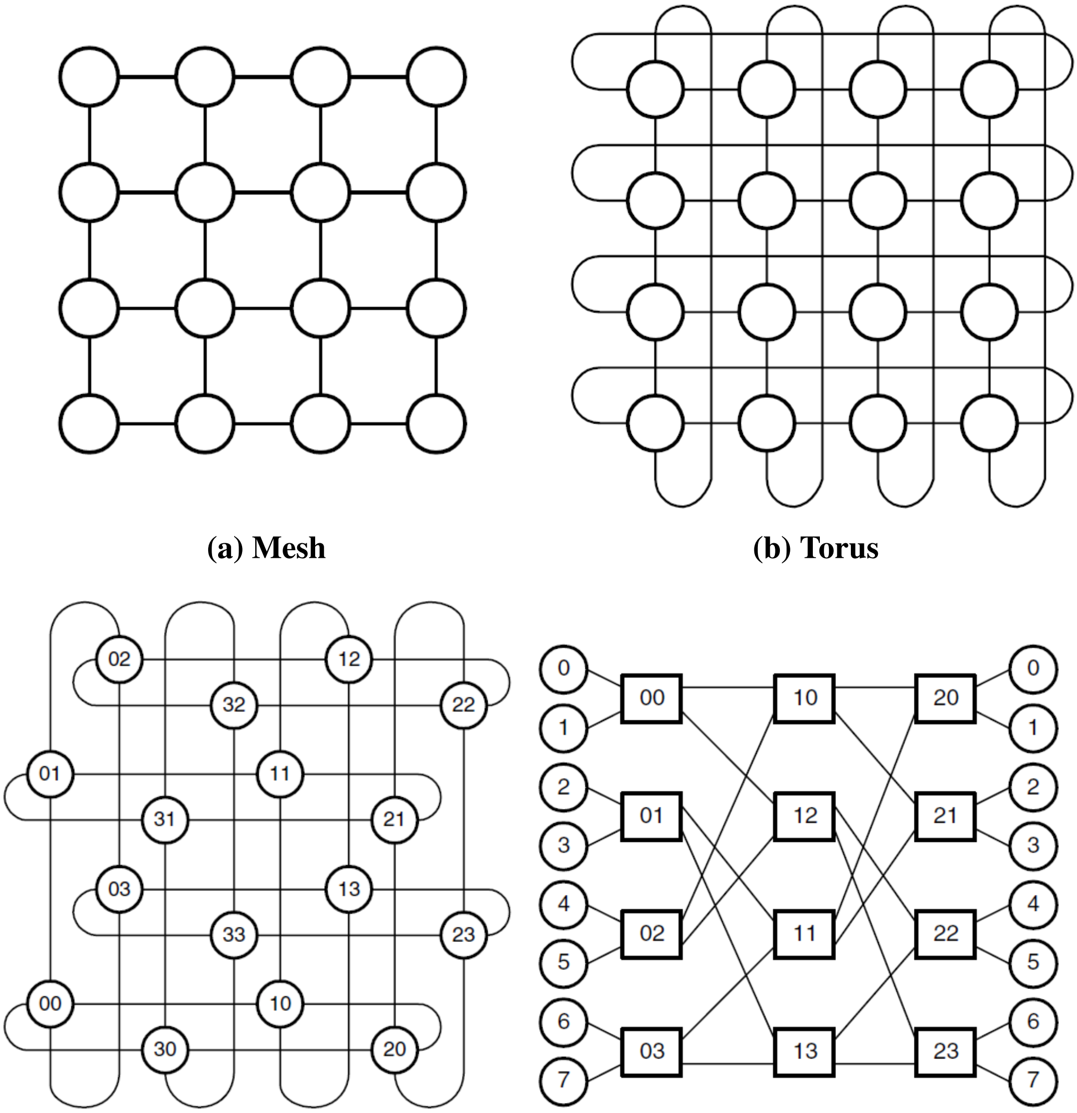
2-D mesh and torus are the most popular topologies for NoC. A mesh topology consists of rows and columns in which the routers are placed at their intersections. The routers' addresses are defined easily by X-Y coordinates. A torus topology resembles a mesh but the two ends of a row or column are connected together.

2.2.2.2 Routing algorithm

The routing algorithm determines the path taken by the packet from the source until it reaches the required destination. It can be deterministic or adaptive. In deterministic routing, the route taken is based only on the location of the source and destination. While in adaptive routing, there are many legal paths between the source and destination, and the choice of the path is determined dynamically based on link performance and congestion. In such algorithm, the complexity of the router increases as it has to decide dynamically the routing direction but it improves the performance under high load condition.

A deterministic routing can be table-driven routing or source-routed. In source routing, the source provides the whole routing path in the packet header so it simplifies the decision of the routing direction in the router. But the number of hops will be limited if the packet header has a fixed length. In table-driven algorithm, Each router will have its routing table that it decides the direction on it. This supports unlimited number of hops but it needs more area.

XY routing is a famous example of deterministic routing algorithm. It suites regular topologies such as 2-D mesh. It is a dimension-order routing; The packet is routed fully in the x-direction and then fully in y-direction. The packet header provides information about the remaining distance for each dimension, so that the routing direction is determined



(a) Mesh

(b) Torus

(c) Folded Torus

(d) Butterfly

Figure 2.9: Network topologies [26]

easily according to it. The XY routing is the most commonly used routing algorithm in NoCs because of its simplicity and it is also proved to be deadlock and livelock free.

2.2.2.3 Flow control

Flow control is the mechanism that determines when the packet advances along its route. It also determines how the network resources, such as channels and buffers, are allocated to a packet. It is used mainly to ensure the correctness and the fairness of the network operations. Avoiding deadlocks, that may occur in complex routing algorithms, is the main issue in ensuring network correctness. Flow control can be divided to buffer-less and buffered flow control.

Circuit-switching is an example of buffer-less flow control. In circuit-switching, a route from source to destination is reserved before the transmission of the data. While in packet-switching network, a buffered flow control, the message is divided into packets that are routed independently through the network. Each packet contains its routing information. A packet is consisted of several flits that are transmitted in series. A flit is the minimum amount of data that can be sent across two nodes. Packet-switching is more commonly used in NoCs than circuit-switching.

Buffered flow control can also be divided into packet-buffer flow control and flit-buffer flow control. The former needs large buffers such as store-and-forward and virtual cut-through. In store-and-forward, the whole packet has to be stored in the router buffers before it is forwarded to the next node. This requires a large buffer in the router to store the entire packet. It has high latency as sending cannot start until the whole packet is received. Virtual cut-through also needs large buffers but it has a lower latency.

Wormhole and Virtual Channel (VC) are two examples of flit-buffer flow control. In the contrast of store-and-forward, flits in wormhole flow control is forwarded to the next node as soon as possible. No need to store the entire packet. So it need smaller buffers compared to store-and-forward. VC flow control is the splitting of the physical channel into several logically separate channels having independent buffers. It reduces the head-of-line blocking that happens when a not ready packet blocks the channel and prevents other ready packets to proceed. VCs can also be used to avoid deadlocks, improve performance and provide Quality-of-Service (QoS).

QoS has two basic types: best-effort services (BE) and guaranteed services (GS). BE NoC provides guarantees only for correctness of the network operation. While in GS NoC, performance guarantees are provided. A GS NoC is typically more complex than a BE NoC.

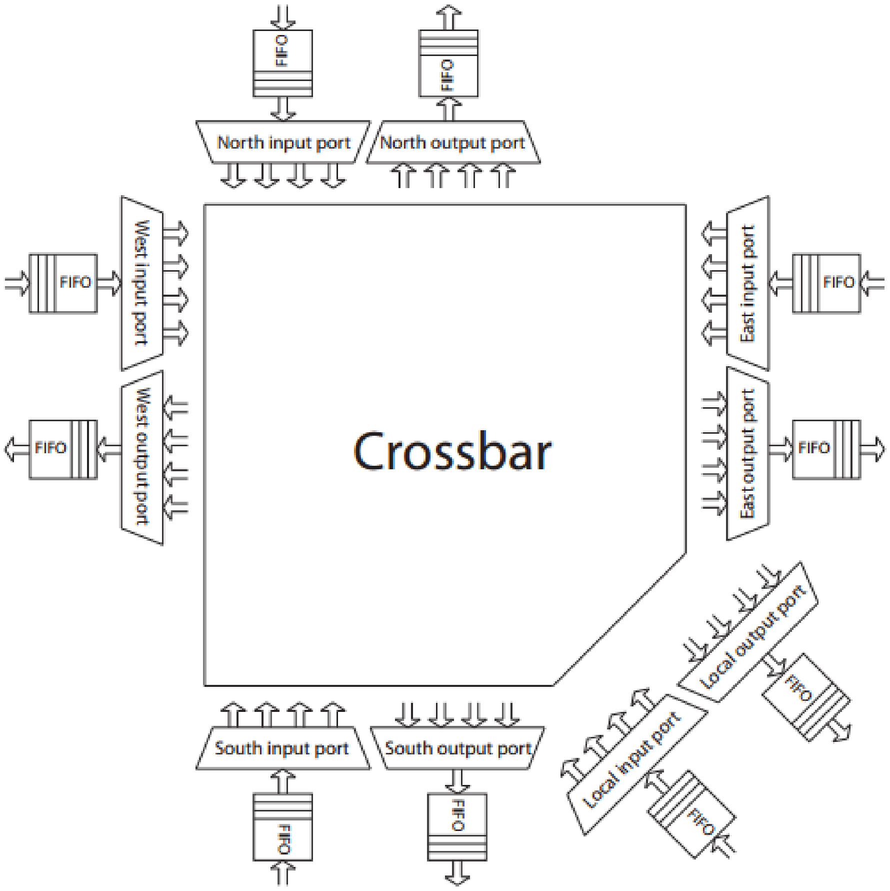


Figure 2.10: Router Architecture

2.2.3 Router Architecture

The router is the main block in NoC as it is responsible for routing packets between the PEs. NoCs performance and efficiency are highly dependent on the router architecture and implementation. The router architecture is also dependent on NoC topology, routing algorithms and flow control mechanisms. However, we can generalize the router design into three main components: the input port, the output port and the crossbar as shown in Fig. 2.10.

A typical NoC router has n input/output ports: (n-1) for connecting the router with the neighboring routers and one for connecting the router with the local PE through the NI. The crossbar is responsible for connecting the input ports with the output ports with some restrictions that each input port is connected to at most one output port, and vice versa. Moreover, a typical router carries out certain crucial tasks for a successful routing of the incoming packets such as routing computation, VC allocation, switch allocation and switch traversal.

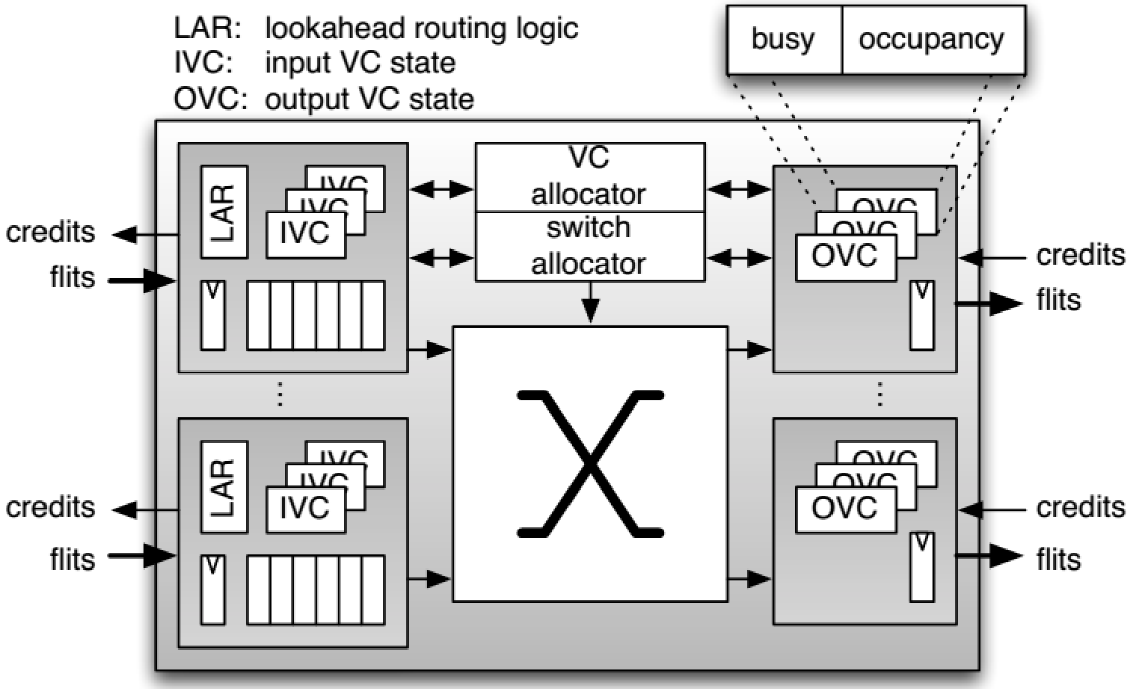


Figure 2.11: SOTA router architecture [27]

After the arrival of the first flit , i.e. head flit, of the packet, the router performs routing computation based on the routing algorithm and the packet destination to select the appropriate output port. This step is carried only on the head flit, and the remaining flits follow it. If a VC flow control is used, the router assigns available output VCs to the packets waiting in input ports. This step is called VC allocation and it is performed only on the head flit after the route computation. After an output port and an output VC, in a VC flow control, are assigned to a packet, it is now ready for switch allocation step. In this step, all ready flits request to traverse the crossbar. Then, granted flits successfully traverse the crossbar to the assigned output port and leave the router.

An example of a high quality generic NoC router is the state-of-the-art router (SOTA) developed in Stanford University [27]. To provide a way of evaluating the different configurations of the router architecture in terms of area, power, delay and network performance, an open source parameterized RTL implementation, described in industry-standard Verilog hardware description language, of SOTA is presented in [28]. The router architecture is shown in Fig. 2.11. It is a input-queued VC router that consists of input ports, output ports, crossbar, VC allocator, switch allocator and routing logic. The input ports contains FIFO buffers that hold the packets that can not be immediately forwarded. FIFO buffers are logically divided into multiple VCs. A look-ahead routing is used to reduce the pipeline stages into three stages only: VC allocation, switch allocation and switch traversal. A flexible buffer management schemes is used to improve buffer utilization. The implementation is very modular and all the blocks are highly configurable.

2.3 FPGA-based NoCs

The FPGA-based NoC can be implemented either on the FPGA configurable resources (soft NoC) or on Silicon as hard blocks (hard NoC). Many Soft and hard NoCs have been proposed in literature. In this section, we will discuss the pros and cons of the two types of implementation and show some examples of each type.

2.3.1 Soft NoCs

Soft NoC is implemented using the configurable resources such as LUTs and routing resources. This offers great flexibility in the NoC design. The topology and the NoC parameters can be reconfigured to match a certain application and its traffic patterns. However, this comes at the cost of the precious configurable resources that are basically used to implement the application. Soft routers are also larger and slower than those implemented on Silicon [6].

The literature shows a lot of work of mapping ASIC-oriented NoCs onto FPGAs, but due to the different structure of FPGAs, NoCs that are optimized for ASIC don't give the optimum performance when mapped to FPGAs. Consequently, some researchers have worked on designing FPGA tailored NoCs that take into consideration the hardware characteristics of FPGAs. PNoC [9], CONNECT [8] and Split-Merge [10] are three famous example of FPGA-oriented NoCs.

PNoC is a lightweight FPGA-friendly NoC that is used for FPGA-based applications. However, it is a circuit-switched NoC. So it can be used only with applications that tolerate the tear-down and the setup delay that characterizes circuit-switched networks. It supports partial dynamic reconfiguration. When a module is added or removed from the system, the local router is notified to update the routing table of the system. PNoC consists of a group of subnets. Each subnet contains multiple nodes connected to a single router that performs the circuit switching between the nodes. The router architecture is shown in Fig. 2.12. The table arbiter receives the connection requests and handle the access to the routing table. The port arbiter creates the desired connection when the destination is available.The switch box provides the actual connections between the modules.

CONfigurable NEtwork Creation Tool (CONNECT) is another FPGA-oriented NoC. It aims to achieve sufficient network performance while minimizing the use of FPGA resources. To target FPGAs, CONNECT router, shown in Fig. 2.13, utilizes a single pipeline stage instead of the typical three or four stages to reduce both the hardware cost and the latency. Moreover, CONNECT routers are tightly-coupled by using wider interfaces to maximize wire usage. It is implemented using Bluespec System Verilog (BSV) language. It supports various network topologies and various router parameters such as the number of input and output ports, the number of VCs, the flit width, the buffer depth, the flow control and the routing algorithm. It supports the traditional credit-based flow

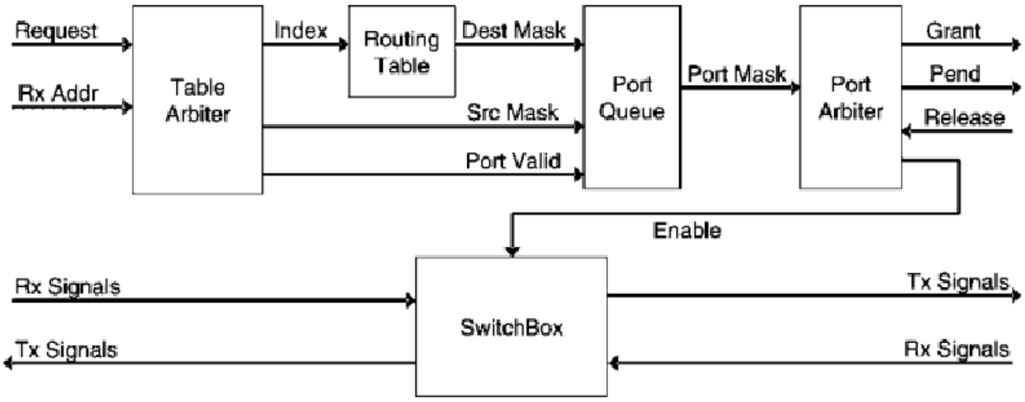


Figure 2.12: PNoC router architecture [9]

control, and the "peek" flow control. The routing algorithm is table-driven where the routing tables are implemented in Distributed RAMs. Flit buffers are organized per input and per virtual channel. Flit buffers are implemented using Distributed RAM and each VC FIFO is implemented as a circular buffer.

Huan proposes an FPGA optimized router which we refer to as Split-Merge. It is based on the split and merge primitives [10]. Split-Merge has different methodology than CONNECT in optimizing NoC for FPGA. Huan claims that FPGA NoCs should be pipelined heavily due to the low speed of configurable interconnects and high registers-to-logic ratio in FPGAs. So, unlike CONNECT, Split-Merge is pipelined to make use of FPGA abundant flip-flops. Moreover, as VCs tradeoff more complexity to achieve high wire utilization, Split-Merge does not support virtual channels as FPGA wires are abundant. The router is also implemented in Bluespec language. The router buffers are organized in both input and output ports as shown in Fig 2.14. This organization eliminates the need for a crossbar. The split module splits the incoming flits among the appropriate output buffers according to the packet destination. The merge then arbitrates between the flits from the multiple buffers to choose a wining flit to leave the output port. The simple backpressure flow control is used due to the absence of virtual channels.

2.3.2 Hard NoCs

Hard NoCs have a higher performance than soft NoCs because routers and links in hard NoCs are implemented directly on silicon, while soft NoCs are implemented on the programmable resources which have more delay [6, 22]. Hard NoCs are also more area-efficient [6] and power-efficient [17]. This is the same motivation behind embedding dedicated blocks such as memories, DSP blocks and hard processors into the FPGA. Moreover, hard NoCs do not consume the valuable configurable resources of the FPGA like soft NoCs. They also enable partial dynamic reconfiguration and parallel compilation with less effort.

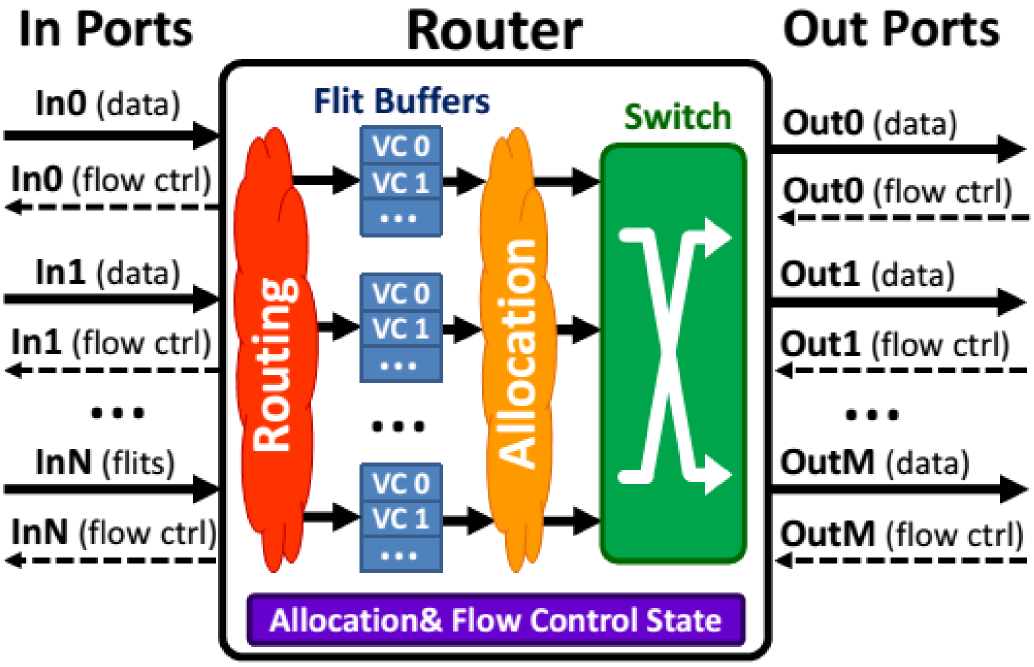


Figure 2.13: CONNECT router architecture [8]

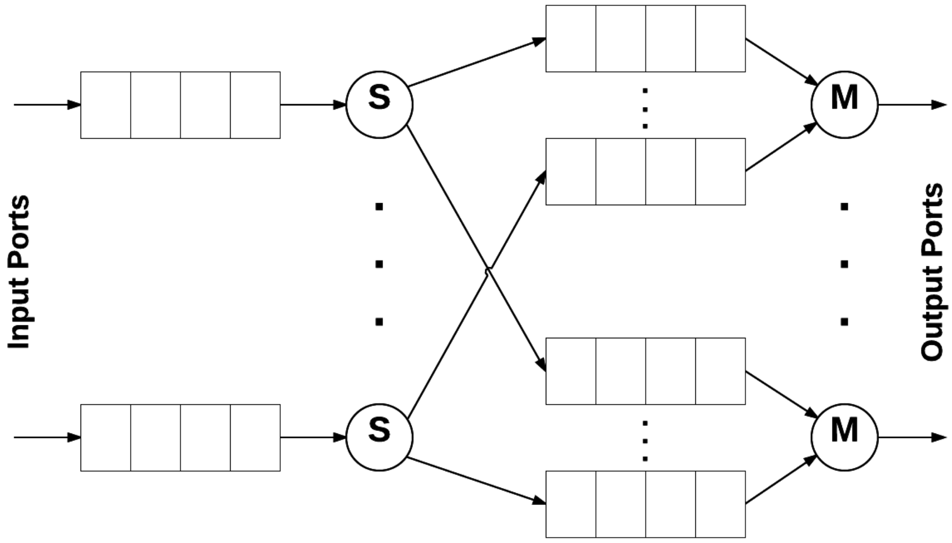


Figure 2.14: Split-Merge router architecture [10]

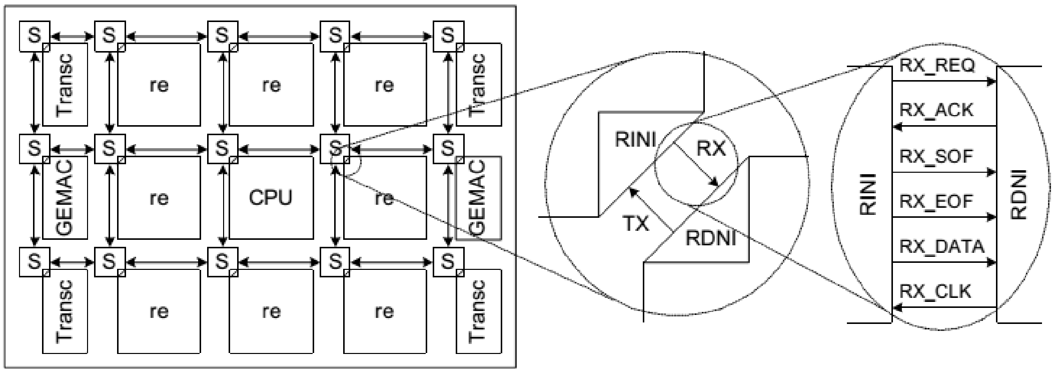


Figure 2.15: Hecht's proposed architecture of future FPGAs [7]

A hard NoC is almost completely disjoint from the FPGA fabric, only connecting through router-to-fabric ports. This makes it easy to use a separate power grid for the NoC with a lower voltage than the nominal FPGA voltage. This is desirable because we can trade excess NoC speed for power efficiency.

The idea of embedding the FPGA with hardwired Network-on-Chip as a high-level routing resource is showed up for the first time in [7]. The motivation behind this idea is supporting the dynamic reconfiguration without consuming the valuable reconfigurable FPGA resources in implementing a soft NoC. But due to the another benefits of NoC in FPGA, the author concludes that supporting dynamic reconfiguration should be seen as an additional advantage for NoC not the only motivation for it. The author argues that most of the FPGA designs nowadays resembles SoC which needs SoC buses and FIFOs so NoC will be a perfect alternative at a much lower cost. His proposed architecture of future FPGAs is shown in Fig. 2.15. The architecture consists of tiles that communicate only through the NoC. This architecture is modeled in systemC and no hardware implementation is given.

In [11], Goossens proposes that NoCs should be hardwired in FPGAs due to their benefits over Soft NoCs. According to him, hardwired NoCs have lower implementation cost (smaller area, higher speed and less power) and it is easier to achieve timing closure and to exploit partial dynamic reconfiguration using hardwired NoCs. He also proposed that the hardwired NoC will be used for both the functional and the configuration interconnects. In this architecture, the FPGA is partitioned into CFRs (configuration and functional regions) as shown in Fig. 2.16. Unlike [7], the neighboring CFRs can communicate directly without using the NoC. All the CFRs are connected to the unified configuration and functional NoC through the Network Interface (NI). The NI is divided into two parts: the kernel and the shell. The NI kernel is hardwired and it is responsible for network layer functions such as routing, while the NI shell is soft implemented as it is responsible for transport layer functions which depend on the application. Goossens makes use of Ethereal NoC [29] to show the advantages of hardwired NoCs over soft NoCs. A 32-bit NoC, consisting of one Ethereal router and five kernel/shell NIs, is implemented on Virtex-4 FPGA (soft) and on

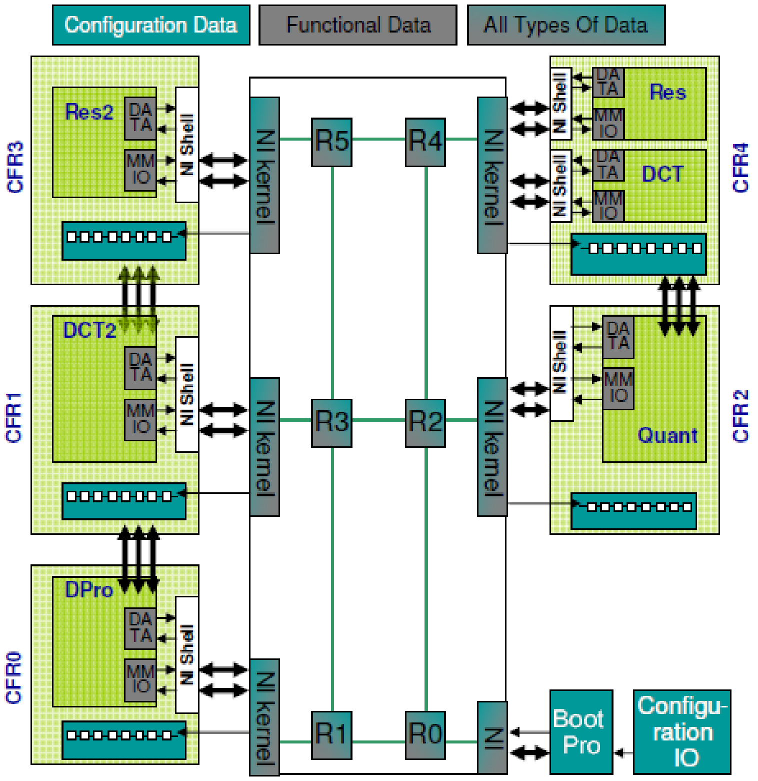


Figure 2.16: Goossens's proposed FPGA architecture with unified hardwired NoC [11]

130nm CMOS technology (hard). The area overhead of the soft NoC is 830% for a CFR size of 1000 LUTs, while the overhead of the hard NoC is only 14% for the same CFR size. The hard NoC operates at 500 MHz while the soft NoC operates at around 120 MHz. This shows 149x improvement in link bandwidth per area of the hardwired NoCs over the soft NoCs.

In [12], Francis suggests using a circuit-switched Network-on-Chip with hard routers and reconfigurable links. The circuit-switched network is used, despite its poor performance, because it needs small input buffers. Francis proposes that the hard routers should replace LUTs within the cluster and have the same access to the configurable wiring as LUTs, as shown in Fig. 2.17, inorder to maintain the homogeneity of the FPGA. As routers implemented on silicon run much faster than soft IP cores, the hard routers are limited by the input data rate from the cores. Therefore Francis suggests using TDM wiring in FPGAs to connect the fast hard blocks with the soft logic. This allows time-division multiplexing of the hard blocks and increases the efficiency of the used silicon area. Muxes and other control circuits need to be implemented in silicon alongside the conventional FPGA wiring to allow TDM. The author then compares the hard NoC, with and without TDM wiring, with a soft-implemented one. The soft NoC, implemented on Virtex-4 FPGA, consumes 4x power than hard NoC, implemented on 90nm CMOS technology. Also, the soft NoC has 12.5x larger area than the hard one.

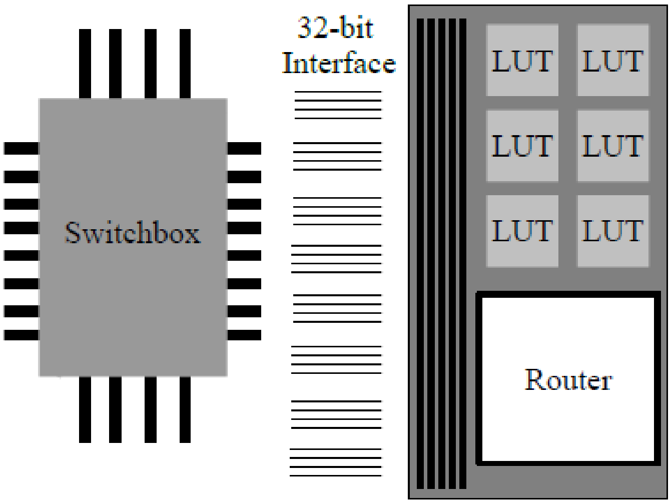


Figure 2.17: Francis's proposed placement of hard routers replacing LUTs [12]

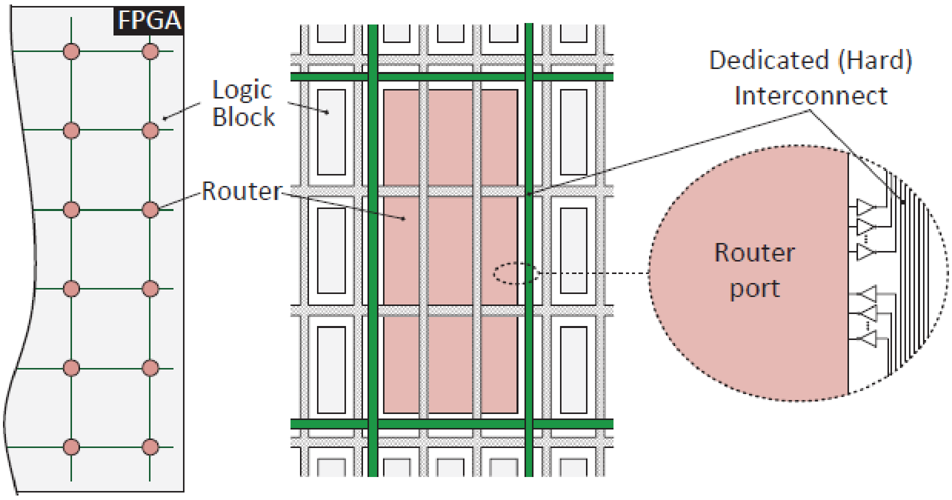


Figure 2.18: Abdelfattah's proposed hard NoC floor plan [18]

In [6, 15, 18], Abdelfattah and Betz at university of Toronto propose augmenting FPGAs with embedded NoCs. They preserve the conventional architecture of FPGAs and suggest that embedded NoCs should be seen as adding highways to the existing roads (conventional routing resources). They use SOTA router from Stanford University to build their NoC. They implement three versions of this NoC: soft (soft routers and soft links), hard (hard routers and hard links) and mixed (hard routers and soft links). The comparison between these three NoCs shows the advantages of hard NoCs. At 65 nm technology, the hard NoC is 23x smaller, 6x faster and 11x less-power than the soft NoC. To embed the NoC inside the FPGA fabric, they propose that the hard router replaces 9 FPGA logic clusters (equivalent area of the router) as shown in Fig. 2.18. Routers are connected together through dedicated hard links, and they interface to the FPGA fabric through programmable multiplexers connected to the soft interconnect.

Chapter 3

Selecting NoC Parameters

As we propose the FPGA-based NoC to be a hard NoC, i.e. implemented on silicon, due to the aforementioned advantages of hard NoCs over soft ones, this implies that the designed NoC will not be configurable. Consequently, we need to study the various NoC parameters and their effect on NoC performance to find the best-fit parameters that will be used in designing our hard FPGA-based NoC. This work is published and presented in IEEE ISCAS 2015 conference [30].

3.1 Performance Measures

The NoC design is evaluated by many performance measures. However, we will classify these measures into two main groups, the network group and the chip group. The former contains performance measures dealing with the network performance which is independent on the platform on which the NoC is implemented, while the latter is concerned with the hardware implementation of the NoC.

3.1.1 Network Performance Measures

There are various ways to measure the network performance. We will focus on the two most important measures: the network throughput and the network latency.

3.1.1.1 Network Throughput

The network throughput represents the maximum traffic accepted by the network [31]. It is measured by counting the number of messages, i.e. flits (the smallest message unit), successfully arrived at their destinations per time unit. Using clock cycles, instead of seconds, as the time unit makes the network throughput independent on gates and wiring delays.

The network throughput is normalized to the size of the network by dividing by the number of nodes. Accordingly, It is measured in flits/cycle/node. A network throughput of value "one" means that every node is receiving one flit every cycle, i.e. the maximum theoretical capacity of the network. Thus, the network throughput is a fraction of the network capacity [26].

3.1.1.2 Network Latency

The network latency is the time taken by the packet to traverse the network and arrives at its destination. Hence, it is the time elapsed between the injection of a head flit into the network and the arrival of the tail flit at the destination [1]. Each packet faces different latency according to the source/destination pair and other overheads. Thus, this time is calculated for all the packets and the average network latency is calculated. It is measured in terms of clock cycles. It also has different values at different traffic. The zero-load-latency is the network latency at almost zero traffic which ignores the contention of the packet with other packets at shared resources.

3.1.2 Chip Performance Measures

These measures depends on the hardware realization of the NoC and the targeted platform whether an FPGA or ASIC. They also affect the overall performance of the NoC.

3.1.2.1 Maximum Operating Frequency

It is the maximum frequency at which the NoC can operate. It is an important measure when selecting design parameters. Some parameters may give a high network performance at the cost of design complexity. For example, a NoC may have a low latency in terms of cycles, but it also has a low maximum operating frequency, i.e. long cycle time, resulting in a high actual latency measured in seconds.

3.1.2.2 Area

The design area is a very important metric that should be considered when choosing the design parameters of a NoC. It represents the silicon area occupied by the design after hardware implementation. The area is consumed by both the router logic and the wiring layout. There is also a trade-off between network performance and area. For example, a large number of nodes, buffers or VCs gives high network performance but occupies larger area.

3.2 Simulation Setup

To study the effects of the various NoC Design parameters on NoC performance, two popular NoCs are used as case studies. The selection of the used NoCs is based on choosing two completely different NoCs: an ASIC-oriented NoC and an FPGA-oriented NoC.

This will help us to find the best-fit parameters for our designed FPGA-based hard NoC. The FPGA-based hard NoC is implemented using ASIC process but within an FPGA environment. So it is important to study both ASIC-oriented and FPGA-oriented NoCs.

Moreover, The selected NoCs should be highly flexible to enable the variation of the different NoC design parameters. They should also support different topologies. Thus, SOTA [27] and CONNECT [8] have been selected for the simulation. SOTA is a highly parameterized open source RTL implementation of a generic VC router. It is an ASIC-oriented NoC. The router is pipelined in three stages: VC allocation, switch allocation and switch traversal, and it is implemented using Verilog hardware description language. While CONNECT is an FPGA-oriented NoC. It has an online generator tool that produces synthesizable RTL implementation of the NoC in Verilog hardware description language [32]. The tool supports various topologies and a variety of network and router design parameters. Unlike SOTA, CONNECT is a single stage router, i.e. not pipelined.

For results' homogeneity, the same simulation setup is used for both SOTA NoC and CONNECT NoC with the same packet source configuration. The network performance including the network throughput and the network latency is measured by cycle-accurate RTL simulations using MODELSIM 10.3c for the two NoCs at different NoC parameters. The two NoCs are implemented on the same platform to obtain the implementation results such as the maximum operating frequency and the area for both NoCs at different configurations. The chosen platform is Xilinx Virtex-6 LX760 FPGA (part xc6vlx760, speed grade -2). The implementation results are obtained by Xilinix ISE 14.6 tool.

We use the standard setup for evaluating interconnection networks [26] as our simulation setup. A packet source is attached to the input of each local port in the NoC. The packet source generates and injects packets into the network according to a specific traffic pattern and a specific packet length. Each generated packet is tagged with its departure time and the number of injected packets is counted. At the output of each local port, a measurement process counts the number of arrived packets and records the arrival time of each packet.

Moreover, a sufficiently large queue is used between the packet source and the network to isolate the traffic generation process from the network. Without the queue, if the NoC buffers are full, the packet source will fail in injecting packets into the network as it can not accept more traffic. In this case, the generated traffic is affected by the network. Hence, it is not the original specified traffic pattern. Thus, this configuration, which is called open-loop measurement, enables the controlling of the traffic independently on the network status. This is crucial in evaluating the performance of the network under a specific traffic pattern [26].

The traffic pattern is a function of the destination address of the generated packet. In a NoC, each router is assigned an address depending on its location in the network and the used topology. For example, in a mesh or a torus network, the router address is its Cartesian coordinate, i.e. x-y, considering the origin is at the bottom left router, i.e. x-y equals 0-0. To test the network performance, synthetic traffic patterns are used. They are simple test patterns that resemble real applications' traffic patterns.

Uniform distribution traffic is the most commonly used synthetic traffic pattern, where each packet source generates packets to every destination except its attached router with equal probability. There are another types of synthetic traffic patterns such as Neighbor, Tornado and Bit Complement [26] in which each source sends packets to only one destination. The used packet source in this simulation setup generates packets according to Uniform distribution traffic pattern. The rate of generation of packets, i.e. offered traffic, is controlled by a parameter called packet injection rate to test the network under different loads.

The network performance is usually measured in the steady-state phase of the network [26]. The transient performance may be of some interest when you are interested in the transient response of the network to a sudden change in parameters or traffic. However, as we are interested in measuring the performance of the network with different design parameters, only steady-state measurements are taken into account.

To consider only steady-state performance, the simulation has to be done in three stages: warming-up, measuring and cooling-down. First, the simulation should run for enough number of cycles, i.e. warming-up cycles, for the network to reach equilibrium before any measurements. After warming-up, the simulation reaches the measuring stage that lasts for a number of cycles called measuring cycles. Packets generated in measuring stage, i.e. measurement packets, are the only considered packets for measuring the network performance. The number of measuring cycles should be large for more reliability of the results. In cooling-down stage, the simulation continues until the arrival of all measurement packets at their destination. If the network is suffering from starvation, the cooling-down stage may never end.

The network throughput is calculated by counting the number of successfully arrived packets (flits) during the measuring phase, dividing it by the number of measuring cycles, and then normalizing it over the number of nodes in the NoC. The network latency is computed by averaging the difference between the arrival time and the departure time of each measurement packet, i.e. packets generated during the measuring phase. This network latency includes also the time spent in the queue located after the packet source.

3.3 Results and Design Recommendations

NoC design space is very broad as there are many design parameters that can be varied. However, we will focus on certain design parameters that have direct effects on the NoC performance. The NoC performance will be measured, according to both the network and chip performance measures mentioned before, at different values of these design parameters using the discussed simulation setup. For each design parameter, we design a figure of merit to help in selecting the optimum value of this design parameter and show the simulation results of both SOTA and CONNECT NoCs.

Based on the simulation results, we propose some design recommendations for selecting the best-fit NoC design parameters. The simulation results and the design recommendations are shown here as we will base our NoC design in the next chapters on them.

3.3.1 Topology

The first design parameter is the network topology. As we have mentioned in chapter 2, selecting the topology is the first step in designing the NoC. It has a direct effect on almost all the performance measures. Moreover, the choice of topology is a trade-off between various performance measures. For example, a complex topology with many links has a low latency due to the small average routing distance between the nodes. However, it is hard to layout and it consumes much area due to the wiring complexity.

Only three topologies are selected for the topology comparison: Mesh, Torus and Flattened Butterfly (FBfly) [33] as they are known for their promising performance. Other topologies such as Ring and Star don't form a high performance NoC. Some topologies provide good network latency but with low maximum operating frequency or low network throughput. For example, A Torus network has a lower latency than a Mesh network because it has a smaller network diameter, i.e. the maximum distance between any two nodes, due to the connection between the end nodes in each dimension. However, a Torus NoC has a higher wiring complexity than a Mesh NoC which may give a lower maximum operating frequency.

Accordingly, the metric used in selecting the optimum topology should take into account these different aspects. It should consider the network latency, the network throughput and also the maximum operating frequency. Therefore, load-latency curves are chosen as our metric in deciding the best topology. A load-latency curve is a curve that shows the network latency at different loads. It also shows the network throughput which equals the load after which the network saturates. The latency is measured in cycles and the load is measured in flits/cycle/node.

The load-latency curves for both SOTA and CONNECT NoCs are shown in Fig. 3.1 and 3.2. Both NoCs have 16 nodes as it is the optimum number of nodes as we will show later. From the curves, a FBfly topology has the lowest network latency. This is because it takes a maximum of three hops to deliver a flit in 16-node network [33], i.e. 2-ary 4-flat network. However, it has lower network throughput than the other two topologies. As expected, Torus has lower network latency and higher network throughput than Mesh.

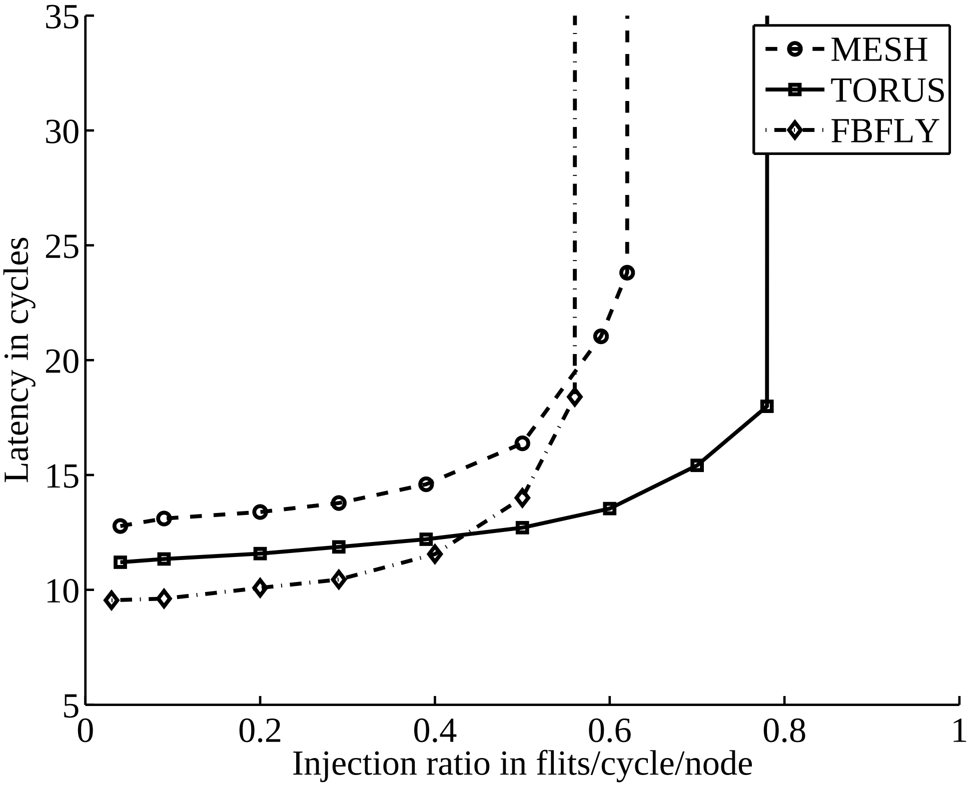


Figure 3.1: SOTA Load-latency curves at different topologies

However, these curves do not consider the effect of long links of torus on the maximum operating frequency and hence, on the overall NoC performance.

In order to take into account the effect of maximum operating frequency on the NoC performance, the load-latency curve is scaled with the maximum operating frequency, i.e. load is measured in flits/s/node and latency is measured in seconds. To be able to measure the accurate operating frequency, Xilinx Floor-planning tool is used to distribute the NoC routers equally within the FPGA in order to get the most accurate maximum operating frequency. In router distribution, we consider two types of Torus topology: the regular Torus and the folded Torus. In regular Torus, routers at the edges of the network are connected together through long end-around links which span the whole x-dimension/y-dimension of the FPGA. However, in folded Torus, routers are distributed using a folded layout, in which long end-around links are eliminated at the expense of doubling the length of all other links.

The load-latency curves, after considering maximum operating frequency, for both SOTA and CONNECT NoCs are shown in Fig. 3.3 and 3.4. From the shown curves, for applications that don't require a high throughput, a FBfly topology may be a good solution as it has the lowest latency. However, for higher loads, Mesh and Torus topologies give higher throughput with a lower latency than FBfly. Although a Torus topology provides better network throughput and network latency than a Mesh topology as shown in Fig.

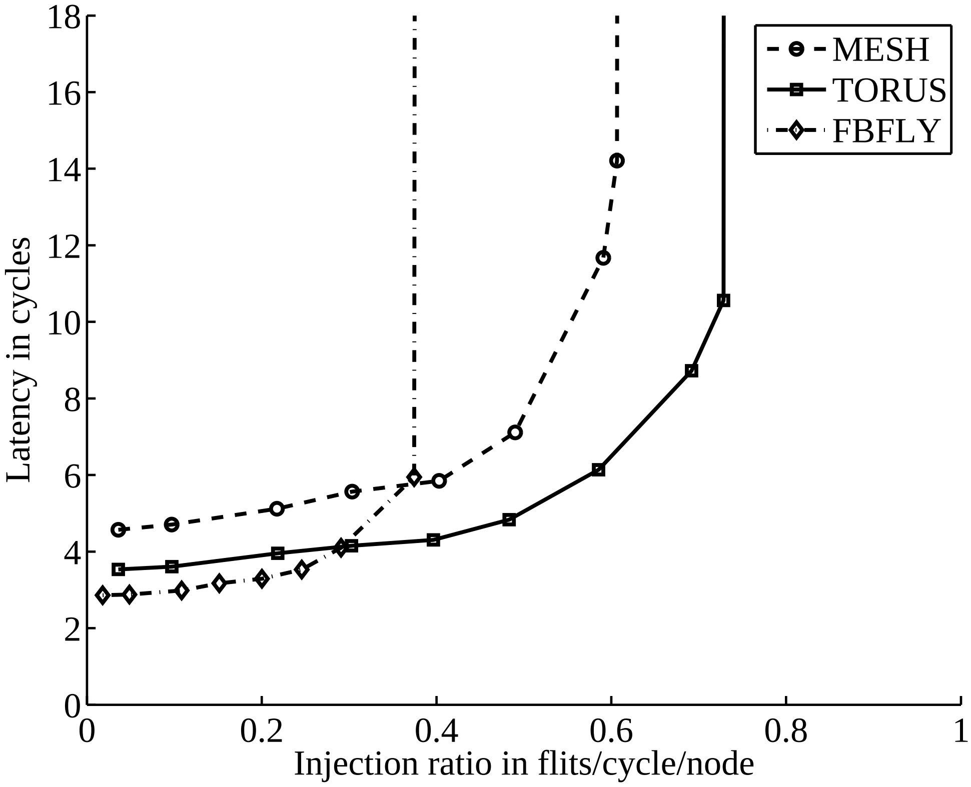


Figure 3.2: CONNECT Load-latency curves at different topologies

3.1 and 3.2, the maximum operating frequency of the regular Torus is significantly lower due to the end-around links. So, the overall NoC performance of a Mesh is better than a regular Torus.

When using the folded layout, a Mesh is still better than a folded Torus in CONNECT NoC because CONNECT router is a single stage router so the delay of double-length links of folded Torus is added to the logic delay of the router reducing the maximum operating frequency of the NoC. However, in SOTA NoC, the double-length links are no longer the limiting factor to the operating frequency; SOTA is fully pipelined router which means that the delay of the links is not added to the router delay, and the logic delay of the router is greater than the delay of the double-length links. Thus, the maximum operating frequency of a folded Torus in SOTA NoC is nearly the same as a Mesh resulting in a better overall NoC performance than a Mesh.

Although the simulation results of SOTA NoC show that a folded Torus has a better NoC performance than a Mesh, a Mesh NoC is still way better than a Torus NoC even with a folded layout for several reasons. First, in hard NoCs, the hard routers, i.e. implemented on silicon, can operate at frequencies near 1 GHz in 65 nm technology [18], but the long links of Torus (regular or folded) cannot survive at these frequencies. For example, a hard link can traverse about one-third of a 65 nm FPGA at frequencies near 1 GHz [18], but in a 16-node folded Torus NoC, each NoC link needs to traverse half the FPGA. Thus,

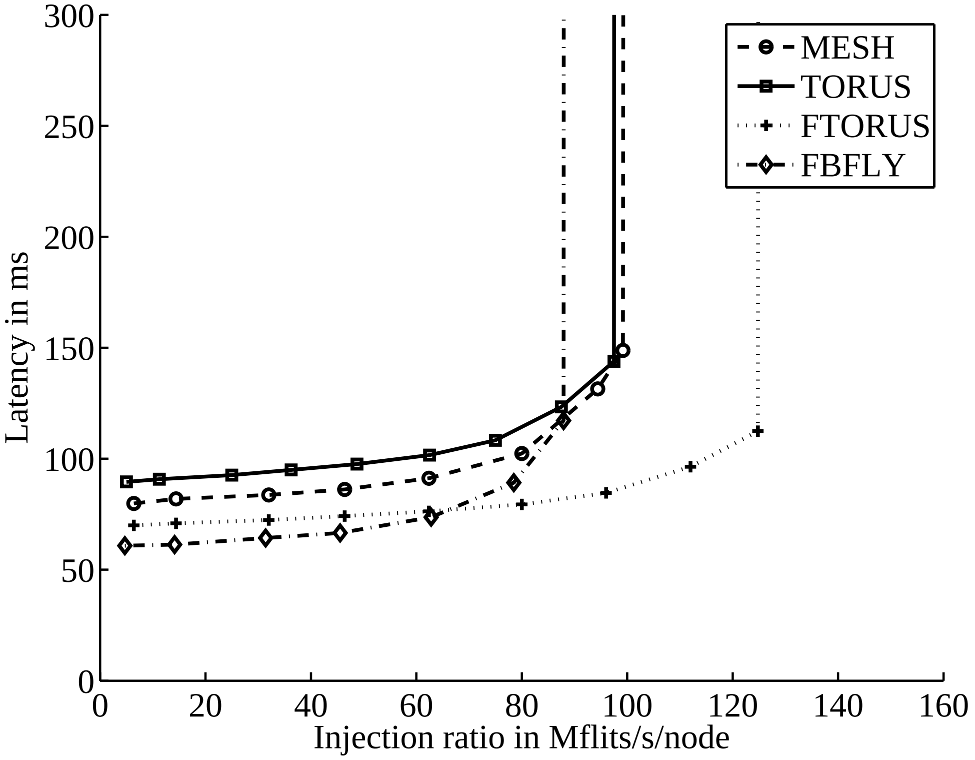


Figure 3.3: SOTA Load-latency curves at different topologies

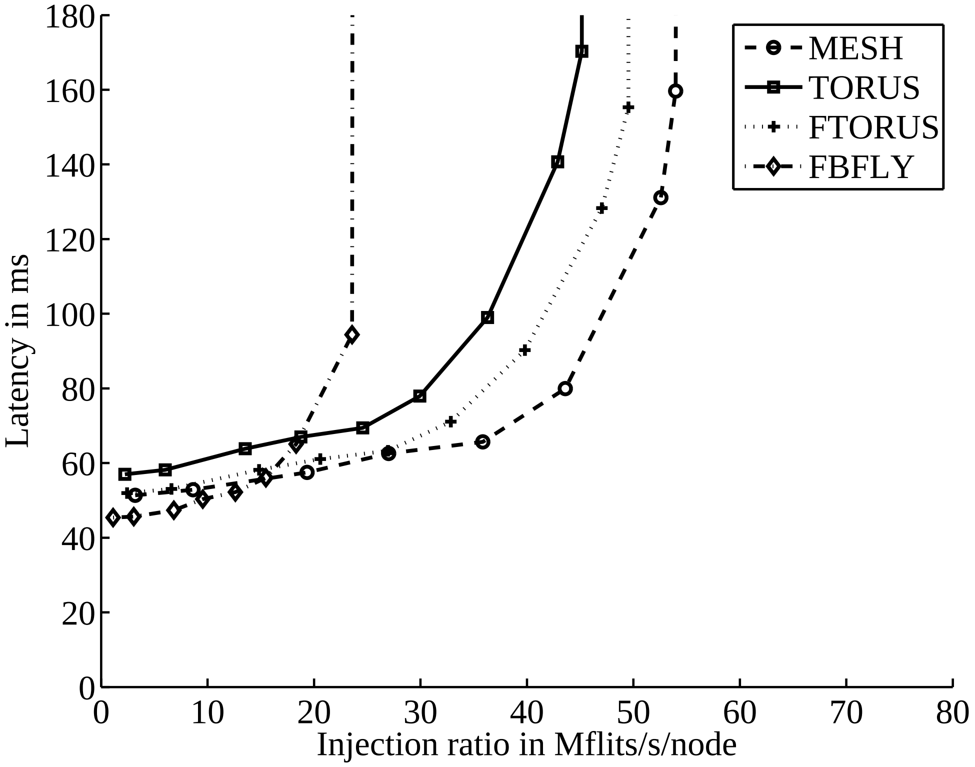


Figure 3.4: CONNECT Load-latency curves at different topologies

folded torus links will limit the maximum operating frequency of the NoC (It is clear that the situation is worse in a regular torus). Secondly, dimension-order routing in Mesh networks is deadlock free, however, it is not in Torus networks (It requires at least two virtual channels to be deadlock free) [26].

As we target a high performance NoC, a Mesh would be the optimum topology among all the three examined topologies. Thus, all further simulations are based on a Mesh topology.

3.3.2 Number of nodes

After deciding the network topology, the next step is to decide the network size, in other words, the number of nodes in the network that are connected by the chosen topology. In FPGA-based NoC, the extreme case is to consider each basic logic element as a node. This will obviously lead to a NoC larger than the FPGA itself. The other obvious extreme case is to consider the whole FPGA as one node which will not benefit from the advantages of NoCs within the FPGA.

From the load-latency curves shown in Fig. 3.5 and 3.6, increasing the number of nodes decreases the network throughput. The results conform with the theoretical capacity of a Mesh network which is inversely proportionally with the radix of the network [26]. The network radix can be defined as the square-root of the number of nodes in a squared Mesh network. However, from the definition of the network throughput mentioned before, the network throughput is normalized to the number of nodes, in other words, it shows the maximum traffic that can be generated from each node. A node can be capable of generating only small traffic but the total traffic from the network may be large due to the large number of nodes.

Thus, it is not fair to decide the optimum number of nodes according to the network throughput. The total network throughput should be considered which shows the maximum traffic that the whole network is capable of carrying. It can be computed by multiplying the network throughput, measured in flits/cycle/node, by the number of nodes of the network. The network latency should also be taken into account. By increasing the network size, the diameter of the network will increase; more hops are needed to reach the destination. Thus, the latency noticeably increases when adding more nodes. Consequently, Selecting the optimum number of nodes is a trade-off between the total network throughput and the network latency. Hence, the figure of merit (FOM) is given by equation 3.1.

$$FOM = \frac{Network\,Throughput \times Number\,of\,nodes}{Zero\,load\,Network\,latency} \tag{3.1}$$

Although the area of the network increases when adding more nodes, it is not taken in consideration in the FOM. That is because the area of the individual router does not change

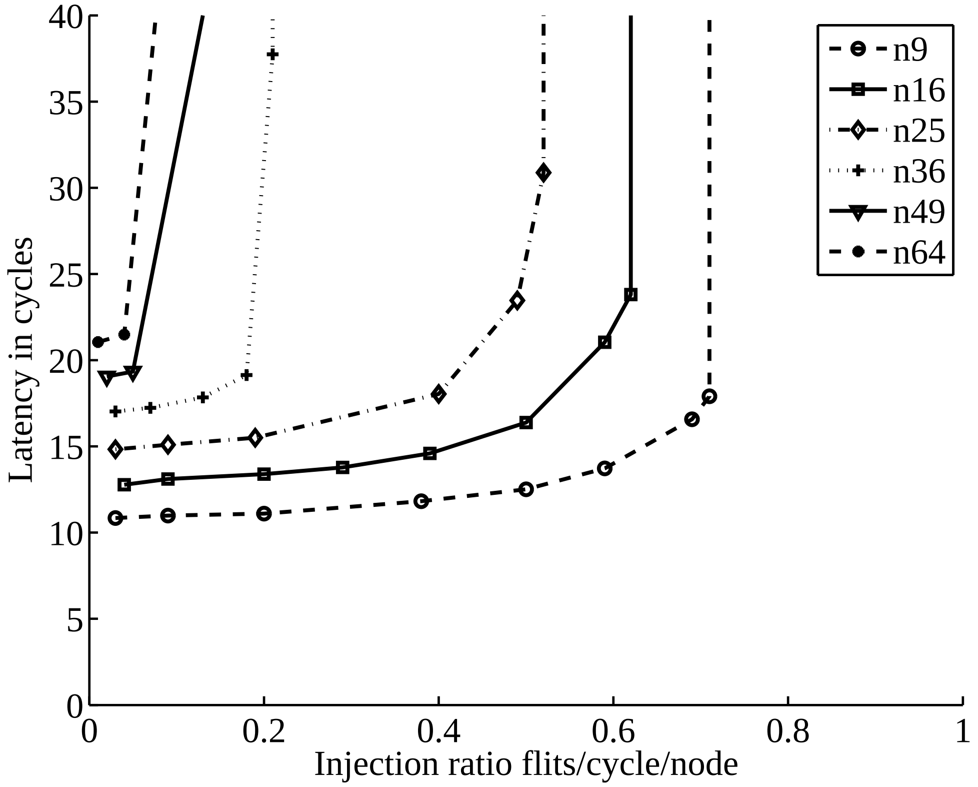


Figure 3.5: SOTA Load-latency curves at different number of nodes

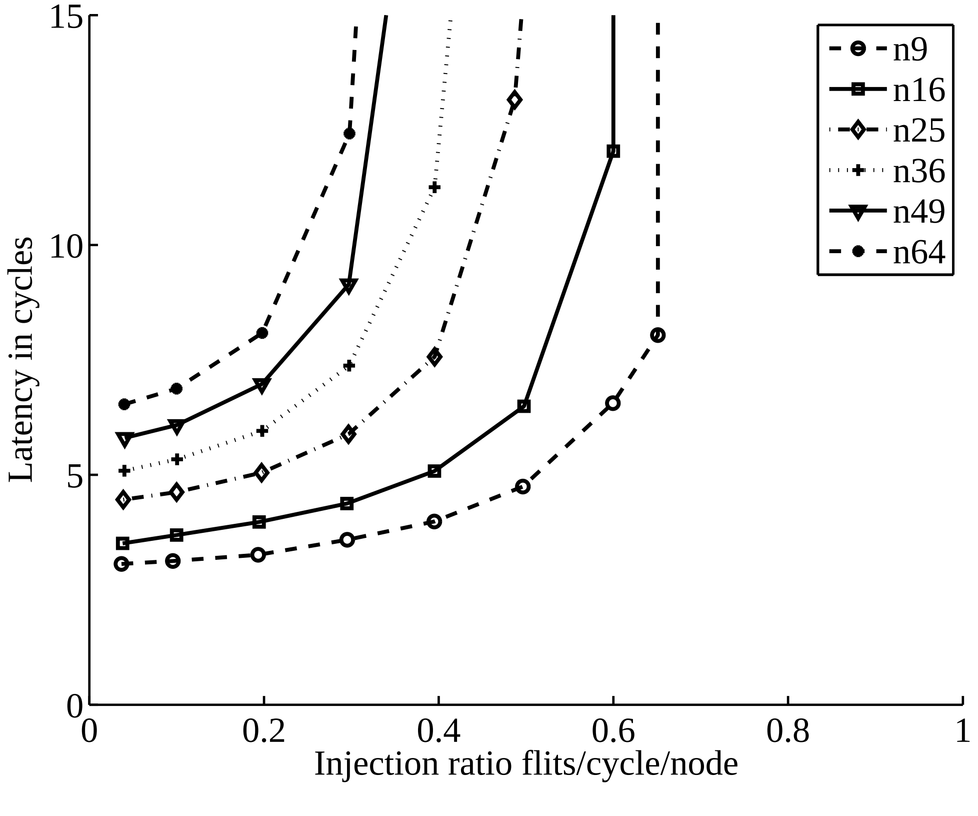


Figure 3.6: CONNECT Load-latency curves at different number of nodes

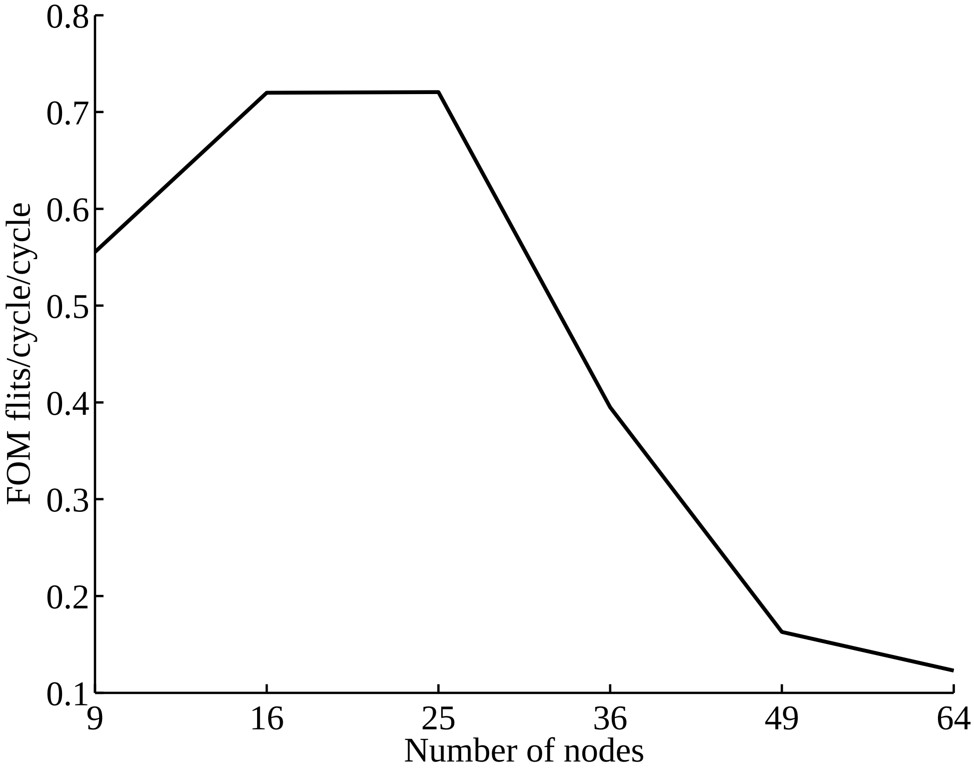


Figure 3.7: SOTA FOM for number of nodes

when adding more nodes. As long as the targeted platform is able to accommodate the whole network and there is enough area for the other computing cores, one should follow the optimum number of nodes given by this FOM to get the optimum performance. In other words, the total area of the network should only be seen as a hardware limit.

The computed FOM for both SOTA and CONNECT NoCs at various number of nodes is shown in Fig. 3.7 and 3.8. From the curve, a 16-node network gives the highest FOM in CONNECT NoC. While for SOTA NoC, 16 nodes and 25 nodes give almost the same performance, but a 25-node NoC has nearly double the total area of 16-node NoC. Hence, the optimum number of nodes is 16 nodes. All other simulations is based on a 16-node network.

3.3.3 Virtual channels

The number of virtual channels (VCs) is a crucial design parameter as it has a great effect on the NoC performance. VCs are used to increase network performance, increase wire utilization, solve deadlock problems, and reduce the head-of-line blocking. However, implementing VCs increases the complexity of the router as it requires complex blocks such as VC allocator. Also, VCs consume more area for buffers and controlling logic. Consequently, the number of VCs is an obvious trade-off between network performance and chip performance.

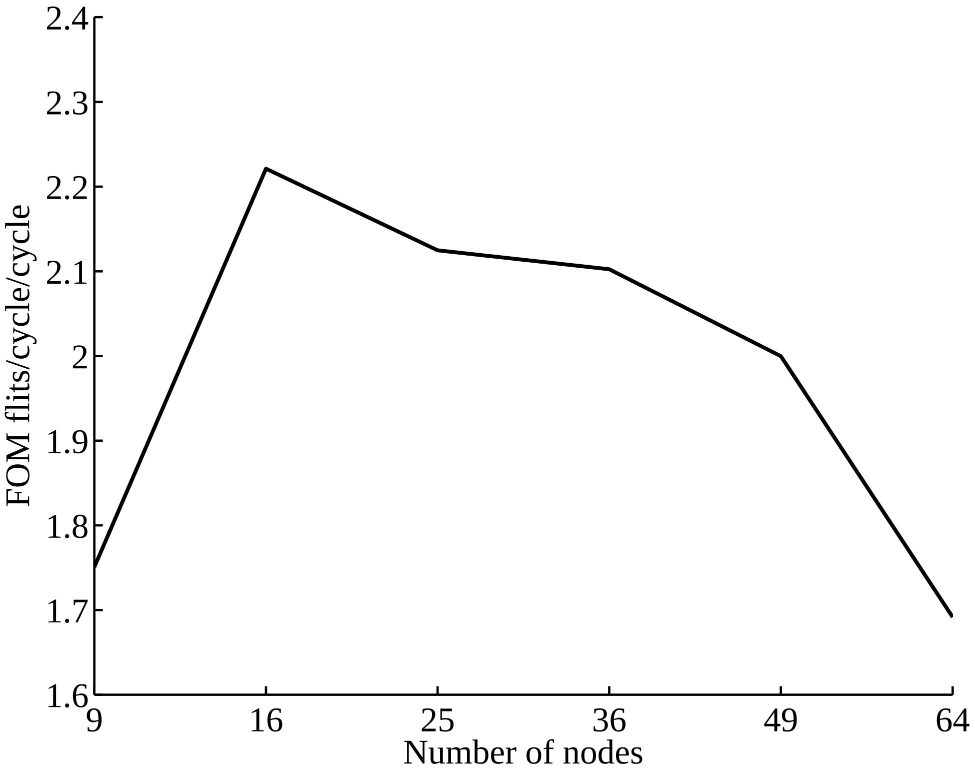


Figure 3.8: CONNECT FOM for number of nodes

This is asserted by our simulation results of different number of VCs. The load-latency curves for both used NoCs, shown in Fig. 3.9 and 3.10, show that the network performance such as the network throughput increases when adding more VCs. The curves also show that the network performance of four or more VCs is nearly the same in a 16-node Mesh NoC. However, increasing the number of VCs adds more complexity to the design that decreases the maximum operating frequency and increases the consumed area, in other words, adversely affects the chip performance. This can be proved by scaling the load-latency curves with the maximum operating frequency of each design. Fig. 3.11 and 3.12 show the load-latency curves after considering the maximum operating frequency.

From the results, in SOTA, a single VC is recommended for an application that requires a low traffic, less than 60 Mflits/s/node, as it has the lowest latency, the highest maximum operating frequency and also the smallest area. However for applications that require a high traffic, a 2-VC NoC is the optimum choice in SOTA as it gives the highest throughput, measured in flits/s/node. This is because that it is a compromise between a high maximum operating frequency and a high network performance. In CONNECT, 1-VC and 2-VC NoCs have nearly the same performance. This is because that CONNECT uses a different router architecture when handling a single VC. However, a 2-VC NoC still has a slightly higher throughput.

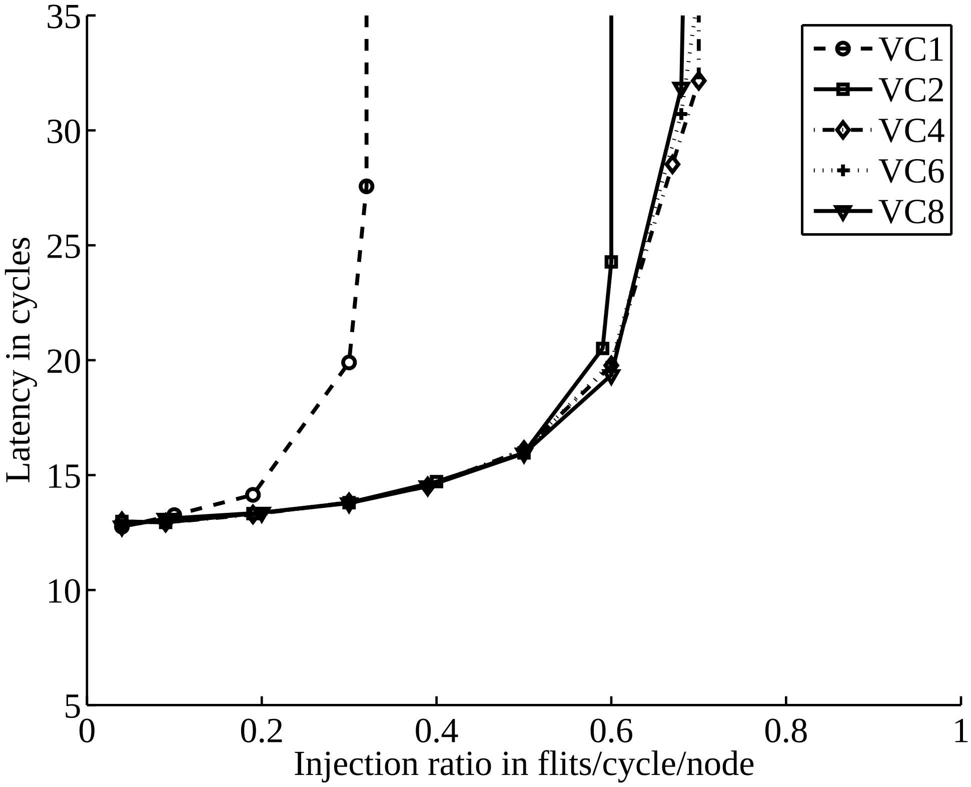


Figure 3.9: SOTA Load-latency curves at different number of VCs

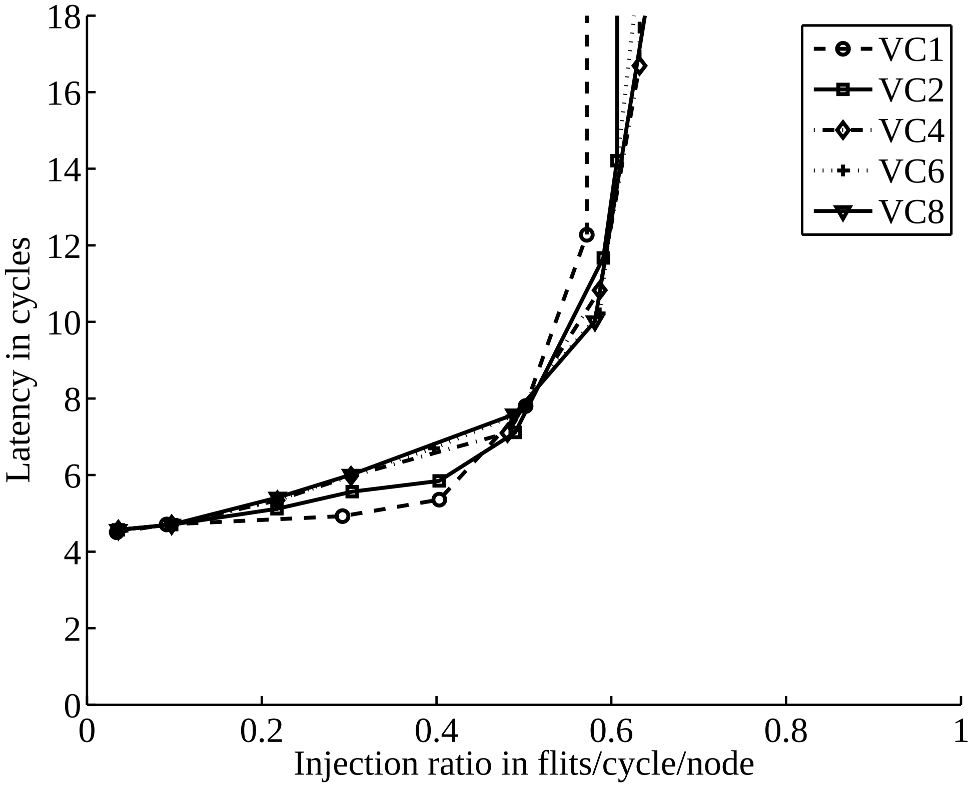


Figure 3.10: CONNECT Load-latency curves at different number of VCs

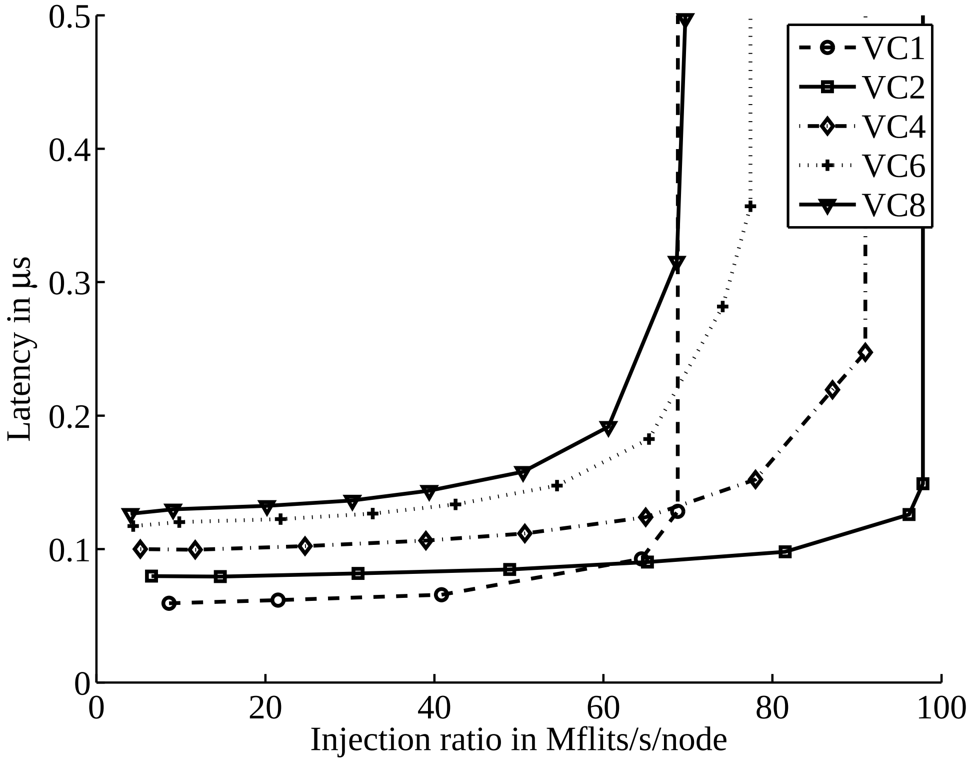


Figure 3.11: SOTA Load-latency curves at different number of VCs

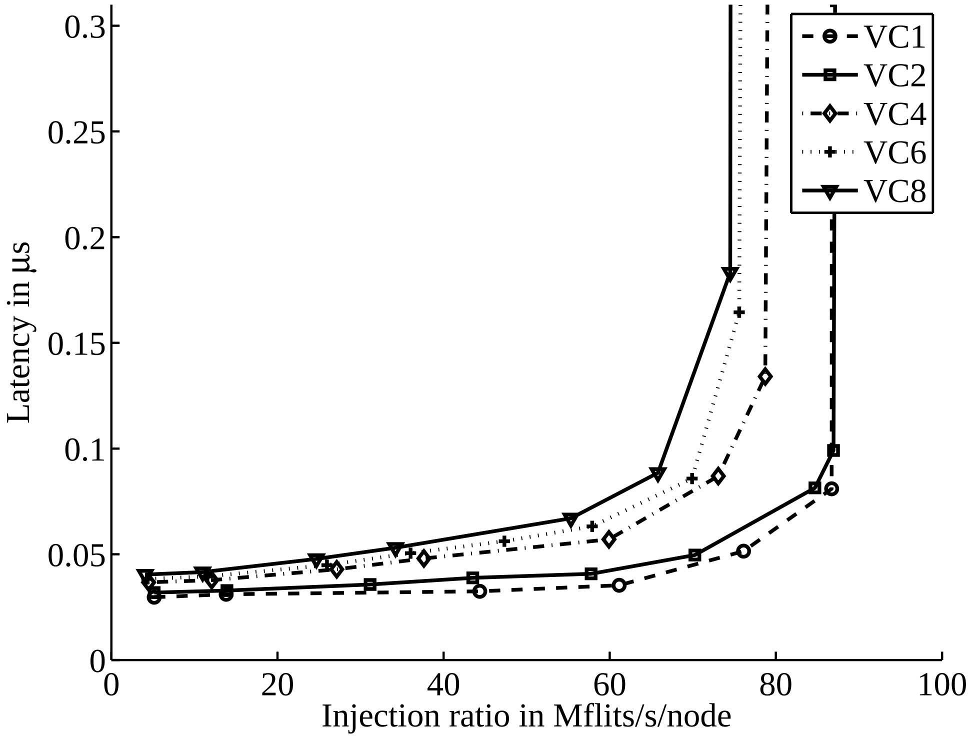


Figure 3.12: CONNECT Load-latency curves at different number of VCs

Consequently, the optimum number of VCs is two that gives the best performance. Hence, a 2-VC NoC is used in the other simulations. Moreover, it is shown from the simulation results that using more than two VCs is useless in a 16-node NoC. It can only be used in NoCs of larger network size.

3.3.4 Buffer depth

Buffers are one of the main components of NoCs. A router in a NoC must assure that the next router has a space in its buffers before forwarding a flit to it as NoCs are lossless networks. The buffer depth and its organization are crucial to achieve high network performance. However, buffers consume a large fraction of the NoC routers' area and power [27]. Hence, the buffer depth is also a trade-off between network performance and cost.

The depth of a buffer is the number of flits it can hold. Fig. 3.13 and 3.14 show the load-latency curves at different buffer depth. The curves assert that increasing the buffer depth improves the network throughput. However, increasing the buffer depth increases the area of the router. Consequently, to decide the optimum buffer depth, the figure of merit (FOM) is computed according to equation 3.2.

$$FOM = \frac{NetworkThroughput}{RouterArea} \tag{3.2}$$

The computed FOM is shown in Fig. 3.15 and 3.16. The curves and the results are calculated for a packet size of two flits. The optimum buffer depth for SOTA is eight flits and for CONNECT is four flits. This is slightly greater than twice the packet size. This buffer depth provides a good network performance without excessive area overheads. Increasing the buffer depth more than this gives a higher network throughput but at the cost of the area. However, we will discuss later a way to get more buffer depth at nearly no cost.

3.4 Summary

In this chapter, NoC design parameters are analyzed using two NoCs, an ASIC-oriented NoC and an FPGA-oriented NoC. A 16-node Mesh NoC is the optimum configuration in both NoCs. VCs increase the network performance but increases the complexity of the router. The optimum number of VCs is two in both NoCs. Increasing the buffer size improves the network performance but consumes more area. A 8-flit and 4-flit buffer depth is optimum for SOTA and CONNECT respectively at a packet size of two flits. In general, CONNECT consumes smaller area than SOTA because CONNECT is designed to target FPGAs. This proves the difference between mapping an ASIC-oriented NoC

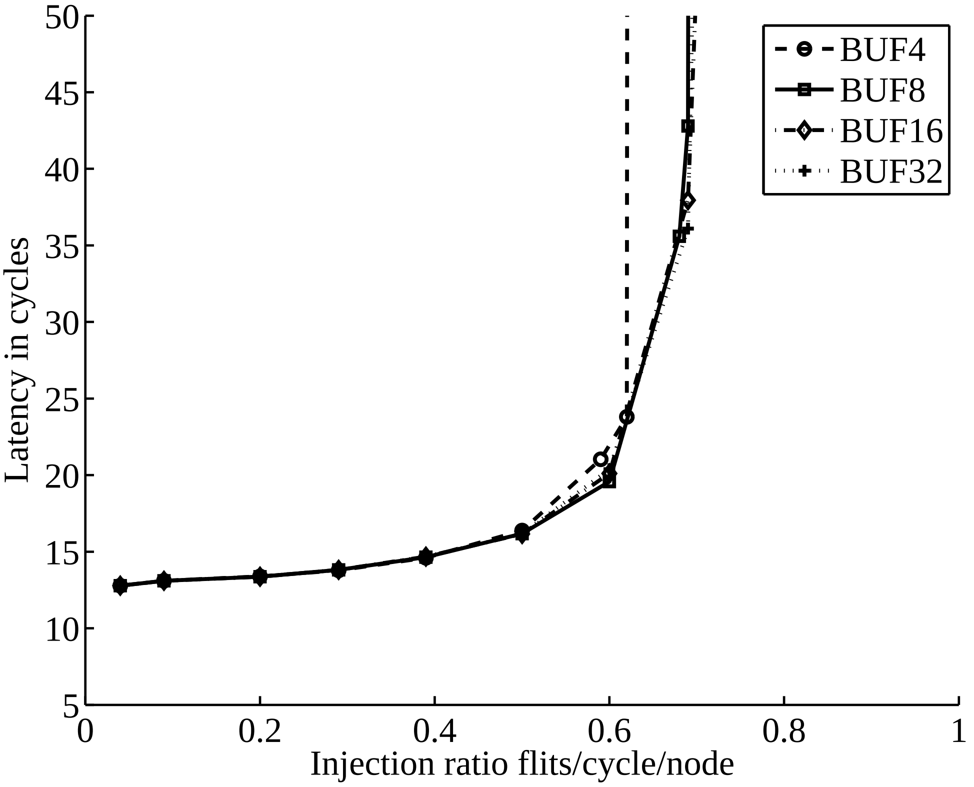


Figure 3.13: SOTA Load-latency curves at different buffer depth

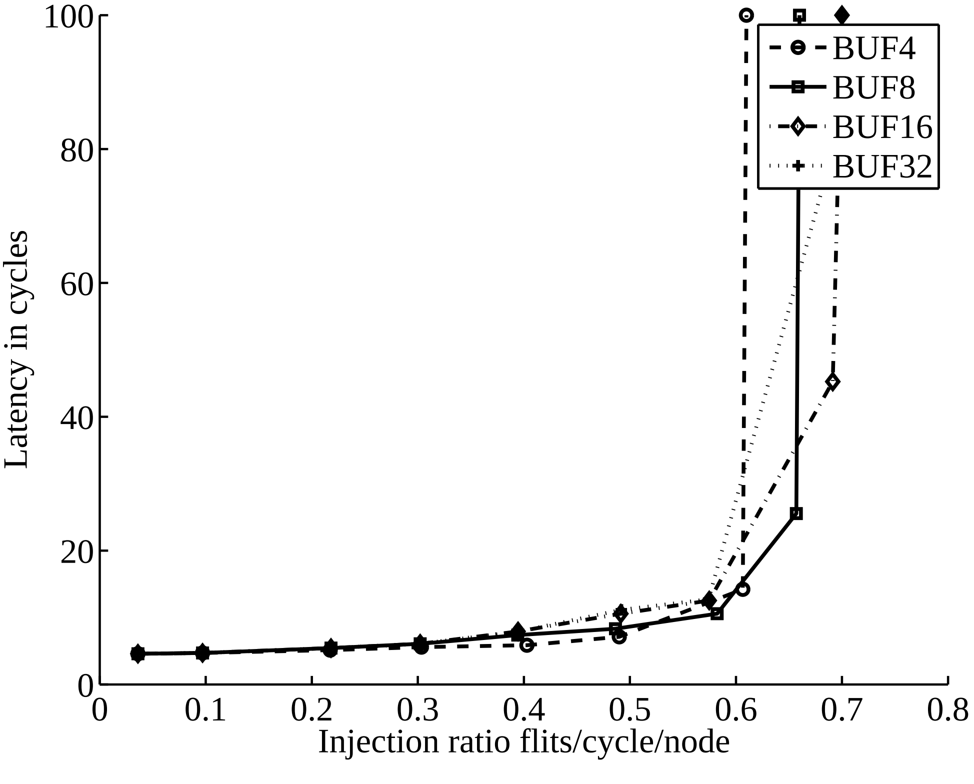


Figure 3.14: CONNECT Load-latency curves at different buffer depth

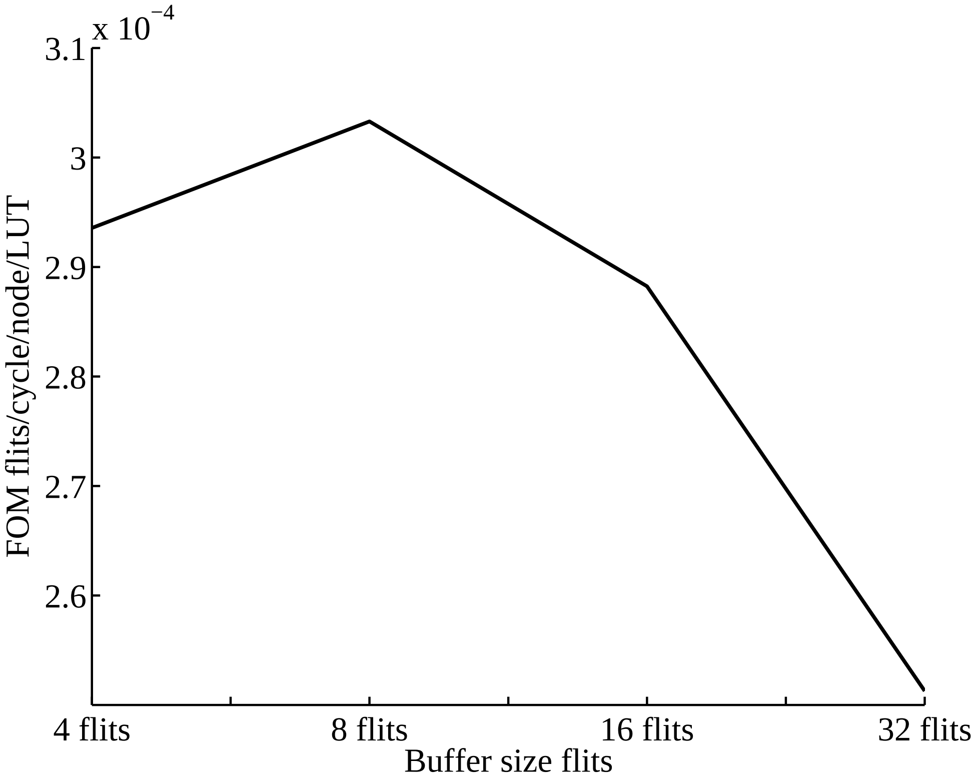


Figure 3.15: SOTA FOM for buffer depth

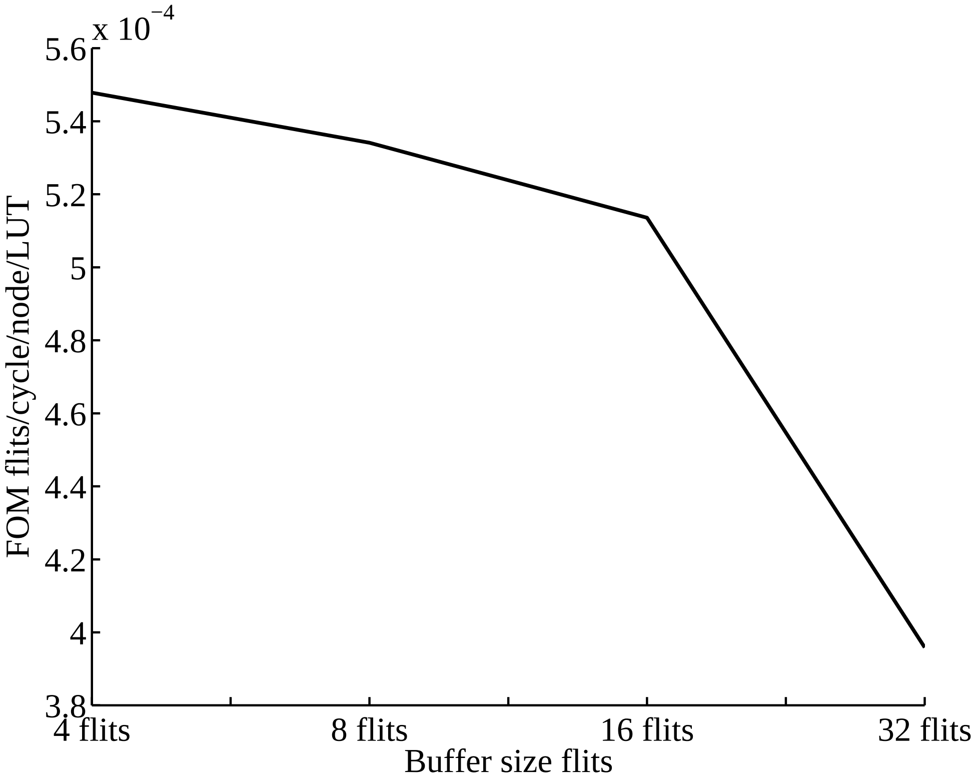


Figure 3.16: CONNECT FOM for buffer depth

onto FPGA and designing an FPGA-oriented NoC. The shown simulation results and recommendations can be used by NoC designers to select design parameters which give the optimum NoC performance for the targeted applications.

Chapter 4

Router Design

Hard NoCs have speed and area advantages over soft NoCs. However, when an implemented design doesn't require a NoC in its operation, the silicon area consumed for implementing the hard NoC is considered a wasted area. This is one of the main disadvantages of hard NoCs. Our main goal is to reduce this wasted area as much as possible by using minimum and share-able resources while providing a high performance at the same time. The router is the main contributor to the NoC area. Moreover, the router architecture has a great effect on the NoC performance. Thus, selecting the router architecture is a critical process to achieve a high performance NoC while using the minimum resources. In this chapter, we will select and implement the best-fit router architecture for our proposed hard NoC.

4.1 Selecting Router Architecture

In a packet-switched loss-less network, buffers are the most important component in the router. On the other hand, bufferless networks involve packet dropping and packet retransmission which reduce the network performance [34]. It is worth noting that recent work proposes deflection routing which can eliminate router buffers without packet dropping, however, they are suitable only for low-to-medium network loads [35, 36]. Buffers can be organized per input or per output. Placing the buffers in the output port is not practical since it requires the router to operate with the sum of the speeds of all the input ports or to use buffers that have many write ports equal to the number of input ports [37]. Consequently, NoC routers buffers are mainly organized per input which is the case in SOTA and CONNECT NoCs discussed before.

Input Buffers hold the packets that can not be forwarded immediately in queues. In a baseline input-queued router, each input port contains only one queue. This router architecture suffers from the head-of-line blocking, i.e. the first packet in the queue blocks packets behind it, that adversely affects the network performance. While in VC input-queued routers, each input port contains multiple queues, a queue for each virtual channel. VCs reduces the effect of head-of-line blocking and increases the network throughput as shown in the previous chapter.

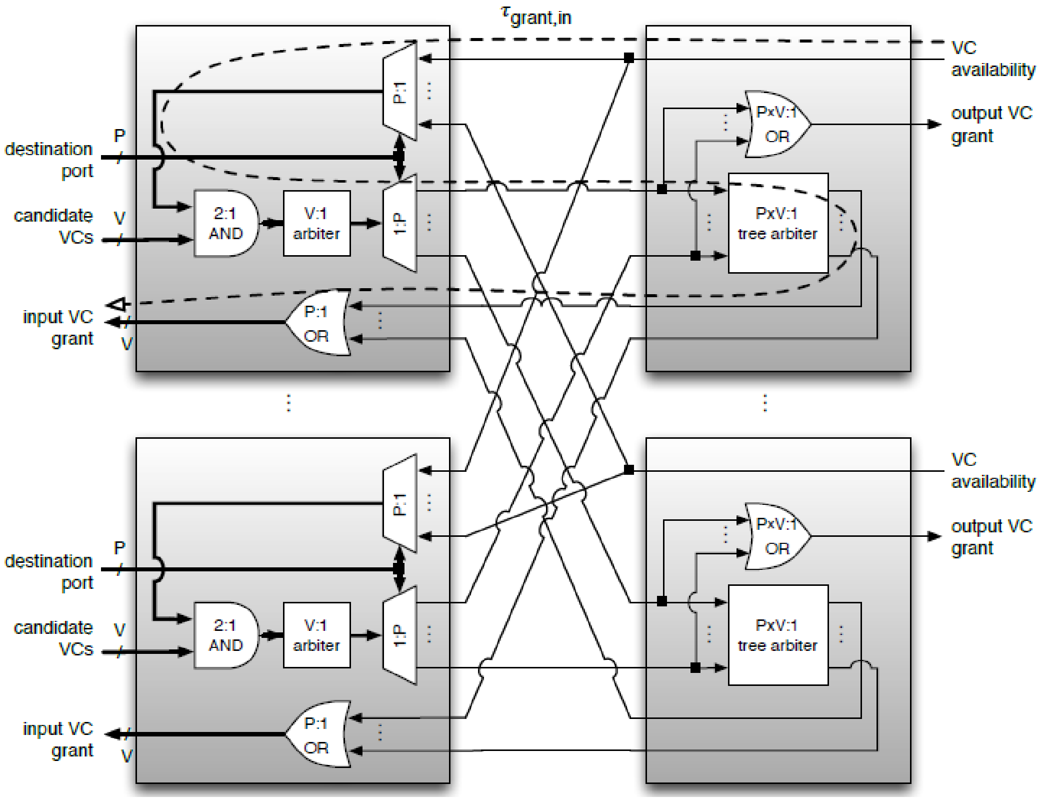


Figure 4.1: A VC allocator [27]

A conventional VC router, e.g. SOTA, consists of input buffers, crossbar, routing computation logic, switch allocator and VC allocator. It has four pipeline stages which are routing computation, VC allocation, switch allocation and switch traversal. This architecture has some drawbacks. It needs at least four cycles to forward a flit, considering each stage is performed in one cycle. This increases the network latency of the NoC. In CONNECT, all these stages are performed in one clock, i.e. single stage pipeline, which decreases the network latency but increases the cycle time, i.e. decreases the maximum operating frequency.

Moreover, the VC allocator, a crucial component of the VC router, is a very complex block as shown in Fig. 4.1. It assigns an output VC to each waiting packet in the input ports as each packet should have an exclusive access to an output VC. Thus, it generates matches between input VCs (P×V requests) and output VCs (P×V resources), where P is number of ports and V is number of VCs. This complex process consumes large area of silicon and adversely affects the maximum operating frequency of the router.

Consequently, both baseline and VC router architectures are not the optimum router architecture for our proposed hard NoC. This leads us to search for another router architecture that can provide a high network performance similar to that of the VC router architecture but with lower complexity, smaller area and higher maximum operating frequency. Virtual Output Queuing (VOQ) [37] would be an good alternative. In VOQ input-queued

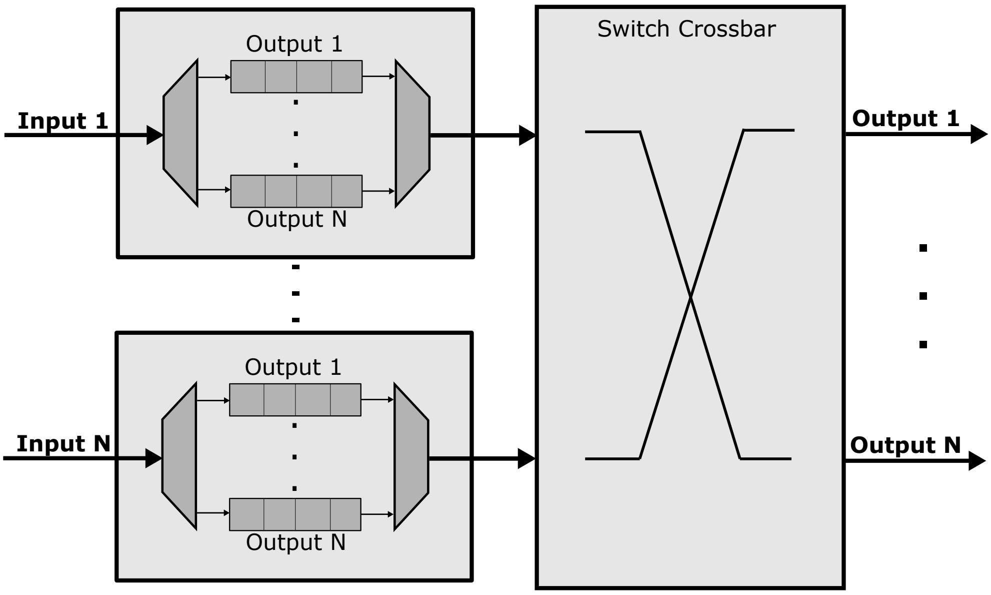


Figure 4.2: A VOQ router

routers, each input port contains a separate queue for each output port as shown in Fig. 4.2. This eliminates the head-of-line blocking which consequently improves the network performance. It also eliminates the need for VC allocation as selecting the output VC depends only on the routing computation. Thus, VOQ is the optimum solution that provides good network performance while using minimum resources.

Therefore, our proposed router will utilize VOQ scheme. According to the NoC design parameters given in Chapter 3, the NoC is 16-node Mesh. In a 2-D Mesh, a router has five ports: East, West, North, South and Local (which is connected to the network interface) as shown in Fig. 4.3. The most famous routing algorithm in 2D-Mesh is XY. In this algorithm, a packets is routed fully in X-dimension then fully in Y-dimension till it reaches its destination. This implies that only packets coming from West and Local ports can come out from the East port and only packet coming from East and Local ports can come out from the West port. Moreover, packets coming from North can only be forwarded to South or Local ports and packets coming from South can only be forwarded to North or Local ports. We will use these routing restrictions to optimize our router and minimize its area.

The packet format is shown in Fig. 4.4. Each packet consists of a head flit, a tail flit, and several body flits. The two most significant bits of a flit distinguish whether it is a head or tail or normal body flit. The following two bits represents the VC of the flit. The VC here is a function of the output port from which the flit will come out as we will discuss in detail later. As a flit can be forwarded only to four ports, i.e. all the five ports except the port it comes from, a 2-bit with four different combinations is enough to represent the VC. The head flit contains an extra field which is the required destination of the packet. In a Mesh, each router has an address consisting of two fields the X and the Y coordinates according to the router position in the Mesh considering the origin is at the bottom right

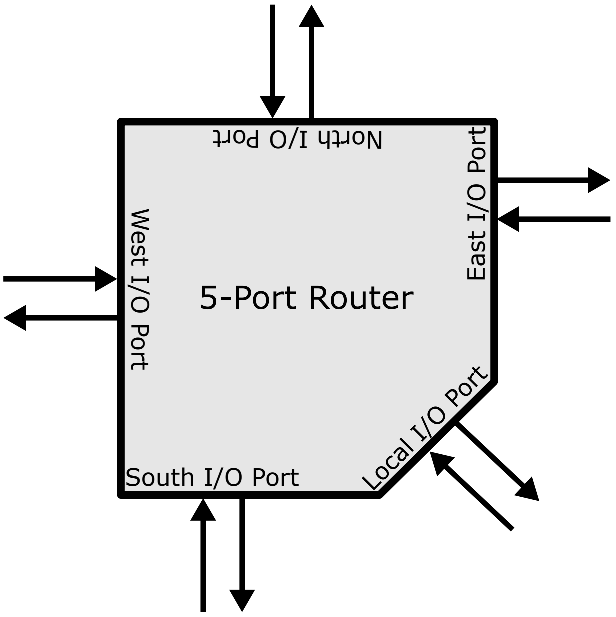
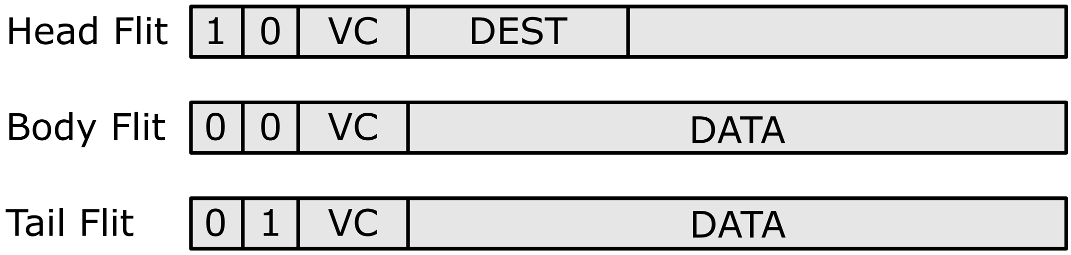


Figure 4.3: The 5-port Mesh-based router



Head Flit	1	0	VC	DEST	
Body Flit	0	0	VC	DATA	
Tail Flit	0	1	VC	DATA	

Figure 4.4: The packet format

router. Thus, in a 16-node Mesh, the X field is two bits and the Y field is two bits, making a 4-bit router address. Consequently, the destination field in the head flit is also 4-bit.

To achieve a loss-less network, a flit should only be forwarded to the downstream router if it is known that the next router has available buffer space. One of the ways to communicate the available buffer space between routers is the credit-based flow control. In this algorithm, each router keeps a count of the number of free slots of the downstream router's VC queue. When it sends a flit to the next router, it decreases this counter. The downstream router then sends a credit to the upstream router when it succeeds in forwarding this flit. This algorithm requires a counter for each VC and involves a high amount of signaling for each flit. To avoid the complexity of credit-based flow control, a simple back-pressure (BP) flow control is used. It is an on/off algorithm in which the downstream router signals the upstream router only when the buffer is full. The round-trip delay between the routers should be handled.

4.2 Router Internal Components

The internal block diagram of our five bidirectional ports router is shown in Fig. 4.5. It consists of five input modules, a switch allocator, a crossbar and five output modules. The router is connected with the other routers (or with the network interface (NI)) through the five input/output ports. Each input port has a DATA_IN port for the incoming data and a VALID_IN signal showing the validity of this data. A BP_OUT signal is generated from each input port for the BP flow control. The width of this signal depends on the number of VCs in each input port. On the other side, each output port has a DATA_OUT port, a VALID_OUT signal and a BP_IN signal connected with the downstream routers (or the NI). The remainder of this chapter deals with the detailed implementation of each block.

4.2.1 Input Module

An input module is associated with each input port. It is one of the most important blocks in the router design. Its main job is to store the incoming flits until they are allowed to traverse the crossbar switch and leave the router. In the baseline architecture, the incoming flits are stored in a single queue following First-In-First-Out (FIFO) buffering strategy. However, in the VOQ scheme, the input module should contain multiple queues (a queue for each possible output port). An incoming flit is stored in one of these different queues according to from which output port it should come out, following also the FIFO buffering strategy. All input modules have four queues representing the four potential output ports, except the input modules associated with the North and South input ports. These input modules have only two queues due to the restrictions and simplifications of the XY routing algorithm.

Conventionally, the routing computation should be done, using the destination field, before storing a flit in order to know the destined output port, hence, knowing in which queue the flit should be stored. However, to speed up our router, the routing computation is performed one hop earlier, i.e. look-ahead routing, and the result of this look-ahead routing is stored in the VC field. Consequently, based on the VC field, a flit is immediately stored in the correct queue without waiting for the routing computation step to finish. Moreover, the look-ahead routing computation, i.e. the routing computation for the next router, can be performed in parallel to the buffering process. This decreases the combinational path delay, hence, increases the maximum operating frequency. Moreover, the routing computation step no longer needs a cycle to be performed. This reduces the number of pipeline stages, hence, decreases the network latency.

The internal implementation of the input module is shown in Fig. 4.6. It consists of the buffer module, look-ahead routing (LAR) logic and the OVC buffering module. The DATA_IN port is passed to the buffer module to be stored. The VALID_IN signal, which

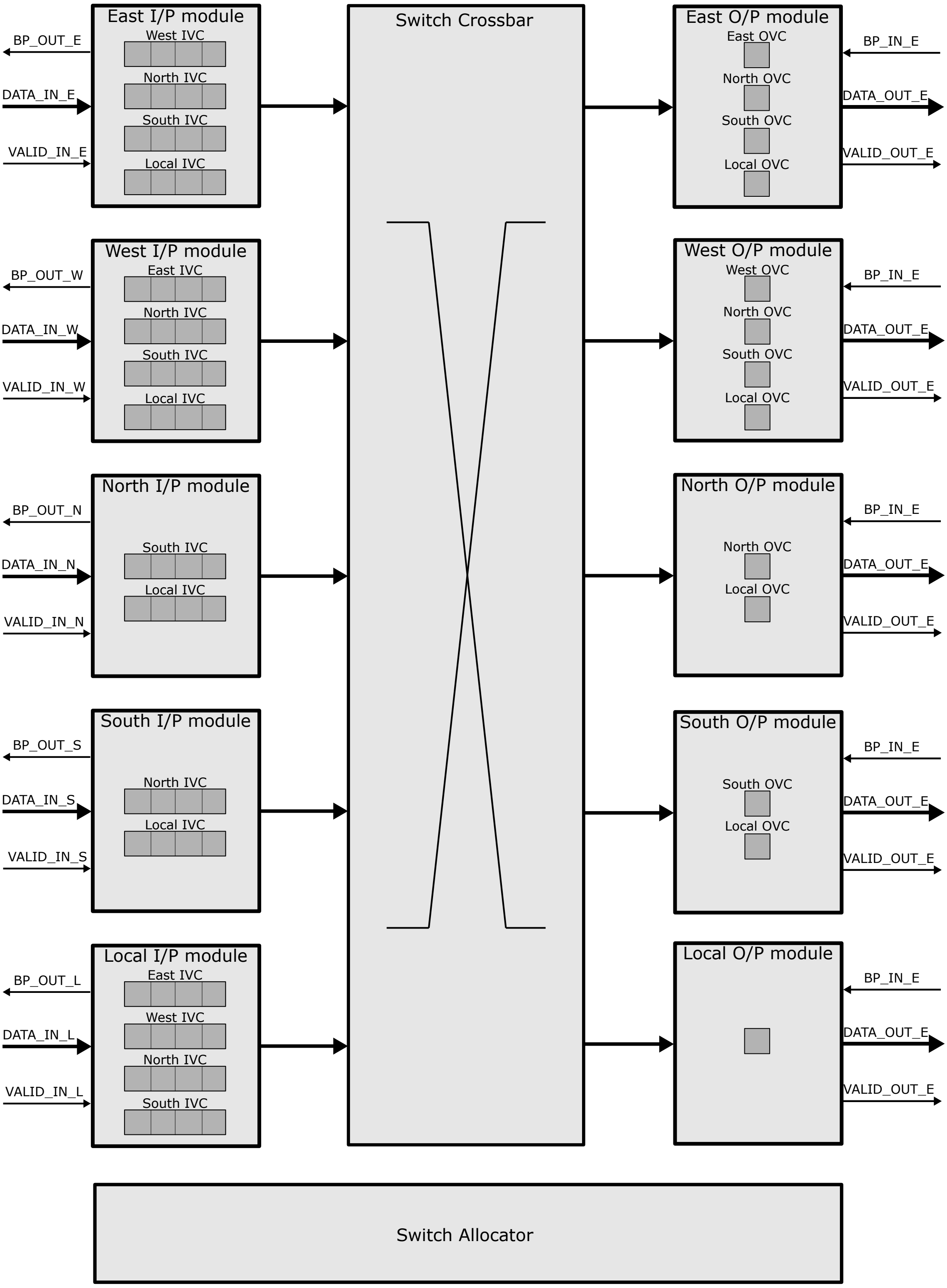


Figure 4.5: The router internal block diagram

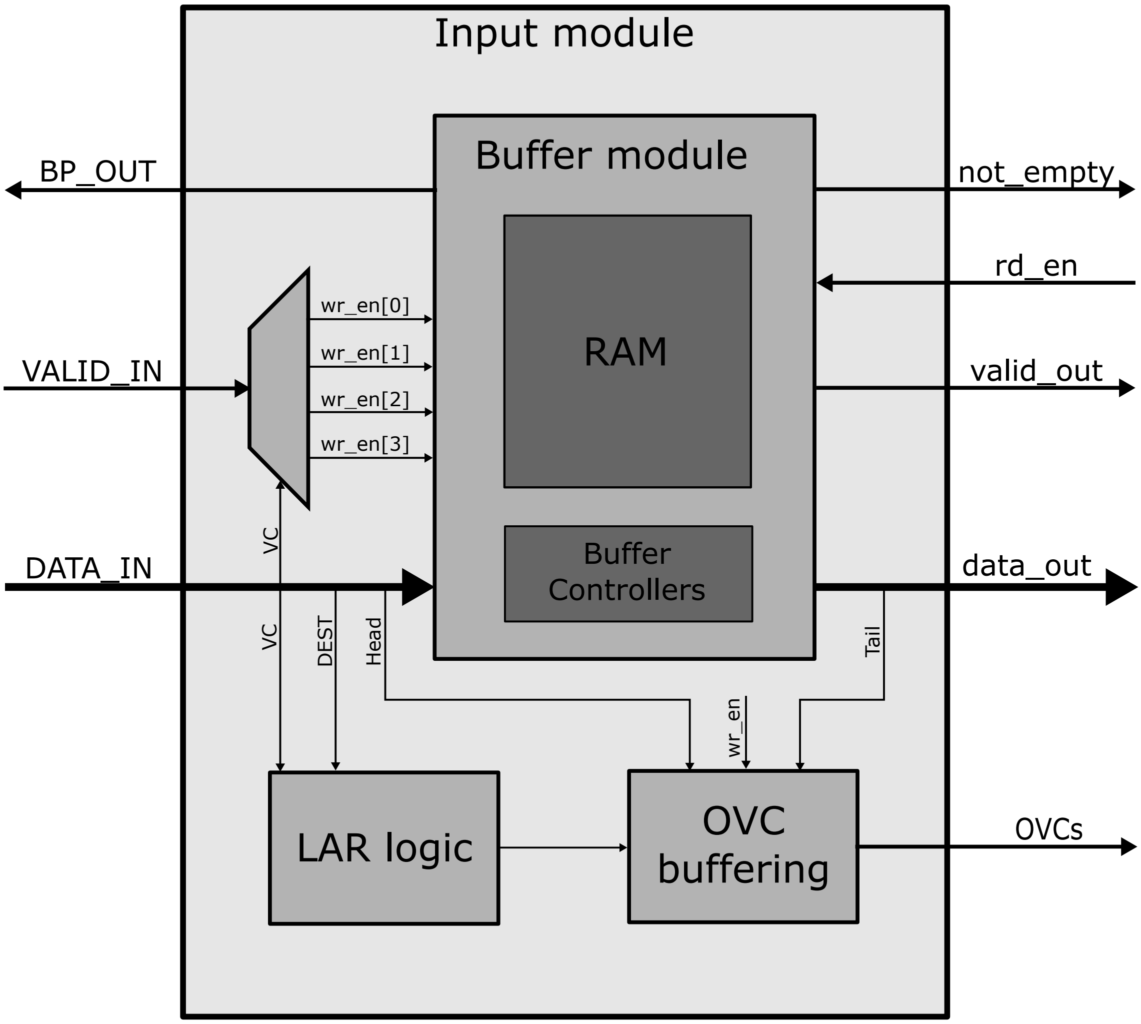


Figure 4.6: The input module

shows the availability of a new flit on the DATA_IN port, is demultiplexed to a 4-bit wr_en signal (a 2-bit in North and South input modules) according to the VC field extracted from the DATA_IN port. This informs the buffer module in which queue the new flit on the DATA_IN port should be stored.

4.2.1.1 Buffer Module

The buffer module contains the four (or two) queues that are used to store the incoming flits. Each queue can be implemented using a separate memory. However, this architecture has many disadvantages. It requires multiple small memories which is not efficient in terms of area and involves a large amount of overhead. It also increases the complexity of the crossbar switch [37]. The alternative is to implement all the queues in a single memory. This architecture is called statically allocated multi-queue (SAMQ). In SAMQ, the buffer is divided into equal portions for each queue. This doesn't affect the rate at which the queues can receive flits as there is only one input data port, i.e. DATA_IN.

In another architecture called dynamically allocated multi-queue (DAMQ), the buffer space is divided dynamically between the VC queues. This allows to dynamically control the size of each queue according to the weight of the incoming traffic. This enhances the network performance. However, implementing DAMQ buffer requires complex control such as linked list which consumes the silicon area. Therefore, an SAMQ buffer is used in our design.

The SAMQ buffer is implemented using a dual-port RAM. The width of this memory is the same as the flit width, so that a flit can be written or read in one cycle. A dual-port RAM has two separate reading and writing ports in which a reading and writing process can be performed at the same time. The write port has a data port connected to the DATA_IN signal, a write enable and a write address. The read port has a data port connected to the crossbar, a read enable signal and a read address. The write enable is the ORing of the 4-bit wr_en signal, while the read enable is the ORing of the 4-bit rd_en generated from the switch allocator. The write address and the read address are generated from the four buffer controllers (two in case of North and South ports).

A buffer controller is associated with each queue. It is responsible of controlling the operation of the associated queue. It is also responsible of keeping track of the status of the queue whether it is empty or full. Each queue is treated as a circular buffer. Each buffer controller generates a write address and a read address. All addresses are initialized with zeros. The write address is incremented when a write enable signal is high, and the read address is incremented when a read enable signal is high.

The width of these addresses are $\log_2 queue\,depth$. Therefore, if the generated address is pointing to the end of the queue, and it is incremented one more time, it overflows and automatically points to the start of the queue (i.e. circular buffer) without any need of special handling. These generated addresses from each buffer controller are then concatenated with two bits corresponding to which queue it handles, so that the addresses from each controller maps to different location in the RAM. All of these concatenated write/read addresses are then multiplexed to generate the physical write/read address of the RAM.

The buffer controller contains also a tracker counter that counts the number of flits in its associated queue. This counter is incremented when a writing process is done in the queue and no reading process associated. It is decremented when a reading process is done from the queue and no writing process associated. When a reading and writing process is done simultaneously, the counter doesn't change. When the counter is equal zero (or one and a RD_EN signal is high), this means that the queue is empty. A not_empty signal is connected with the switch allocator to inform that there is a waiting flit in this VC, so that it participates in the arbitration process.

On the other side, when the tracker counter is equal to the depth of the queue, a FULL signal is asserted. This signal is connected to the BP_OUT to tell the upstream router that there is no space in this VC in order to stop sending more flits. However, to handle the round-trip delay of the BP_OUT signal, the FULL signal is asserted early (when the counter is equal to the depth of the queue minus two). Moreover, each buffer controller generates also a valid_out signal to inform the crossbar about the validity of the RAM read data port and from which queue the data comes out to know the destined output port.

4.2.1.2 LAR Logic

The look-ahead routing (LAR) logic performs the routing computation one hop earlier. The routing computation determines the output port from which the packet should depart according to its destination. The LAR logic determines the destined output port of the next router, hence, determining the input VC of the packet at the next router (i.e. the output VC of the packet at the current router) in VOQ scheme. Thus, look-ahead routing is not important only for performance enhancement but it is important in deciding the output VC of the packet.

The LAR logic takes the destination field of the packet as an input and produces two bits representing the destined output port of the next router or the output VC of the packet. Conventionally, as the router has five output ports, three bits should be used to represent the output port. However, because of the simplification of XY routing algorithm such as that the packet can not depart from the same port it comes from, these three bits can be reduced to only two.

In XY routing algorithm, each packet is routed fully in X direction then in Y direction. Thus, to perform routing computation, the destination of the packet, known from the destination field of the head flit, is compared with the router address. If the X part of the destination is larger than the X part of the router address, the packet is forwarded to the EAST output port. If it is smaller, the packet is forwarded to the WEST output port. Otherwise, the Y part of the destination is compared with the Y part of the router address. If it is larger, then the destined output port is North. If it is smaller, then the destined output port is South. If they are equal, then the packet has reached its destination and should be forwarded to the local port.

However, in look-ahead routing (LAR), as the routing computation is done one hop earlier, the destination should be compared with the next router address. The next router address can be known from the output port to which the packet will be forwarded, i.e. the input VC of the packet. This algorithm is optimized particularly for each input module. For example, Fig. 4.7 and 4.8 show flowcharts describing the algorithm of the LAR logic in East and North input modules.

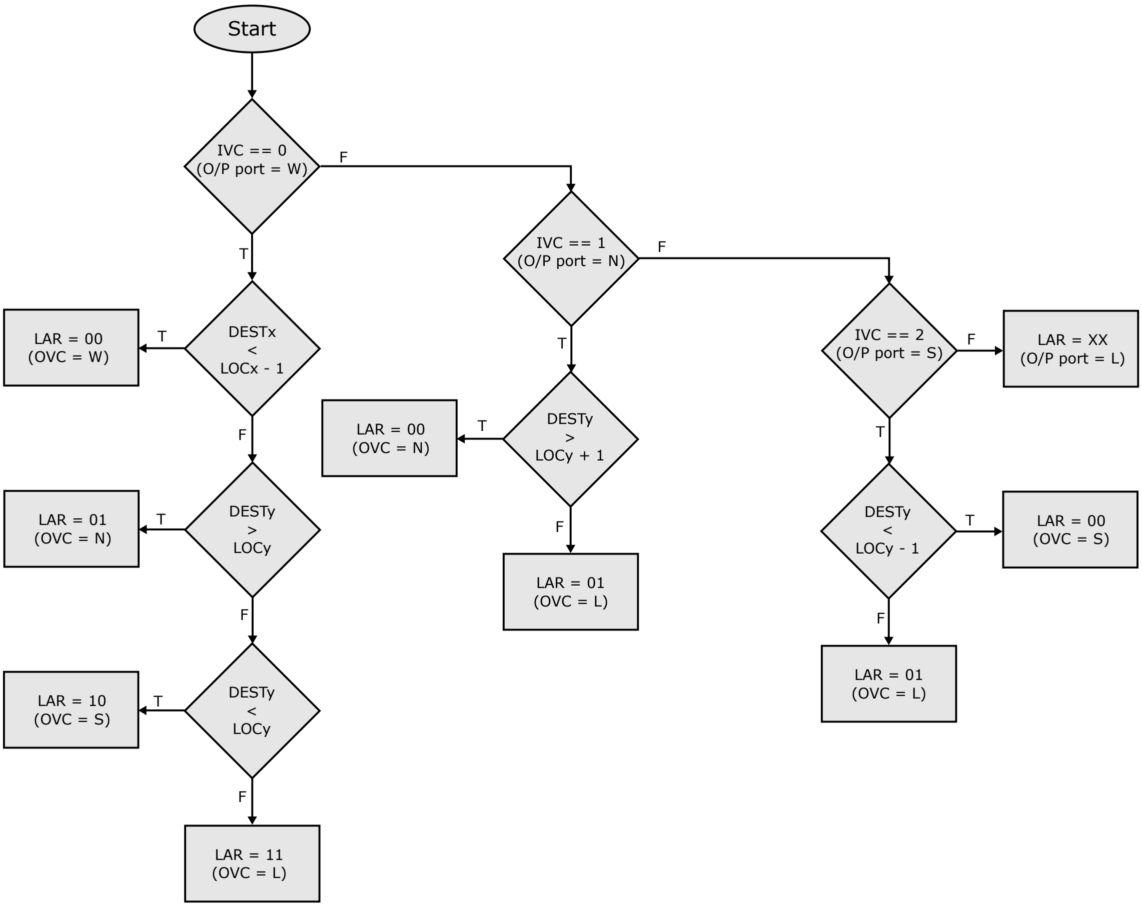


Figure 4.7: East input module LAR logic flowchart

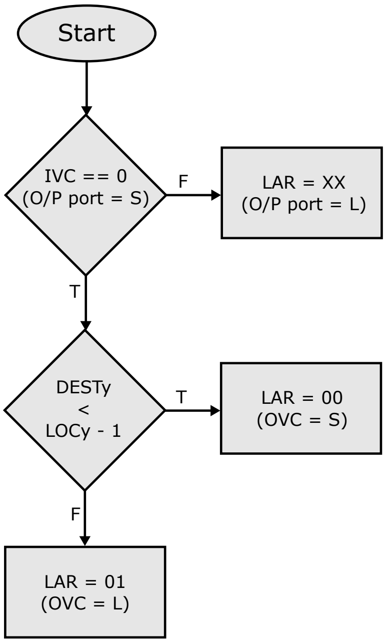


Figure 4.8: North input module LAR logic flowchart

4.2.1.3 OVC Buffering Module

Each packet is assigned to an output VC (OVC). The OVC of the packets should be communicated to the switch allocator in order not to include a packet whose OVC is full in the arbitration process. So, as we have multiple queues, the OVCs of the packets at the head of each queue should be known to the switch allocator. These OVCs are stored in separate registers not in a single buffer so that they are available all the time to the switch allocator. To allow more than one packet to be stored in a queue, a shift register is used to store the OVCs of all the packets in the queue.

The main function of OVC buffering module is to store the OVCs of all the packets buffered in all the queues and to provide the switch allocator with the OVCs of the packets located at the head of each queue. The module contains four shift registers (a shift register for each queue). The input of these shift registers is the output of the LAR logic. The output of the LAR logic is only valid when the incoming flit is a head flit (as it is the only flit that contain the destination field), so the writing in the shift registers is enabled by the ANDing of the head signal and the wr_en. The size of these shift registers is a function of the queue depth and the packet length. For example, if the queue depth is 32 flits and the packet length is 8 flits, the queue can contain flits from 5 different packets, so the size of the shift register will be 10 bits (it carries two bits OVC for each packet).

When a tail flit is leaving the input module, the OVC of this packet is removed from the shift register. The OVC of the next packet will be available next cycle. However, this leads to a pipeline stall to allow the performing of the switch allocation on the new OVC. Conventionally, to avoid this pipeline stall, the data buffer that contains the flits is implemented using a memory that has two read ports to allow the reading of the tail flit and the next head flit at the same time [6]. Such a memory is more complex, consumes more area and encounters more delay. In our design, the data buffer has only one read port, and this pipeline stall is solved in this module. This is done by always reading two consecutive OVCs from the simple shift register and selecting between them using a multiplexer. When a tail flit is leaving the input module, this multiplexer passes the second OVC. By doing this, the switch allocation step of the next head can be done in parallel with the switch traversal step of the tail flit. This is the main advantage of our OVC buffering module.

4.2.2 Switch Allocator

The switch allocator is considered as the main controller of the router. It decides when a flit is transferred from the input port to its destined output port. When a flit reaches the head of a queue, it sends a request to the switch allocator to be allowed to traverse the crossbar. As there are many queues and limited crossbar connections between input and output ports, an allocator is needed for scheduling these crossbar connections. Unlike routing computation, the switch allocation is performed on each flit not only the head flit.

In a conventional VC router, the switch allocation step is preceded with VC allocation. However, in our VOQ router, the output VC of each packet is already determined by the LAR logic. Thus, no VC allocation is needed, and flits can undergo switch allocation directly.

The switch allocator performs a matching between requests from input VCs at each input port (P^2requests in a conventional P-port VOQ router) and crossbar connections to the output ports. The quality and the fairness of this matching has a great effect on the network throughput and latency. The quality of a matching is measured by the number of generated grants. An allocator with a high matching quality improves network throughput and maximizes resource utilization. However, this comes at the expense of more area and more delay.

There are two main types of allocators: separable allocators and wavefront allocators. In general, a wavefront allocator has a higher matching quality than a separable allocator as it achieve a maximal matching [26, 27], i.e. a matching in which no more grants can be generated without replacing an existing grant. However, the best implementation of wavefront allocators consumes more area and has a higher delay than any separable allocator [27]. It also suffers from weak fairness [26]. Moreover, in a Mesh topology, the difference in matching quality between wavefront allocators and separable allocators is not translated to a much better network performance. That is mainly because of the routing restrictions which lead to a biased distribution of the requests resulting in a request matrix with many zeros. A wavefront allocator can not achieve a maximum matching if the request matrix is sparsely populated. Thus, in a Mesh, wavefront allocators and separable allocators give nearly the same network performance [27].

For the aforementioned reasons, the switch allocator of our router is implemented using an input-first separable allocator. Separable allocators divides the allocation process into two successive arbitration phases. In an input-first separable allocator, an active input VC (the requester) is selected from each input port in the first arbitration phase, then, in the second phase, the winning VCs from input ports compete for their destined output port (the resource). In this way, only one input VC from each input port can be granted. That is important because there is only one connection from each input port to the crossbar. Also, this way ensures that each output port is granted at most once to one of its requesters (input VCs). The only difference between the input-first and output-first separable allocator is that the latter performs the output arbitration phase first, however, it has a longer combinational path than the input-first allocator [27].

Each arbitration phase is performed using P (the number of router's ports) arbiters. The difference between an allocator and an arbiter is that an allocator performs matching between multiple requesters and multiple resources while an arbiter matches a single resource to one of its requesters. The arbiters in each phase work independently from each others. Thus, they can all work in parallel leading to short delay and fast allocation. The only drawback is that the independent arbiters in the first phase can grant multiple VCs requesting the same output port which leads to a non-maximal matching. The quality of matching can be increased by carrying out multiple iterations of separable allocation such as in Parallel Iterative Matching (PIM) method (not practical in NoCs due to delay constraints) or by managing arbiters priorities using iSLIP scheduling algorithm [27, 38].

In a conventional five-port VOQ router, a ten 5-input arbiters are required to perform separable switch allocation. However, due to the routing restrictions in our NoC, the request matrix R, i.e. a matrix in which rows representing input ports (East, West, North, South and Local) and columns representing output ports, would be as shown in equation 4.1 where “X” means impossible requests.

$$R = \begin{bmatrix} X & 1 & 1 & 1 & 1 \\ 1 & X & 1 & 1 & 1 \\ X & X & X & 1 & 1 \\ X & X & 1 & X & 1 \\ 1 & 1 & 1 & 1 & X \end{bmatrix} \tag{4.1}$$

This simplifies the hardware to only six 4-input arbiters and four 2-input arbiters as shown in Fig. 4.9. The arbiters in the first stage, i.e. input arbiters, receive requests from all the input VCs. A request is sent only when the queue representing an input VC is not empty, which means that there is a flit at the head of the queue waiting for switch allocation. So, the input of input arbiters is the NOT_EMPTY signals generated from all input modules’ queues. However, as we are using a back-pressure flow control, an input VC can not be granted access to the crossbar if the destined output VC is full (the queue corresponding to this output VC in the downstream router has no space). Thus, an information about the destined output VC of each input VC is needed which is provided by the OVC buffering module. This information is used to check if the destined output VC of each input VC is full or not. If the BP_IN of this output VC is asserted, the input VC request is deleted.

Moreover, it is possible that multiple packets desire to be forwarded to the same output VC. However, to ensure the correctness of packets delivery, a packet can not be forwarded to an output VC that is currently in use by another packet. Consequently, it is required to track the status of each output VC and check whether it is in use or not. In a conventional VC router, this check is performed in the VC allocator, however, as there is no VC allocator in our VOQ router, the tracking hardware is implemented in the switch allocator.

The proposed tracking hardware consists of a simple register associated with each output VC. Each register contains number of bits corresponding to the number of potential input VCs requesting this output VC. For example, a 2-bit register is attached with East output VC in the East output port, as only the first input VC in West input port and the first input VC of the local port could forward a flit to this output VC. All the registers are initialized with ones. When a grant is generated for a certain input VC, the register

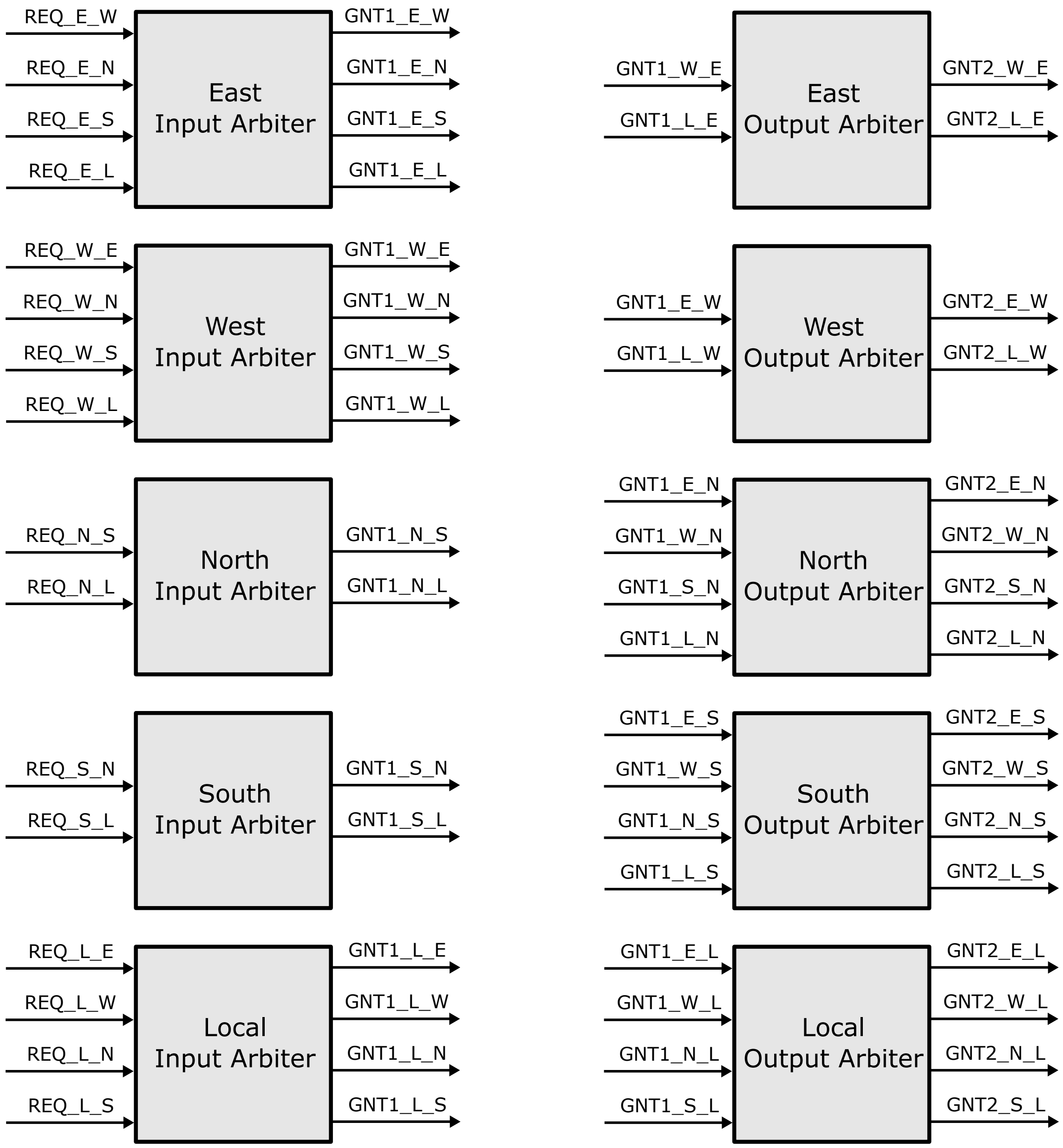


Figure 4.9: The switch allocator

attached with the destined output VC is zeroed except the bit corresponding to the granted input VC. This reflects that this output VC is currently assigned for this certain input VC so that when an another input VC requests to be forwarded to the same output VC, the corresponding bit to this input VC will be used to delete the request. When a tail flit from the granted input VC is forwarded to its output VC, the associated register is initialized again with ones, permitting any further access to this output VC.

Input arbiters then choose a winning input VC for each input port. The requests from these winning input VCs are then passed to the arbiters in the second stage, i.e. output arbiters. Output arbiters then choose a winning input port for each output port. The grants from this stage determine who win the access to the crossbar. Thus, these grants are used as the rd_en signals of the buffers in input modules to prepare the flit at the head of the winning input VC for switch traversal. Moreover, according to these grants, the switch allocator informs the output modules about the winning input VCs and their destined output VC.

In separable allocators, it is clear that the most important building block is the arbiter. It affects the speed and the area of the switch allocator. Thus, the implementation of the arbiter should be handled carefully. Moreover, the switch allocator fairness, which affects the network throughput of the router, depends on the used arbiter's fairness. A fixed priority arbiter does not guarantee any fairness. A variable priority arbiter changes the priority of the requesters from cycle to cycle to prevent starvation of any of them. Round-Robin (RR) arbiters provide strong fairness by assigning the lowest priority to the last granted request [26]. In our paper [39], a detailed comparative review of different RR arbiters' architectures are provided [39]. Parallel Prefix RR arbiter is one of the fastest RR arbiters. It generates each grant signal using a parallel prefix tree. However, it is not scalable and consumes large area in large N-input arbiters. As we only need 2-input and 4-input arbiters in our design, a parallel prefix arbiter is an optimum choice as it provides the highest maximum operating frequency while consuming a small area.

In a conventional RR arbiter, when a request is granted, the arbiter updates the priority with the new priority vector which is a shifted version of the grant vector. However, to improve the network performance of the router, the iSLIP scheduling algorithm is used [38]. In this algorithm, the priority of the internal arbiters, i.e. the arbiters in the first arbitration stage, doesn't change unless the granted requests of these arbiters are also granted in the second arbitration stage.

4.2.3 Crossbar Switch

A multiplexer-based crossbar switch is used to connect the input modules with the output modules. The crossbar allows the traversal of five flits simultaneously from different input ports to different output ports. A multiplexer-based 5x5 crossbar, that connects five inputs to five outputs, normally consists of five 5-port multiplexers (the width of each port

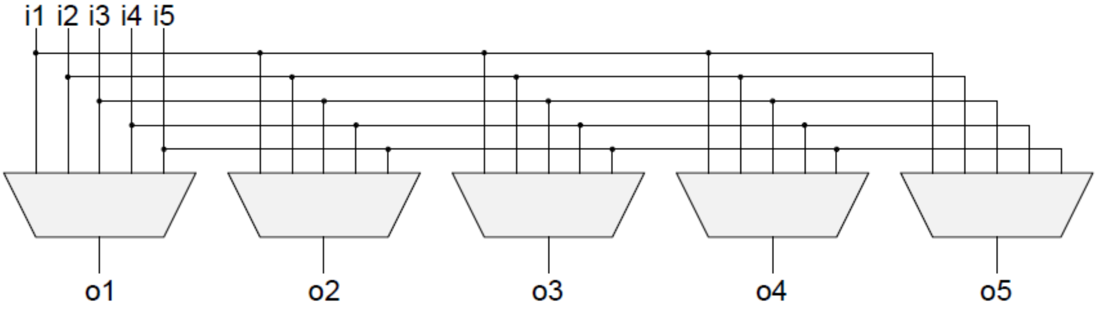


Figure 4.10: A 5-port multiplexer-based crossbar switch

equals the flit width) as shown in Fig. 4.10. However, due to routing restrictions in XY algorithm, the crossbar needs only three 4-port multiplexers and two 2-port multiplexer as shown in Fig. 4.11. This is a huge reduction in area and complexity.

4.2.4 Output Module

After the switch traversal, the flits reach the output modules. In the output module, the original VC field of a flit is replaced with its output VC, i.e. its input VC in the next router. To enhance the maximum frequency of the router, a pipeline stage can be added here to cancel the effect of link delay between routers. Thus, the incoming flits from the crossbar are stored in registers before leaving the router. If the link delay is not large, this pipeline stage can be removed reducing the pipeline stages of the router to only two stages: switch allocation and switch traversal.

4.3 Summary

In this chapter, we investigated different router architectures to select the best-fit one for the hard NoC. A VOQ architecture was found to be a suitable choice as it provides good network performance while using minimum resources. Then, we designed and optimized the internal components of the VOQ router according to the selected design parameters in the previous chapter.

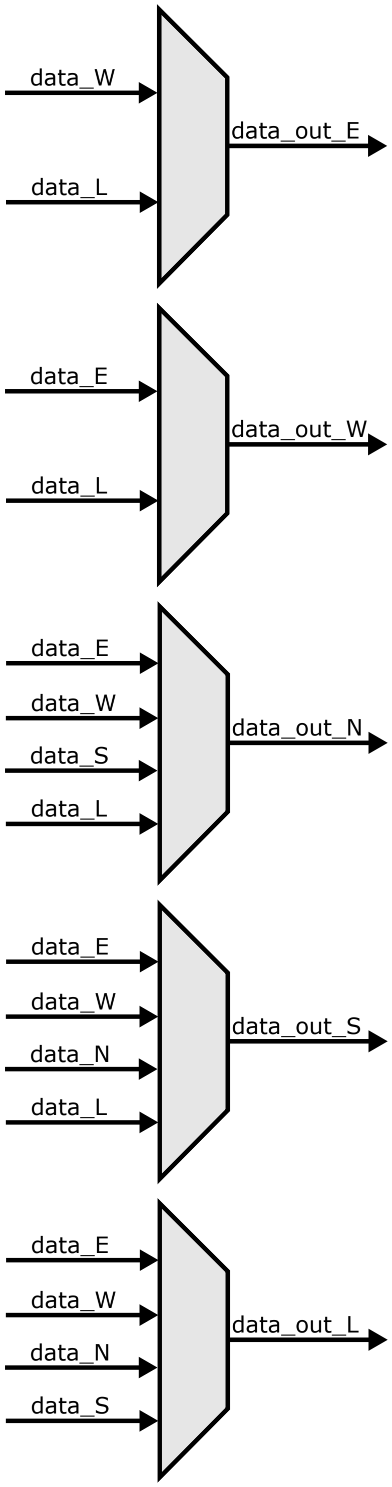


Figure 4.11: The crossbar switch

Chapter 5

Embedding the Hard NoC

In this chapter, we design the network interface that connects the soft modules on the FPGA fabric to the NoC. We also propose some ideas about embedding the hard NoC into the FPGA.

5.1 Network Interface

The network interface (NI) is the second main component in a NoC which is used to interface the IP modules to it. In FPGA-based hard NoCs, the NI's main purpose is to provide the FPGA fabric an access to the high communication bandwidth of the hard NoC. It connects the soft modules implemented on the FPGA fabric, running at any frequency, to the high speed NoC. Obviously, the NoC, implemented on silicon, is much faster than the soft modules. Thus, to achieve the highest performance, the hard NoC should runs at its maximum frequency while the slow soft modules should have a wider datapath than the hard NoC. This allows the fabric to achieve and utilize the large NoC bandwidth. It also allows the NoC to have narrow datapath and narrow link width which decrease the NoC area. The NI should perform clock crossing and width adaptation between the NoC and the FPGA fabric to match their bandwidth [18].

Our target is to design a simple NI, completely implemented on silicon, that provides an easy interface to use the NoC without involving the user in the NoC design details, and doesn't cause any degradation in the NoC performance or throughput. The NI is divided into two parts, a part for each direction between the fabric and the NoC.

5.1.1 From FPGA Fabric to NoC

The main functionality of this part of the NI is to take the outgoing data, required to be sent on the NoC, from the soft modules and inject it, in the form of network flits, to the local input port of the associated NoC router. The hard NoC approximately runs 4x faster than the FPGA fabric [16]. Thus, to match their bandwidth, [16] proposes to use a time-domain multiplexing (TDM) with a TDM factor of 4:1. So, every incoming data from the soft modules will be packetized into a packet of four flits. Then, the flits are injected one

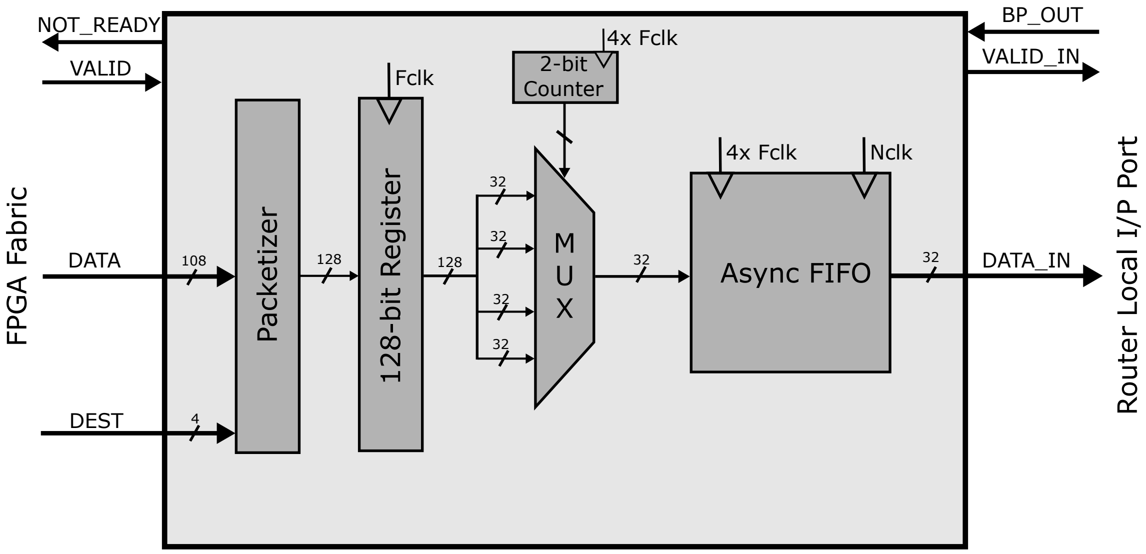


Figure 5.1: Network Interface: From FPGA fabric to NoC

by one into the NoC. This configuration allows the FPGA fabric to fully utilize the NoC bandwidth.

The block diagram of this part of the NI is shown in Fig. 5.1. The first block is the packetizer. It encodes the data into four flits and adds the control bits to each flit. Each flit has four control bits (H, T, and 2-bit VC) except the head flit has extra four bits for the destination. This sums up to 20 control bits. The 2-bit VC is calculated through a look-ahead routing (LAR) logic, while the H and T is constant depending on the type of the flit. The destination of the packet, which is needed in the head flit and to calculate the VC bits, is set by two ways. It can be inputted to the NI every fabric cycle with the data incoming from the soft modules. This allows the soft module to communicate with multiple different modules at different destination across the NoC. It is also important in case of multiple modules are connected to the same NI. The destination can also be configured during the FPGA configuration time to a certain value. This is used if the connected module has fixed destination.

For a flit width of 32 bits, the 20 control bits is added to the 108-bit incoming data to form the four flits. The four flits is then registered, if the VALID signal is asserted, in a 128-bit register every fabric clock cycle. The output of the register is connected to a 4-input 32-bit TDM multiplexer. The selection line of the multiplexer is a 2-bit counter with a frequency equals to 4x the fabric frequency. The output of the multiplexer is then connected to a small dual-port asynchronous FIFO. The asynchronous FIFO is used to perform clock domain crossing between the fabric frequency and the NoC frequency. This allows the soft modules to run at any speed instead of being fixed to quarter the speed of NoC (The optimum performance is at NoC frequency equals four times the fabric frequency due to 4:1 TDM factor). The (full) signal of this FIFO is used to generate the NOT_READY signal that informs the soft modules that the NoC is not ready for receiving extra data.

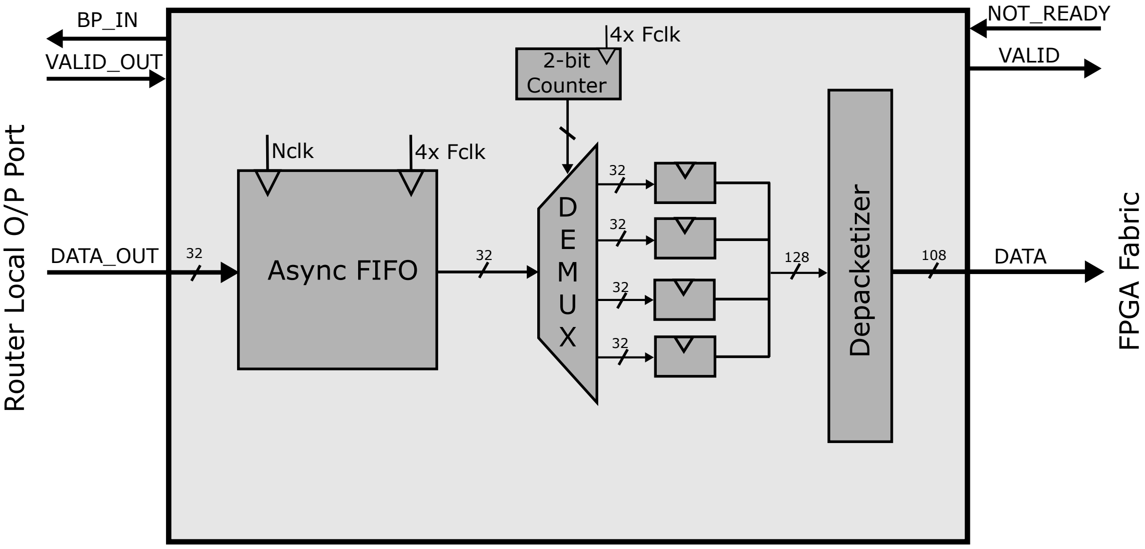


Figure 5.2: Network Interface: From NoC to FPGA fabric

The output of the FIFO is connected to the Local input port of the associated router. As long as the FIFO is not empty and the BP signal from the router Local port is not asserted, the FIFO injects a flit into the NoC every NoC clock cycle.

5.1.2 From NoC to FPGA Fabric

This part is the inverse of the previous part. It receives the flits from the NoC, then decodes it and delivers the raw data to the soft modules at their running speed. The block diagram is shown in Fig. 5.2. The incoming flits from the local output port of the associated router is buffered in a small asynchronous FIFO to convert to the fabric clock domain (4x fabric frequency). A TDM demultiplexer is then used to buffer the four flits of the same packet into four 32-bit registers. The control bits are then removed from the output of the four registers, and the 108-bit raw data is delivered to the soft module with a VALID signal. If the soft module is not ready to receive the incoming data, it asserts the NOT_READY signals which pauses sending the incoming data to the soft modules.

As illustrated above, this NI eases the interfacing of soft modules with the NoC. When a soft module needs to send a data to another module connected to the NoC, it only outputs that data to the DATA_IN port of the NI and asserts the VALID signal. It doesn't need to know anything about the internal architecture of the NoC. Then, the destined soft module will receive that data without any modification through the DATA_OUT port of the NI with an asserted VALID signal showing its validity. The frequency of the soft module doesn't have to be related to the NoC frequency. Moreover, the frequencies of the two communicating modules don't have to be the same. Thus, the user of the FPGA no longer has to deal with the clock domain crossing and metastability issues when connecting two different frequency modules.

Multiple soft modules can be connected to the same NI by using small soft logic. If all these soft modules have the same destination, a soft logic can simply concatenate the data of all of them. However, if they have different destinations, a soft logic can multiplex between them using round-robin arbitration and sets the DEST port of the NI to the correct value.

5.2 Sharing Resources

Until now, we have designed the router (and NI) and minimized the resources it takes by optimizing it according to the selected NoC parameters. Our aim is to further decrease the wasted silicon area when the hard NoC is not in use. Thus, we also work on sharing the NoC resources, or in other words, exploiting it when the NoC is not needed. The largest contributor to the NoC area, and also the NoC power, is the input buffers of the routers [27, 40]. In our optimized VOQ router, the buffers account for around 85% of the overall area of the router. Reducing the buffer depth decreases the area consumed by the buffers, hence, the overall area of the NoC. However, as illustrated before, it adversely affects the network performance especially the maximum throughput. Thus, the buffer depth is not reduced in our design, however, instead of implementing NoC buffers using dedicated memories that can only be used by the NoC, we propose to implement them using memories that can be accessed and utilized by the FPGA fabric.

FPGAs have two type of internal memories. The first type is distributed RAM which is a small size memory implemented using LUTs. The another type is called block RAMs. Block RAMs, shown in Fig. 5.3, are dedicated hard blocks of several kilo-bits of static RAM. They have configurable data width and depth. They can also act as single-port memory, simple dual-port memory (one read port and one write port), or true dual-port memory (two read/write ports). Block RAMs are very valuable resources that are essential in every nowadays FPGAs. They are placed in columns distributed among the FPGA as shown in Fig. 5.4. Compared to distributed RAM, they are more compact and way more efficient in implementing large memories. For example, in Altera Stratix III FPGA, the bit density of 9-kbit block RAM is 142 kbit/mm2 while it is only 23 kbit/mm2 for distributed RAM, which means that a block RAM with a 16% utilization is more area efficient that a fully utilized distributed RAM [41].

Therefore, we propose to implement the input buffers using FPGA block RAMs. This can be viewed as either sharing existing FPGA block RAMs with our NoC instead of implementing our own dedicated memories (saving area), or implementing our own memories that can be shared with FPGA fabric to use them as conventional block RAMs (adding more valuable resources to FPGA). In order to implement routers' buffers using FPGA black RAMs, they must match one of the configurations of block RAMs memories. For example, SOTA router's [27] input buffers can not be implemented using block RAMs as they require a memory that has one write port and two read ports to prevent a pipeline

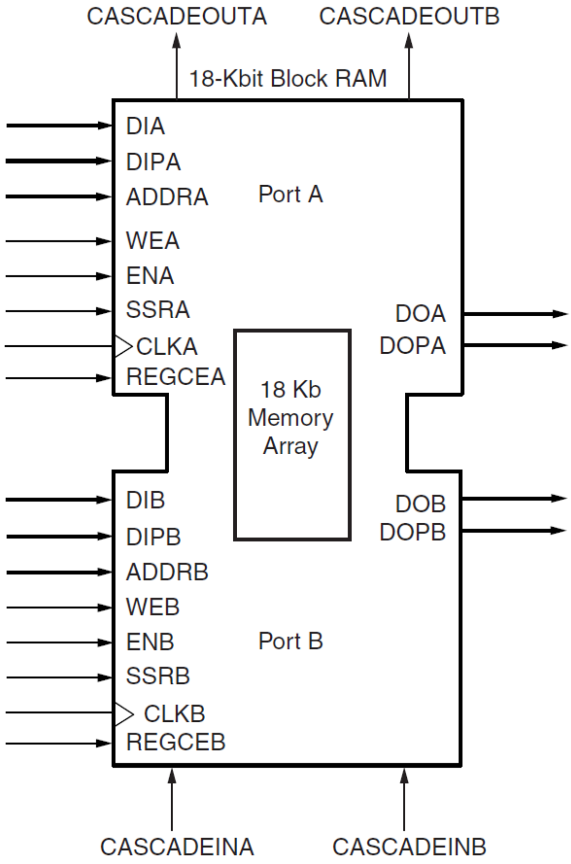


Figure 5.3: Xilinx 18 kb Block RAM

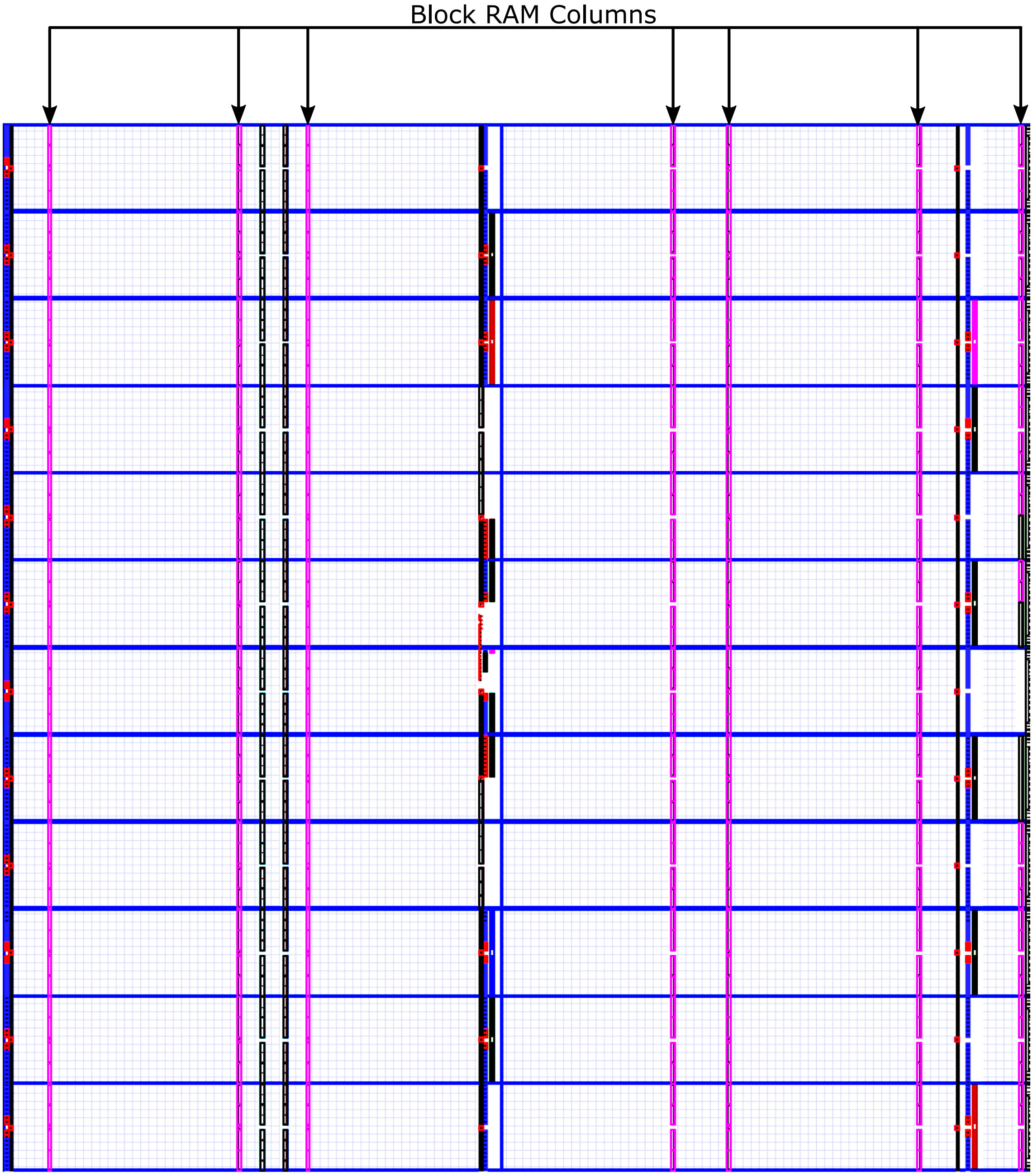


Figure 5.4: Xilinx Virtex 5 Floorplan

stall in allocation stage, while block RAMs can only support two read/write ports (It can only be done by using two block RAMs for each memory). In our optimized VOQ router, a simple dual-port memory is needed to implement the input buffers (the pipeline stall is solved by the OVC buffering module). Consequently, a conventional FPGA block RAM can be used.

This architecture dramatically decreases the wasted silicon area; The memory units that act as the routers' input buffers can normally be used and accessed by the FPGA fabric when the NoC is not in use. The detailed connections between the hard router, block RAMs and FPGA fabric will be discussed at the end of this chapter. This architecture also provides the ability to increase the buffer depth until the maximum size of the block RAM which will further improve the network performance at almost no cost. In addition, Block RAMs have built-in error correction which adds more reliability to our NoC.

5.3 Keeping the Homogeneity

In this section, we discuss the ways of embedding the hard NoC inside the FPGA. This involves the location of the routers and the connectivity between the FPGA fabric and the network interface. Embedding NoCs inside FPGAs is a critical process. Our target is to maintain the structure of tiles and routing resources as much as possible, or in other words, keeping the homogeneity of the FPGA.

In our proposed hard NoC, routers, network interfaces and links are implemented on silicon. The hard NoC consists of 16 nodes in a mesh topology in which the 16 routers are distributed evenly across the FPGA. Routers are connected together using direct hard links providing a higher communication layer over the existing FPGA routing resources. Each router local port is connected to a network interface. Network interfaces are connected to the soft interconnect of the FPGA fabric. The question is how to embed these hard blocks (routers and network interfaces) inside the FPGA fabric? Also, how to connect network interfaces to the FPGA soft interconnect?

To address these questions, we look at nowadays FPGAs that contain embedded hard blocks like block RAMs and DSPs. Hard blocks use the existing routing channels when interfacing with the FPGA and doesn't alter the FPGA routing structure. This is in order to preserve the regularity and the homogeneity of the FPGA. The FPGA regularity is crucial in manufacturing which is the reason for its high yield and its fast adoption of emerging technologies [42].

The previous trials to embed the hard NoC in FPGAs, with their contemporary structure, have followed the same principle of using the existing routing channels when embedding hard blocks. They have suggested that routers replace some logic clusters [6]. Moreover, they have used the routing resources of these clusters to connect the router to

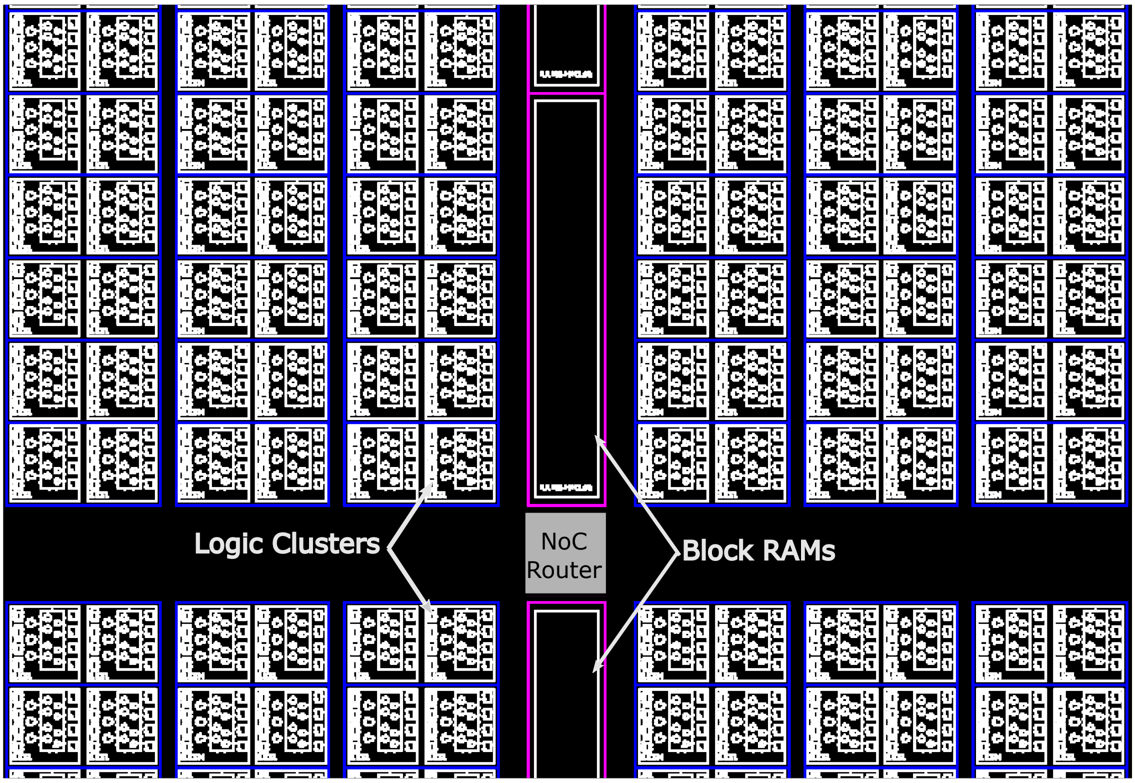


Figure 5.5: Proposed NoC router placement

the FPGA Fabric. This method preserves the routing resources structure. However, replacing some logic will adversely affect the homogeneity of the existing arrangement of the logic clusters (stacked in full columns). This may complicate and increase the amount of strain on CAD tools and flow.

Therefore, we avoid in our design to remove any logic cluster in order to keep the homogeneity of the FPGA. Moreover, as we have proposed to implement the input buffers of the router using FPGA block RAMs, the router would better be placed near them. For the aforementioned reasons, we propose to embed the routers in the same column of block RAMs as illustrated in Fig. 5.5. This configuration does not change the structure of tiles. It reduces the amount of strain on CAD flow, as the hard router will be treated the same as the hard block RAMs.

In order to preserve also the homogeneity of routing resources, the connections between the router and block RAMs, and between the network interface and the FPGA fabric must be handled carefully. During NoC operation, block RAMs that act as input buffers are fully controlled by the hard router. This means that the routing resources, i.e. multiplexers, tri-state buffers and wires, that connect the block RAM to the Fabric routing channels are not in use during the NoC operation. Consequently, these routing resources can be utilized to connect the network interface to the fabric. The network interface has 114 input ports and 110 output ports, which is much less than the I/O ports of the five used block RAMs. This allows more flexibility in connecting the network interface to the FPGA fabric.

To connect the router with the block RAMs, a simple 2-port multiplexer is added at each port of the block RAM to select whether it is connected to the FPGA fabric or to the hard router. These multiplexers and the wiring should be inserted inside the column containing the block RAMs in order not to alter the regularity of the soft fabric. The selection lines of these multiplexers are controlled by SRAM bits that are programmed during the conventional FPGA configuration process in the same as programmable multiplexers of the routing resources. Fig. 5.6 illustrates the connectivity among the Block RAMs , the hard router, the network interface and the FPGA fabric.

Consequently, the proposed hard NoC has two modes of operation. First, when the NoC is needed in a certain design, the added multiplexers are programmed to connect the adjacent block RAMs to the hard router to act as its input buffers, and to connect the network interface to the FPGA fabric, allowing the FPGA fabric to use the high communication bandwidth of the NoC. On the other hand, when an implemented design does not need the NoC, the block RAMs are disconnected from the hard router and can be used normally by the FPGA fabric.

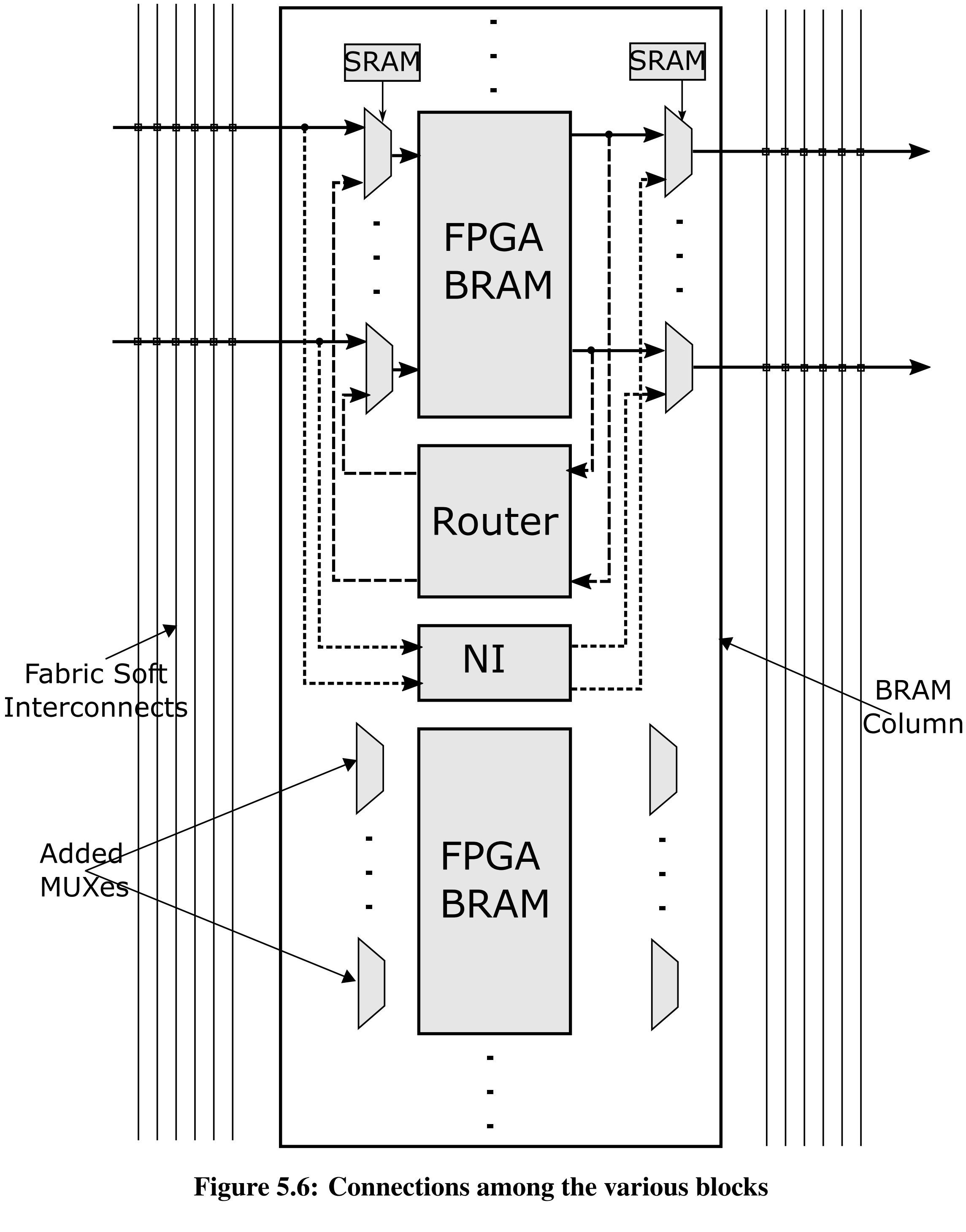


Figure 5.6: Connections among the various blocks

Chapter 6

Results and Discussion

In this chapter, we evaluate the performance of the proposed hard NoC and compare it with other NoCs presented in literature. In order to evaluate the NoC according to various performance measures discussed in Chapter 3, a parameterized RTL implementation of the proposed router is developed using Verilog (the industry-standard hardware description language).

6.1 Network Performance

To measure the network performance of the proposed NoC, the standard setup for evaluating interconnection networks, which is discussed in Chapter 3, are used. Sixteen routers are connected together to form a 4 x 4 mesh network. A packet source is connected to the local input port of each router. Both the packet injection rate and the traffic pattern of the packet source are configurable to allow evaluating the network performance at different loads. The packet source performs look-ahead routing to determine the VC field bits of the generated packets according to their destination. Network throughput and network latency are measured by RTL simulations using the same way described in Chapter 3. The "accepted vs. offered traffic" and "load-latency" curves of the implemented NoC, under different traffic patterns, are shown in Fig. 6.1 and 6.2. The proposed NoC has a maximum throughput of 0.66 flits/cycle/node and a zero-load latency of 8.6 cycles under uniform traffic.

6.2 Hard and Soft Implementation

The proposed router is implemented on UMC's 65 nm ASIC technology (hard implementation). The area, delay and power estimation of the implemented router are provided by Synopsys Design Compiler 10.3.

First, the whole router (with internal input buffers) is implemented. This implementation will not be used in our hard NoC as we have proposed to use the FPGA block RAMs as our input buffers. It will be used only to compare our proposed router with other routers presented in literature. The design achieves a maximum operating frequency of 1 GHz. It has a dynamic power of 39 mW. The reported router area is 67000 μm^2.

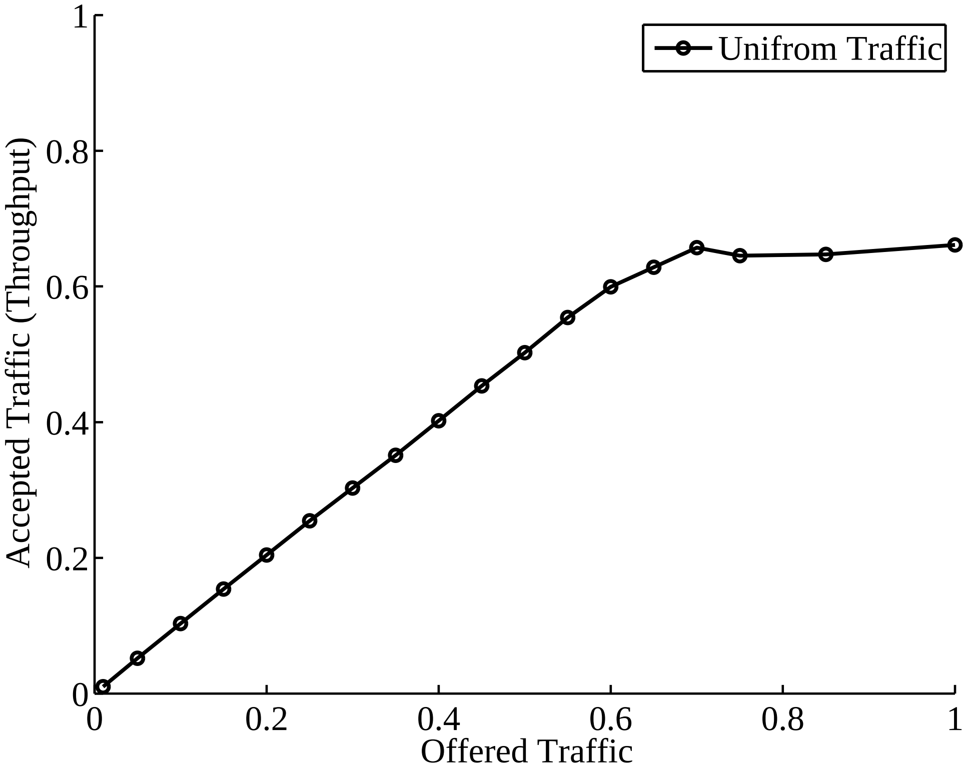


Figure 6.1: Accepted traffic vs offered traffic curves under different traffic patterns

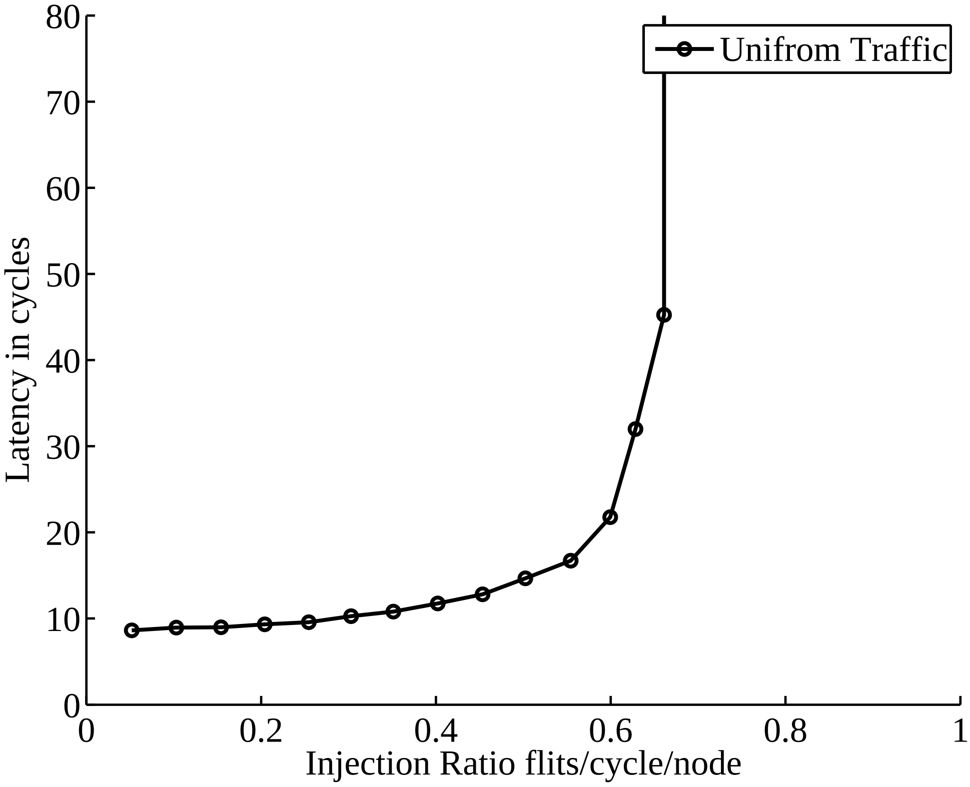


Figure 6.2: Load-Latency curves under different traffic patterns

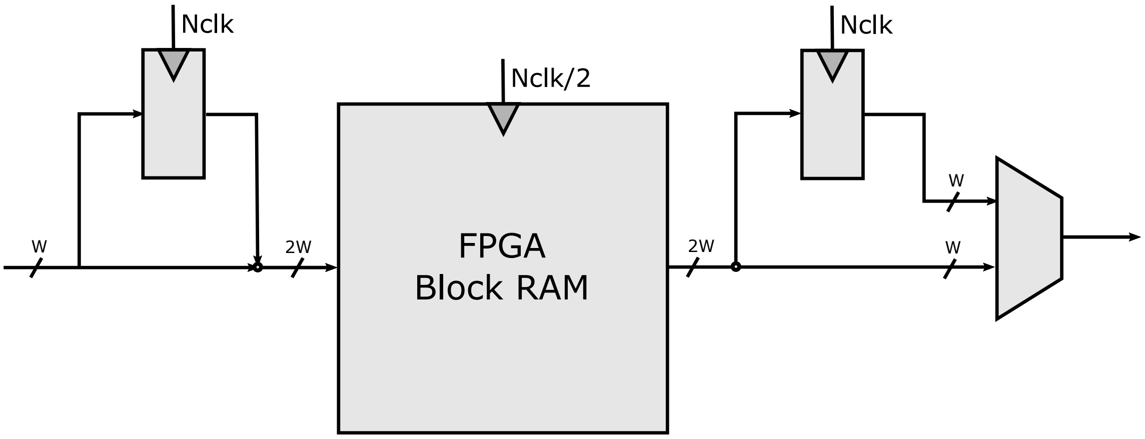


Figure 6.3: Doubling the datapath between the router and block RAM

Then, the router with external input buffers is implemented which will be used in the proposed hard NoC. As FPGA's block RAMs will be used in this NoC, we must assert that they will not be the delay bottleneck of the NoC. From the Virtex-5 (a 65 nm FPGA) data sheet, the block RAM clock to output delay is 0.61 ns only which will not affect the maximum operating frequency of the router [43].

In case the link between the router and the block RAM limits the maximum operating frequency of the router, this can be solved by doubling the datapath between the router and block RAM, as shown in Fig. 6.3. Instead of storing the incoming flit directly in the memory, the flit is temporally stored in a register until another flit from the same packet arrives, and then they are stored together in the same memory location. The same procedure is done when reading from the memory. In this way, the Block RAM can work at half the speed of the router. This architecture only adds one cycle latency.

To be sure that the hard links between routers will not limit the frequency, a study has been done in [18] which shows that a hard link can traverse more than one-third of the largest 65 nm FPGA's dimension at frequency of about 1 GHz. As we have four routers in each dimension, our proposed router can work at its maximum frequency which is equal to 1 GHz. The dynamic power consumed in the router (excluding the Block RAMs) is only 5.4 mW.

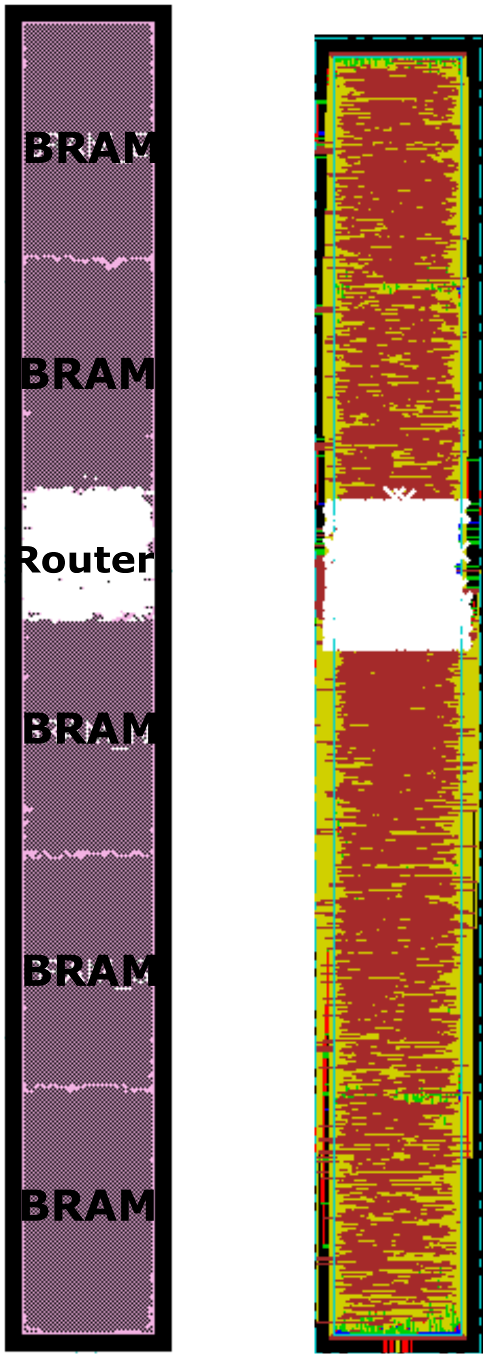
The router hard area (router area without the associated block RAMs) has reduced dramatically to reach 9513 μm^2. As our target is to reduce the cost of embedding hard NoC in FPGA and to reduce the wasted area when the NoC is not in use, this result is the most important achievement in our work. To show the significance of this result, we compare it with the area of the different components of Stratix III, a 65 nm ASIC technology FPGA, which is reported in [41]. The area of one LAB (FPGA cluster) is 22100 μm^2 which is more than double the area of our proposed hard router.

After synthesizing our design using Design Compiler, Cadence Encounter is used to complete the ASIC flow until the layout generation. First, the floorplan is specified as a column so it can be attached to an FPGA fabric layout. The Block RAMs are arranged in the column, and the router and the NI are placed among them, as shown in Fig. 6.4a. Moreover, the programmable multiplexers are added which enable the usage of block RAMs as conventional memory (for the FPGA fabric) or as the router's input buffers. After specifying the floorplan, the power rings and strips are added to connect the VDD and GND to the standard cells. Then, the standard cell are placed according to the specified floorplan. After that, timing optimizations and clock tree synthesis are performed. The final step is routing the design.

The final layout is shown in Fig. 6.4b in which the highlighted area is the layout of both the router and the NI. This layout acts only as a proof of concept because the layout of Block RAMs is not accurate. The synthesis process carried by the Design compiler maps everything into standard cells. Hence, instead of implementing the Block RAMs in the conventional compact way as array of SRAM cells, Design compiler maps each bit in the buffer memory into D flip-flop. For correct implementation, a Memory Compiler software should be used to generate the SRAM memory model and layout for the Block RAMs.

The proposed router is implemented also on Xilinx Virtex-5 FPGA xc5vlx50t-3ff1136 (Soft implementation) which is also a 65 nm ASIC technology FPGA. Xilinx ISE 14.6 software are used for synthesizing, mapping, placing and routing the design. It is used also for estimating the design's area, delay and power after place and route. The proposed router has a maximum operating frequency of 205 MHz and consumes a dynamic power of 137 mW. The design utilizes 929 LUTs, 603 registers and 5 Block RAMs.

These results show the significant advantages of using hard NoCs over soft NoCs. Both hard and soft implementation are done on a 65 nm technology. The hard router is ~5x faster than the soft router. The hard implementation consumes less than third the power of the soft implementation. There is no direct way to compare the area, however, some rough estimation can be done. The soft implementation uses 1029 BLEs (LUT-FF pair). There is no reported information about the area of Virtex-5 BLE, however, the reported area of Stratix III BLE can be used instead as both of them are implemented using the same technology. This neglects the difference in architecture between the two BLEs, however, it is acceptable as a rough estimation. The area of Stratix III BLE is 1100 μm^2 [6, 41], leading to a soft router area of ~1 mm^2 (excluding the block RAMs). Consequently, the hard router is ~100x smaller than the soft router.



(a) Floorplan **(b) Layout**

Figure 6.4: Floorplan and layout of the proposed NoC

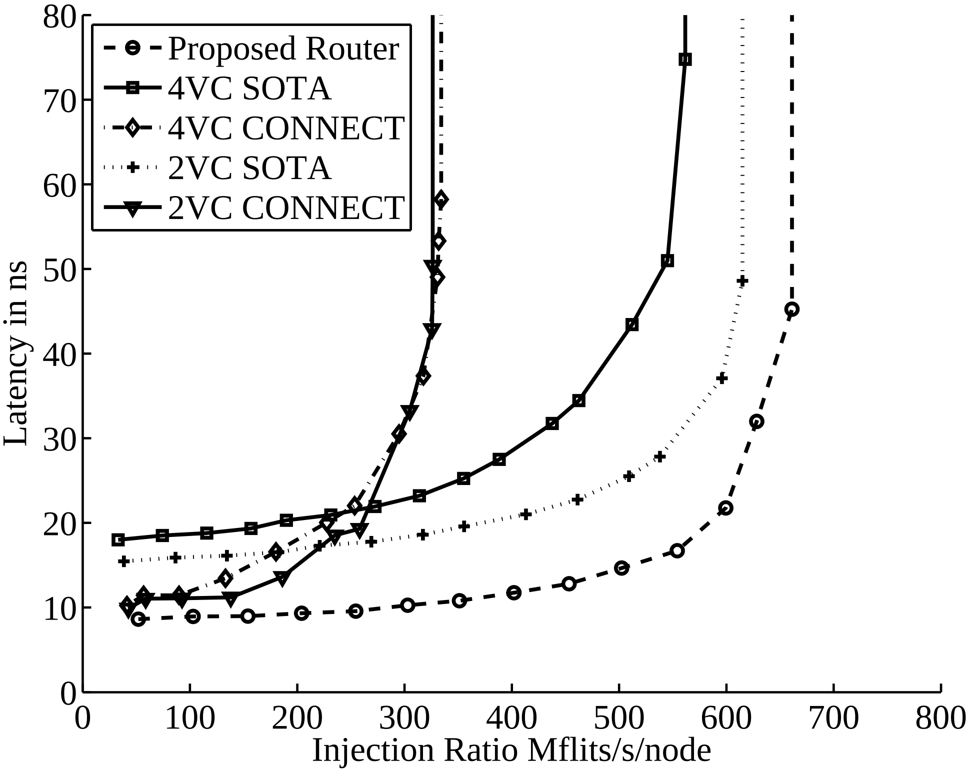


Figure 6.5: Load-Latency curves under uniform traffic

6.3 Comparison

In this section, the previous results of the router is compared to other routers presented in literature. The used routers in the comparison are SOTA and CONNECT, discussed in Chapter 3, as both of them are configurable parameterized routers. To achieve a fair comparison, SOTA and CONNECT are configured using the same parameters of the proposed router. The flit width is 32-bit, the packet length is 4-flit, the buffer depth is 8-flit/VC, and the topology is Mesh with 16 nodes.

As most of input buffers of our VOQ router have four VCs, a 4-VC version of SOTA and CONNECT is used in the comparison. A 2-VC version of SOTA and CONNECT is also used because it is the optimum number of virtual channels in a 16-node NoC as shown in chapter 3. The same simulation setup are used for all the participating routers, and all of them are implemented on UMC's 65 nm ASIC technology.

To compare the overall NoC performance, a load-latency curve scaled with the maximum operating frequency of each router is computed for each NoC. The load-latency curves are shown in Fig. 6.5 under uniform traffic. The proposed router has the highest throughput. Although CONNECT is a single stage router which leads to a low network latency, measured in cycles, the proposed router has the lowest latency, measured in ns, due to CONNECT's low operating frequency.

	This work	SOTA 4VC	SOTA 2VC	CONN. 4VC	CONN. 2VC
Max. Freq. (MHz)	1000	826	961	543	561
Area (μm^2)	67 k	124 k	62 k	96 k	50 k
PDP (pJ)	39	60	31	59	30

Table 6.1: A comparison between this work, SOTA and CONNECT in terms of maximum frequency, area and power-delay product

Table6.1 compares maximum frequency, area and power-delay product (PDP) of the routers. The proposed router has the lowest delay and a comparable power. The area of the proposed router is significantly lower than 2-VC routers despite they have nearly the same number of buffers. The proposed router has slightly higher area than 2-VC routers which have nearly half the number of buffers. This shows that the area of the router without the buffers (which we refer to as the hard area) is significantly lower than other routers. CONNECT consumes small area but it has a very low operating frequency on silicon (it is designed for FPGAs not ASIC) which results in a very bad NoC performance as shown in 6.5.

Then, we compare the proposed embedded hard NoC with the another proposal for embedding hard NoCs in FPGA presented by researchers in University of Toronto [18]. The used NoC in the Toronto proposal is a Mesh NoC with 64 nodes. It utilizes the 2-VC SOTA router with the following parameters: a flit width of 32-bit, a packet length of 4-flit, and a buffer depth of 5-flit/VC. It is implemented also using a 65 nm ASIC technology. The Toronto work reports only the implementation results and does not show the network performance of the NoC. Thus, to fully compare our work with Toronto work, we regenerate the used NoC and measure its network performance.

In Toronto work, they inflated the Design Compiler's reported area by 66.7% to account for the area used for wiring, inserted buffers and whitespaces (filled by fillers) in the placement and routing process. Thus, for a fair area comparison, we did the same to the reported area of our router. The total area of the NoC consists of hard routers, network interfaces, and hard and soft input/outputs. We also use the same methodology of Toronto work in computing the area of input/outputs. For hard inputs/outputs connecting between the routers, they used a CMOS driver of 13 $\mu m^2$to drive each hard link [18]. For soft input/outputs connecting between the network interface and FPGA fabric, they used an interconnect multiplexer of 120 μm^2 at each input/output [18].

In this comparison, we will compute two values for the area: a total area and a hard area. The total area comprises the area of routers including their shareable input buffers, network interfaces, CMOS drivers and interconnect multiplexers. While the hard area represents the total silicon area exclusively used by the hard NoC (the added silicon area to the FPGA to enable NoC). It will be the same as the total area excluding the input buffers (FPGA block RAMs) and the interconnect multiplexers. The interconnect multiplexers are

	This work	[18]
Total Area (mm^2)	2.2	11.3
Hard Area (mm^2)	0.32	11.3
Max. Freq. (MHz)	1000	943
Total PDP (nJ)	0.6	2.8
Total Throughput (Gflits/s)	10.5	6
Zero-load latency (ns)	8.6	21.8

Table 6.2: A comparison between this work and [18] in terms of area, maximum frequency, power-delay product, total throughput and zero-load latency.

excluded because, as we discussed in chapter 5, we will utilize the existing interconnect multiplexers connecting the FPGA fabric to the block RAM, so we will need only a simple 2:1 multiplexer of 3.6 μm^2 at each input/output for selecting between the block RAM and the network interface. Thus, the total area and hard area are given by equation 6.1 and equation 6.2 respectively.

$$\begin{aligned} TotalArea &= (RouterAreawithbuffers + NIArea) * 1.667 * 16 \\ &+(No.of hardIO * 13\mu m^2 + No.of NIIO * 120\mu m^2) * 16 \end{aligned} \quad (6.1)$$

$$\begin{aligned} HardArea &= (RouterAreawithoutbuffers + NIArea) * 1.667 * 16 \\ &+(No.of hardIO * 13\mu m^2 + No.of NIIO * 3.6\mu m^2) * 16 \end{aligned} \quad (6.2)$$

Table 6.2 shows a detailed comparison between the two NoCs in terms of total area , hard area, maximum frequency, total PDP (including the power of the shareable buffers) , total NoC throughput (NoC throughput times the number of nodes) and zero-load latency. The bad performance of the NoC used by the Toronto proposal is due to its large number of nodes which is not the optimum number of nodes, as we have proved in our work in Chapter 3.

In their most recent work, published in 2016, they modified the NoC parameters in order to be suitable for 65 nm FPGA [19]. The number of nodes was chosen to be 16 only. Moreover, they used a flit width of 128-bit instead of 32-bit, in order to transport the full bandwidth of DDR3, i.e. an FPGA I/O interface that has the highest bandwidth, on a single NoC link. The implementation results of this configuration is available on their FPGA-NoC Designer online tool [44]. To achieve a fair comparison, we modified the flit width of our proposed NoC to the same value. The detailed comparison is shown in Table 6.3. (N.B. At a flit width of 128-bit, the conventional FPGA block RAMs should be modified to support this data width)

These results show that the optimizations done in chapter 4 have significantly minimized the resources needed for implementing the router without affecting the NoC performance. Moreover, the proposed idea of sharing the block RAMs has dramatically reduced

	This work (128-bit)	[19]
Total Area (mm^2)	8.1	8.5
Hard Area (mm^2)	0.72	8.5
Max. Freq. (MHz)	990	917
Total PDP (nJ)	2.3	2.5
Total Throughput (Gflits/s)	10.4	8
Zero-load latency (ns)	8.6	15.9

Table 6.3: A comparison between this work and [19] in terms of area, maximum frequency, power-delay product, total throughput and zero-load latency.

the cost of embedding the hard NoC withing the FPGA. The proposed NoC has a total throughput of 330 Gbps or 1330 Gbps (at a flit width of 32-bit or 128-bit respectively). This NoC can be embedded inside the FPGA by adding a small silicon area of 0.32 mm^2 or 0.72 mm^2 to the conventional FPGA. It is worth noting that the total core area of 65-nm Stratix III FPGA is 412 mm^2 [41].

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis proposed an efficient FPGA-based hard NoC. First, we investigated various NoC design parameters using two NoCs, an ASIC-oriented NoC and an FPGA-oriented NoC. We also provided simulation results and design recommendations that can be used by NoC designers to select NoC parameters which give the optimum NoC performance for the targeted applications. The simulation results show that a mesh network with 16 nodes is the optimum configuration for a NoC especially on an FPGA platform. The optimum buffer depth is equal to two whole packets.

Secondly, different router architectures were studied to select the best-fit architecture for the hard NoC. A VOQ architecture was found to be a suitable choice as it provides good network performance while using minimum resources. A detailed design of each of the internal components of the router (input module, switch allocator, crossbar and output module) was performed. Then, each of these components was optimized according to the selected design parameters.

Moreover, we designed a network interface that connects the soft modules to the proposed NoC. We proposed that the FPGA's block RAMs should be shared between the NoC and the FPGA fabric. This sharing dramatically decreases the wasted area when the NoC is not in use. We also proposed a novel way for placing the hard NoC's routers inside the FPGA. Placement of routers in the same column of block RAMs would preserve the homogeneity of the FPGA and minimize the consumed area.

Finally, the hard NoC was implemented on a 65 nm ASIC technology and was compared to other NoCs presented in literature. The proposed NoC outperforms other previous hard NoCs in terms of area, maximum frequency, power-delay product, total throughput and zero-load latency. The proposed hard NoC provides FPGAs with high performance communications structure that is capable of transporting a traffic of 10 Gflits/s which is equivalent to 330 Gbps (at a 32-bit flit width) or 1.3 Tbps (at a 128-bit flit width) on 65 nm technology at a negligible cost of 0.32/0.72 mm^2 which is equivalent to 16/32 logic clusters (A 65nm Stratix III FPGA contains up to 13500 clusters).

7.2 Future Work

- Using VTR tool to model the FPGA with the hard NoC and evaluating it against different applications.
- Completing the whole layout of the FPGA, and replacing the standard-cell implementation of block RAMs with a full-custom implementation.
- Using the hard NoC also as the FPGA's configuration network to simplify parallel compilation and partial dynamic reconfiguration.

References

[1] 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, Aug. 2005.

[2] W. J. Dally and B. Towles, “Route packets, not wires: On-chip interconnection networks,” *Design Automation Conference (DAC)*, pp. 684–689, 2001.

[3] P. H. W. Leong, “Recent trends in FPGA architectures and applications,” *IEEE International Symposium on Electronic Design, Test and Applications*, pp. 137–141, Jan. 2008.

[4] C. Chiasson and V. Betz, “Should FPGAs abandon the pass-gate?” *International Conference on Field programmable Logic and Applications (FPL)*, pp. 1–8, Sep. 2013.

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

[6] M. S. Abdelfattah and V. Betz, “Design tradeoffs for hard and soft FPGA-based networks-on-chip,” *International Conference on Field-Programmable Technology (FPT)*, pp. 95–103, Dec. 2012.

[7] R. Hecht, S. Kubisch, A. Herrholtz, and D. Timmermann, “Dynamic reconfiguration with hardwired networks-on-chip on future FPGAs,” *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 527–530, Aug. 2005.

[8] M. K. Papamichael and J. C. Hoe, “CONNECT : Re-examining conventional wisdom for designing NoCs in the context of FPGAs,” *International Symposium on Field-Programmable Gate Arrays (FPGA)*, pp. 37–46, 2012.

[9] C. Hilton and B. Nelson, “Pnoc: A flexible circuit-switched NoC for FPGA-based systems,” *IEEE Proceedings of Computers and Digital Techniques*, vol. 153, no. 3, pp. 181–188, May 2006.

[10] Y. Huan and A. DeHon, “FPGA optimized packet-switched NoC using split and merge primitives,” *International Conference on Field-Programmable Technology (FPT)*, pp. 47–52, 2012.

[11] K. Goossens, M. Bennebroek, J. Y. Hur, and M. A. Wahlah, “Hardwired networks-on-chip in FPGAs to unify functional and configuration interconnects,” *ACM/IEEE International Symposium on Networks-on-Chip*, pp. 45–54, Apr. 2008.

[12] R. Francis and S. Moore, “Exploring hard and soft networks-on-chip for FPGAs,” *International Conference on ICECE Technology*, pp. 261–264, Dec. 2008.

[13] R. Pau and N. Manjikian, "Implementation of a configurable router for embedded network-on-chip support in FPGAs," *Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA)*, pp. 25–28, 2008.

[14] M. E. S. Elrabaa and A. Bouhraoua, "A hardwired NoC infrastructure for embedded systems on FPGAs," *Microprocessors and Microsystems*, vol. 35, no. 2, pp. 200–216, Mar. 2011.

[15] M. S. Abdelfattah and V. Betz, "The case for embedded networks on chip on field-programmable gate arrays," *IEEE Micro*, vol. 34, no. 1, pp. 80–89, Jan. 2014.

[16] M. S. Abdelfattah, A. Bitar, and V. Betz, "Take the highway: Design for embedded NoCs on FPGAs," *International Symposium on Field-Programmable Gate Arrays (FPGA)*, pp. 98–107, 2015.

[17] M. S. Abdelfattah and V. Betz, "The power of communication: Energy-efficient nocs for fpgas," *International Conference on Field programmable Logic and Applications (FPL)*, pp. 1–8, Sep. 2013.

[18] M. S. Abdelfattah and V. Betz, "Networks-on-chip for FPGAs: Hard, soft or mixed?" *ACM Transactions on Reconfigurable Technology and Systems*, vol. 7, no. 3, 20:1–20:22, Sep. 2014.

[19] M. S. Abdelfattah and V. Betz, "Power analysis of embedded NoCs on FPGAs and comparison with custom buses," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 24, no. 1, pp. 165–177, Jan. 2016.

[20] V. Betz and J. Rose, "Cluster-based logic blocks for FPGAs: Area-efficiency vs. input sharing and size," *IEEE Custom Integrated Circuits Conference*, pp. 551–554, May 1997.

[21] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 277–282, Mar. 1991.

[22] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture: Survey and challenges," *Foundations and Trends® in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2008.

[23] R. Olay, *Accelerating time-to-market using an FPGA and customizable SoC methodology*, RTC Magazine, Aug. 2012, [Accessed November 2016]. [Online]. Available: http://rtcmagazine.com/articles/view/102721.

[24] R. A. Rutenbar, M. Baron, T. Daniel, R. Jayaraman, Z. Or-Bach, J. Rose, and C. Sechen, "(when) will FPGAs kill ASICs?" *Design Automation Conference (DAC)*, pp. 321–322, 2001.

[25] Z. Or-Bach, *FPGAs as ASIC alternatives: Past and future*, EE Times, Apr. 2014, [Accessed November 2016]. [Online]. Available: http://www.eetimes.com/author.asp?doc_id=1322021.

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

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

[28] *Stanford Concurrent VLSI Architecture Group, open source network-on-chip router RTL*, [Accessed: May 2016]. [Online]. Available: https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router.

[29] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal network-on-chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, Sep. 2005.

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

[31] É. Cota, A. de Morais Amory, and M. S. Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-chip*. Springer Science & Business Media, 2011.

[32] *CONfigurable NEtwork Creation Tool (CONNECT)*, [Accessed: May 2016]. [Online]. Available: http://users.ece.cmu.edu/~mpapamic/connect/.

[33] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," *IEEE/ACM International Symposium on Microarchitecture*, pp. 172–182, 2007.

[34] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis, "Evaluating bufferless flow control for on-chip networks," *ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, May 2010.

[35] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," *International Symposium on Computer Architecture*, pp. 196–207, 2009.

[36] C. Fallin, C. Craik, and O. Mutlu, "Chipper: A low-complexity bufferless deflection router," *IEEE International Symposium on High Performance Computer Architecture*, pp. 144–155, 2011.

[37] Y. Tamir and G. L. Frazier, "High-performance multi-queue buffers for VLSI communications switches," *IEEE International Symposium on Computer Architecture*, pp. 343–354, 1988.

[38] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, Apr. 1999.

[39] K. Helal, S. Attia, T. Ismail, and H. Mostafa, "Priority-select arbiter: An efficient round-robin arbiter," *IEEE International Conference on NEW Circuits and Systems (NEWCAS)*, Jun. 2015.

[40] K.-L. Tsai, H.-T. Chen, and Y.-A. Lin, “Power and area efficiency NoC router design for application-specific SoC by using buffer merging and resource sharing,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 19, no. 4, 36:1–36:21, Aug. 2014.

[41] H. Wong, V. Betz, and J. Rose, “Comparing FPGA vs. custom CMOS and the impact on processor microarchitecture,” *International Symposium on Field Programmable Gate Arrays (FPGA)*, pp. 5–14, 2011.

[42] R. M. Francis, “Exploring networks-on-chip for FPGAs,” PhD thesis, University of Cambridge, 2009.

[43] *Virtex-5 FPGA data sheet: DC and switching characteristics*, DS202, v5.5, Xilinx, Inc., Jun. 2016. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds202.pdf.

[44] M. S. Abdelfattah, *FPGA-NoC designer*, [Accessed November 2016]. [Online]. Available: http://www.eecg.toronto.edu/~mohamed/noc_designer/.

ملخص الرسالة

لقد أصبحت مصفوفات البوابات المنطقية القابلة للبرمجة منصة مفضلة لنظم الرقائق الإلكترونية. و مع زيادة حجم نظم الرقائق الإلكترونية، برزت شبكات الرقائق الإلكترونية كحل واعد لمشاكل الاتصال بين وحداتها. و بناء على ذلك، ازدادت أهمية شبكات الرقائق الإلكترونية الخاصة بمصفوفات البوابات المنطقية القابلة للبرمجة، ليس فقط لحل مشاكل الاتصال بين وحدات نظم الرقائق الإلكترونية و لكن ايضا كحل لبطء روابط مصفوفات البوابات المنطقية القابلة للبرمجة و لتسهيل اعادة تشكيل المصفوفة بصورة جزئية. شبكات الرقائق الإلكترونية الثابتة لديها أداء أفضل من شبكات الرقائق الإلكترونية القابلة للبرمجة كما انها تستهلك طاقة و مساحة أقل، و لكنها غير قابلة للتشكيل و تتسبب في هدر للمساحة عندما تكون الشبكة غير مستغلة مما جعل تصميمها أكثر صعوبة.

في هذه الرسالة، يتم تقييم عوامل التصميم المختلفة لشبكات الرقائق الإلكترونية من أجل الوصول للعوامل المناسبة التي يمكن استخدامها في شبكات الرقائق الإلكترونية الثابتة. و كذلك نقيم تصاميم مختلفة لبنية الموجهات لاختيار البنية المثالية. يقلل الموجه الذي تم تصميمه المساحة المهدرة عن طريق استخدام موارد قليلة و قابلة للمشاركة. علاوة على ذلك، نقترح في هذة الرسالة طريقة مثالية حديثة لإدراج شبكات الرقائق الإلكترونية الثابتة بداخل مصفوفات البوابات المنطقية القابلة للبرمجة. تمد الشبكة المقترحة مصفوفات البوابات المنطقية ببنية تحتية للاتصالات ذات أداء عال بتكلفة زهيدة.

مهندس: سامح عطيه احمد عطيه
تاريخ الميلاد: ١٩٩١/١٠/٠١
الجنسية: مصري
تاريخ التسجيل: ٢٠١٣/١٠/٠١
تاريخ المنح: ٢٠١٦/.../.....
الدرجة: ماجستير العلوم
القسم: هندسة الإلكترونيات و الاتصالات الكهربية

المشرفون:

أ.د. حسام على حسن فهمي
د. حسن مصطفى

الممتحنون:

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

عنوان الرسالة:

تحسين شبكات الرقائق الإلكترونية الثابتة الخاصة بمصفوفات
البوابات المنطقية القابلة للبرمجة

الكلمات الدالة:
شبكات الرقائق الإلكترونية، مصفوفات البوابات المنطقية القابلة للبرمجة، الموجه، ذاكرة الوصول العشوائى

ملخص الرسالة:

في هذه الرسالة، يتم تقييم عوامل التصميم المختلفة لشبكات الرقائق الإلكترونية من أجل الوصول للعوامل المناسبة التي يمكن استخدامها في شبكات الرقائق الإلكترونية الثابتة. و كذلك نقيم تصاميم مختلفة لبنية الموجهات لاختيار البنية المثالية. يقلل الموجه الذي تم تصميمه المساحة المهدرة عن طريق استخدام موارد قليلة و قابلة للمشاركة. علاوة على ذلك، نقترح في هذة الرسالة طريقة مثالية حديثة لإدراج شبكات الرقائق الإلكترونية الثابتة بداخل مصفوفات البوابات المنطقية القابلة للبرمجة. تمد الشبكة المقترحة مصفوفات البوابات المنطقية ببنية تحتية للاتصالات ذات أداء عال بتكلفة زهيدة.

تحسين شبكات الرقائق الإلكترونية الثابتة الخاصة بمصفوفات البوابات المنطقية القابلة للبرمجة

اعداد

سامح عطيه احمد عطيه

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

يعتمد من لجنة الممتحنين:

أ.د. حسام على حسن فهمي – المشرف الرئيسي

أ.د. أمين محمد نصار – الممتحن الداخلي

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

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٦

تحسين شبكات الرقائق الإلكترونية الثابتة الخاصة بمصفوفات البوابات المنطقية القابلة للبرمجة

اعداد

سامح عطيه احمد عطيه

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

تحت إشراف

أ.د. حسام على حسن فهمي	د. حسن مصطفى
أستاذ	مدرس
قسم هندسة الإلكترونيات و الاتصالات الكهربية	قسم هندسة الالكترونيات و الاتصالات الكهربية
كلية الهندسة – جامعة القاهرة	كلية الهندسة – جامعة القاهرة

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٦

تحسين شبكات الرقائق الإلكترونية الثابتة الخاصة بمصفوفات البوابات المنطقية القابلة للبرمجة

اعداد

سامح عطيه احمد عطيه

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

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٦