2009

Design and Evaluation of a Parameterizable NoC Router for FPGAs

Michael Brugge
University of Windsor

Design & Evaluation of a Parameterizable NoC Router for FPGAs

By

Mike Brugge

A Thesis
Submitted to the Faculty of Graduate Studies
Through Electrical and Computer Engineering
In Partial Fulfillment of the Requirements for the
Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada
2009

Design & Evaluation of a Parameterizable NoC Router for FPGAs

By

Mike Brugge

APPROVED BY:

_____

N. Zamani
Mechanical, Automotive, and Materials Engineering

_____

K. Tepe
Electrical and Computer Engineering

_____

M. A. S Khalid, Advisor
Electrical and Computer Engineering

_____

Mitra Mirhassani, Chair of Defense
Electrical and Computer Engineering

September 21, 2009

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

The Network-on-Chip (NoC) approach for designing (System-on-Chip) SoCs is currently emerging as an advanced concept for overcoming the scalability and efficiency problems of traditional on-chip interconnection schemes. This thesis addresses the design and evaluation of a parameterizable NoC router for FPGAs. The importance of low area overhead for NoC components is crucial in FPGAs, which have fixed logic and routing resources. We achieve a low area router design through optimizations in switching fabric and dual purpose buffer/connection signals. We propose a component library to increase re-use and allow tailoring of parameters for application specific NoCs of various sizes. A set of experiments were conducted to explore the design space of the proposed NoC router using different values of key router parameters: channel width (flit size), arbitration scheme and IP-core-to-router mapping strategy. Area and latency results from the experiments are presented and analyzed.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Mohammed A. S. Khalid, where my research would not be possible without his advice and wisdom that guided me over the course of this research. I first met Dr. Khalid during the third year of my undergraduate degree, when he taught a class in digital system design. He stimulated my interest in this field. Although this class was based on processor design, he also provided an introduction into VHDL and embedded system design. The next year, I was able to get into his embedded system design class where I developed my skills and received excellent experience in the exciting field of reconfigurable computing. For this very reason, it had motivated me to continue my academic career in this field. My appreciation also goes out to my thesis committee members, Dr. K. Tepe and Dr. N. Zamani, for their time to sit on my committee and for reviewing my thesis.

I want to thank my family for all their constant support and encouragement. Thanks to my parents for their understanding throughout. Thanks for helping to keep me focused day after day. Thanks to my brother Shawn for providing some less stressful activities such as movies and Spitfire games. Thanks to my girlfriend Amy for always listening and helping me work through any and all problems.

Finally, I need to acknowledge my friends and fellow graduate students at the University of Windsor. Pat, Steven, and Chris thank you for your friendship and guidance. Thanks to Matt for your company and making those long days at the office a little less stressful. You could always provide me with the help I needed. Lastly, thanks to the rest of my colleagues; to Thuan, Lin Lin, Omar, and everyone else who made this great milestone in my life so enjoyable.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
ASIC	Application Specific Integrated Chip
BRAM	Block Random Access Memory
CAD	Computer Aided Design
CPU	Central Processing Unit
CS	Circuit Switched
DSP	Digital Signal Processing
deMUX	De-Multiplexer
FIFO	First In / First Out
Flit	Flow Control Unit
FPGA	Field Programmable Gate Array
FSM	Finite State machine
HLP	High Level Protocol
HOL	Head-Of-Line
I/O	Input / Output
IC	Integrated Circuit
IOB	I/O Block
IP	Intellectual Property
LE	Logic Element
LUT	Lookup Table
MLPR	Multi-Local Port Router
MUX	Multiplexer

NEP	Signal (Non-Existent Port)
NoC	Network-on-Chip
NRE	Non-Recurring Engineering
PS	Packet Switched
QoS	Quality of Service
RRA	Round Robin Arbiter
SAF	Store and Forward
SoC	System-on-Chip
VCT	Virtual Cut-Through
VHDL	Very high speed integrated circuit Hardware Description Language
WH	Wormhole

Chapter 1

Introduction

The complexity of a system on silicon is comparable to other macro systems such as space shuttle or skyscrapers, when measured in terms of the number of basic elements intricately connected together, but at a micro level [22]. Moore's law describes an important trend in the history of the integrated circuit (IC): the number of transistors that can be placed on an IC is increasing exponentially, doubling approximately every two years. This trend has continued for more than half a century. Increasing transistor density, higher operating frequencies, shorter time-to-market and reduced product life cycle, characterize today's semiconductor industry. As semiconductor technology evolves, electronic industries continually push the envelope for greater functional and performance capabilities in new electronic systems. This is creating a continuing need for new design methodologies and design space exploration.

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. Embedded systems range from portable devices such as digital watches, cameras and MP3 players, to large stationary units like traffic lights and factory controllers. Complexity varies from low, with a single micro-controller chip, to very high with multiple intellectual property (IP) cores and peripherals. The exponential growth in chip density is opening the door for the implementation of even larger and more complex systems, where complete embedded systems can be built onto a single chip. This paradigm shift is known as System-on-Chip (SoC) and is becoming increasingly common and complex. SoCs may contain many

hardware and/or software blocks, such as processors, DSPs, memories, peripheral controllers, gateways, and other custom logic blocks.

The communication architecture implemented in SoCs is an important contribution to the overall performance. Since the introduction of SoC concept, designers relied on a custom-designed ad-hoc mixture of buses and dedicated wires as communication mechanisms. Dedicated wires are effective for systems with a small number of cores, but available routing resources are quickly used up as system complexity grows. They also provide poor reusability and flexibility. A shared bus is a set of wires common to multiple cores, which increases both reusability and scalability. This scheme works well for Master-Slave communication patterns, where peripherals (slaves) wait for data to be received or requested from a more complex processing IP core (master). However, when there are several masters in the system, contention creates a bottleneck which gets worse as complexity grows. And although using hierarchical bus models separated by bridges may reduce some of these constraints, it also complicates protocols while failing to fully eliminate the scalability problem. Design and verification times also grow with SoC complexity [13].

With the current trend in integration of more complex SoCs, there is a need for better communication infrastructure on chip that will solve the scalability problem by supporting multiple concurrent connections between IP cores, allow for pre-tested design reuse to minimize design and verification times, all while maintaining a low area-overhead. Many architectural templates have been proposed for hardware platforms for future SoCs to provide standardized communication. NoC has been introduced as a new interconnection paradigm able to integrate IP cores in a structured and scalable way. This idea aims to allow system modules to communicate with each other over an on-chip network and has been gaining support world-wide. NoCs are based on the concepts adopted on the building of interconnection networks for parallel computers. Each router has a set of ports which are used to connect routers with its neighboring routers and with the IP cores of the system. This solution also promotes independent design of IP cores. NoC is still an active area of research, but many works [12], [13], [19], [22] have provided promising performance results over current communication strategies (dedicated wires, shared

and locked buses) for FPGAs. There is a great need for research in hardware implementation of NoC systems to determine the feasibility of implementing various parameters, and also to accurately determine what design tradeoffs are involved in NoC implementation.

ASICs are increasingly being replaced by Field Programmable Gate Arrays (FPGAs) for applications with low to medium volume, due to longer design cycles and high cost [14]. FPGA's have also continued to grow with the increase in chip density. Modern FPGA's have various hardware and/or software blocks embedded within them, such as DSP blocks, memory, and even processors. These blocks, along with customizable logic blocks, makes them the perfect candidate for NoC designs. A fundamental difference between ASICs and FPGAs is that wires in ASICs are designed such that they match the requirements of a particular design. Wire parameters such as length, width, layout and the number of wires can be varied to implement a desired circuit. Conversely, in an FPGA, area is fixed and routing resources exist whether or not they are used. The electrical characteristics of the FPGA are solved by the chip vendor, not by the user [3]. Exploiting the advantages of NoC in FPGAs for implementing SoC designs is an active area of research where the goal becomes implementing a circuit within the limits of available resources. Hence, the importance of designing a generic light-weight router whose area can be traded-off for performance in many different ways, to meet applications requirements.

This thesis is primarily concerned with the challenges of parameter selection for a NoC-based system. The emphasis is on the evaluation of NoC router parameters targeted for implementation on FPGAs, since FPGAs serve as an excellent platform for rapid prototyping and design space exploration. Recent research suggests the shift of larger SoC implementations on FPGAs as well as the design of light-weight, FPGA based NoC routers, prompting possible future NoC implementations.

1.1 Thesis Objectives

The main goal of this research was to evaluate NoC router parameters based on area and latency to allow designers to make informed choices for the creation of large embedded systems on FPGAs. This research has the following major objectives:

1. Investigate the feasibility of NoC router implementation on FPGAs.
2. Explore the effects of varying NoC router parameters on area and latency. To date, not much research has been done to address this issue.
3. Investigate and design benchmarks with features that would severely test the NoC router implementations.

For the first objective, an experimental framework was developed using VHDL, allowing synthesis in Altera Quartus II CAD tool design environment. A parameterizable NoC router was designed and tested. Literature survey was conducted that showed a lack of results on NoC router implementation for FPGAs. Parameters that were not explored in previous research were selected and design space exploration was conducted for different values of those parameters. To address the third objective, benchmarks for each parameter were developed based on application and random traffic patterns. Finally, the proposed router was experimentally evaluated, using different parameter values, based on metrics such as area, latency, throughput, FPGA on-chip memory utilization and FPGA routing resource utilization.

1.2 Thesis Organization

This thesis is primarily concerned with evaluating the trade-offs for area and latency for many NoC router parameters. Emphasis is placed on the design of NoC routers targeted for implementation on FPGAs. The outline of this thesis is as follows. Chapter 2 presents a background on NoC router design, FPGA architecture and provides a description of recent related academic research. An overview of the proposed router architecture is given in Chapter 3. In Chapter 4, we present experimental evaluation results for the proposed router used in a variety of mesh configurations. Chapter 5 concludes the paper with a summary and discussion of future work.

Chapter 2

Background and Previous Work

In this chapter, the background and previous work that is relevant to this research is presented. This chapter begins with an overview of Network-on-Chip (NoC) and NoC parameters. That is followed by a section discussing NoC evaluation metrics. Next a section describing FPGA technology is presented. The chapter concludes with a discussion of previous work closely related to NoC router design and evaluation.

2.1 Overview of NoC

There are many research papers and books dealing with micro-networks, with many subtle differences in definitions, concepts, and theories. In this section, for the sake of clarity, we present a collection of concise definitions of relevant concepts and theory that holds true for most NoC systems including our proposed router architecture. Emphasis is placed on how such concepts relate to FPGA implementations wherever necessary.

2.1.1 NoC Building Blocks

NoC aims to allow computational components (IP cores) to communicate over an on-chip network. An example of a NoC interconnection network is shown in Figure 2.1, which consists of four basic functional blocks. These blocks include the IP cores, the network adaptor, the routing node, and the links. IP cores are specific to the application and not considered part of the NoC design.

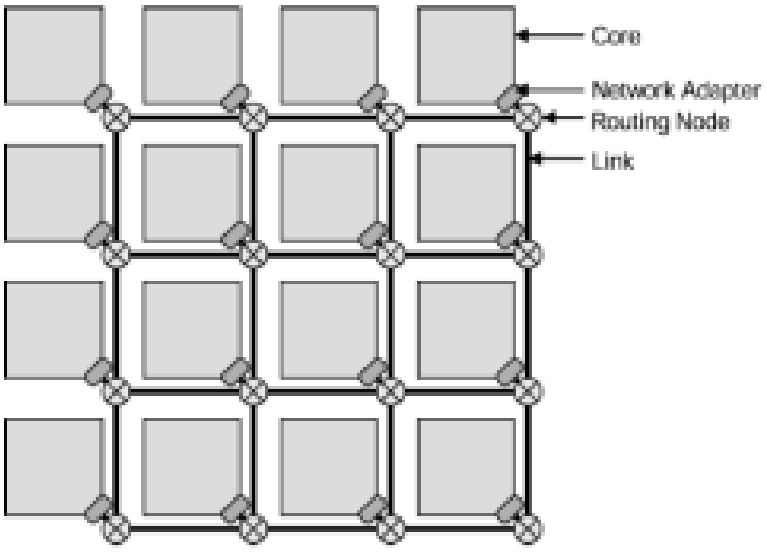


Figure 2.1: Illustration of Four Basic NoC Building Blocks [23]

2.1.1.1 Links

This component provides connections for a routing node with a network interface or another routing node. It may provide buffer resources and separate control lines for connection establishment and teardown.

2.1.1.2 Network Interface

This component provides the conversion between the high level protocol (HLP) that the IP uses and the packet-based communication protocol of the NoC. This component may be responsible for buffering packets, storing IP core addresses, creating and disassembling messages, implementing end-to-end flow control, crossing clock domains, and other higher level network issues.

2.1.1.3 Routing Node

This component carries out the task of receiving and forwarding messages inside the network based on NoC parameters. The Router is the central component in a NoC interconnection network. Therefore, its area and speed play a big role in the performance of the overall system. NoC interconnection networks have a large range of parameters

which are all focused around router design. Research in this area still lacks in useful implementation results.

2.1.2 NoC Parameters

Network parameters are an important research topic among NoC designers. To further enhance performance, the parameters of the NoC should be chosen based on the specific application. Therefore, the goal in a general network design is to leave as much designer flexibility as possible. Not every network parameter can be created flexible and many of the parameters are dependent on each other. Evaluation and testing can provide insight into how to select these parameters, although a better solution may be a flexible library of interchangeable components. We have chosen to create such a library using VHDL, and use an FPGA to provide fast prototyping for results. Due to time and resource constraints, limitations had to be set on the amount of design space explored. Network parameters can be broken into three groups as in [2]: Infrastructure, Communication Mechanism, and Mapping. Each of these groups will be discussed separately below.

2.1.2.1 Infrastructure

Infrastructure aims to determine the network architecture and includes channel width, topology, buffering and floor planning. These parameters are all application specific and should be left to the designer's discretion.

2.1.2.1.1 Channel Width

This parameter describes the size of the data passed between routers. It is important since it directly affects bandwidth but can lead to the side affects of increased area/power. Our library allows for a parameterizable channel width which will also be tested for resulting area and latency tradeoffs.

2.1.2.1.2 Topology

This parameter refers to the way routers are connected in the network. It should be chosen to minimize area, while maximizing utilization without causing bottlenecks.

Saldana et al. evaluate different topologies in terms of area and routing resources [3]. Figure 2.2 shows some popular NoC topologies. Ring and star achieve slightly better results, although both fail to provide solutions to the scalability problem. As the number of nodes increases, ring suffers large end to end delay and star suffers from a central bottleneck. Narasimhan et al. compare the performance of a two dimensional torus to mesh, showing a slight edge for two dimensional torus [4]. They however, do not compare the extra routing resources needed or the increase area of each router due to a more complex routing algorithm. We restrict the topology to mesh, which is most common among FPGA networks, but allow for various implementation sizes up to an 8 x 8 network. We also create multiple local ports (up to four per router), which allows for multiple IP cores connected to each router or multiple router connections for single IP cores. This increases the possible number of IP cores connected in the network from 64 to 256. With available FPGAs, it would be impractical to build anything larger due to area and routing resource constraints.

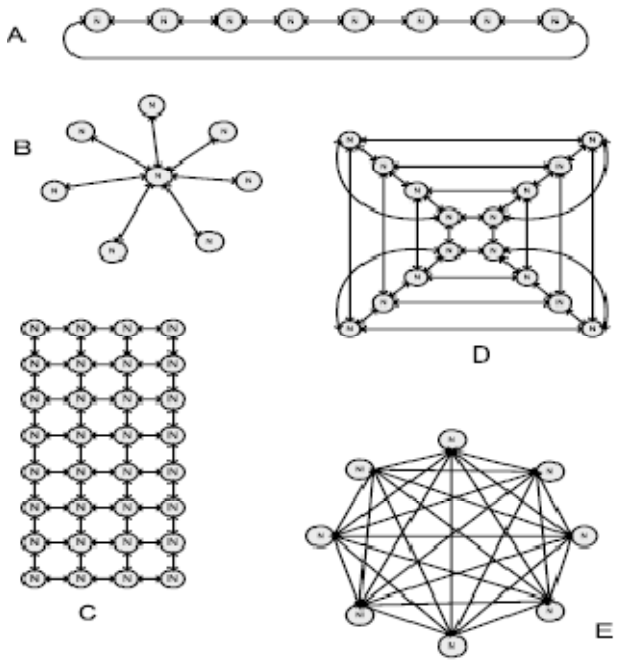


Figure 2.2: Popular NoC Topologies [3]

2.1.2.1.3 Buffering

This parameter defines the approach used to store messages while they cannot be scheduled. This has a serious impact on the area overhead of the network, however, it can also have a serious impact in reducing network latency. We use input and output buffering to prevent head-of-line blocking (HOL). This occurs when a packet or packets, experience blocking and cause the blocking of later packets which could otherwise be processed. The inclusion of an output buffer allows the blocked packet to move out of the input buffer, to unblock the later packets for processing. Buffer allocation should be based on traffic patterns. The authors of Hermes [8] design a generic router which has a parameterizable buffer depth. They also include insight through testing various buffer sizes for area and performance values.

2.1.2.1.4 Floor Planning

Floor planning involves the placement of network components. This is not important in FPGA-based NoC designs as it is done by vendor specific CAD tools (Altera Quartus II CAD tool).

2.1.2.2 Communication Mechanism

Communication mechanism deals with how data flows through the network and includes flow control, switching mode, switching mechanism, and routing algorithm. These parameters are usually set when designing the NoC platform.

2.1.2.2.1 Flow Control

This parameter deals with the allocation of channels and buffers to data as it travels from source to destination. The two extremes are packet switching (PS) and circuit switching (CS). In circuit switching, there is a dedicated connection between the two modules in which raw data can be transmitted freely. This technique requires a setup time to build and tear down connections, and its channel reservation nature often leads to idle times and causes unreliable blocking. The only upside to this method is its ability to provide

guaranteed bandwidth during connection times. This method does not scale as well and is not a popular choice for NoC systems. In packet switching, data is broken into packets which carry routing information. Packets can further be broken down into flow control units (flits). Modules can send packets at any time and there are often many different packets in flight at a given time. The routers must process and redirect each packet accordingly.

2.1.2.2.2 Switching Mode

This parameter only exists in PS networks and defines how packets move through the network. The most important schemes are store-and-forward (SAF), virtual cut-through (VCT), and wormhole (WH). In SAF, a router cannot forward a packet until all its flits have been received. Therefore, latency is proportional to packet size and it carries large buffer requirements. In WH, the first flit (header) determines the next hop and all remaining flits follow and can be sent as soon as it's received. Therefore, latency is proportional to flit size. This method combines packet switched flow control with circuit switched ideas but also leads to channel reservation. It also requires a complex routing algorithm. VCT uses a combination of both ideas to provide latency based on flit size without idle times by guaranteeing buffering before setting up the connection. However, this method uses large buffer amounts and very complex routing algorithms making it unsuitable for light-weight networks. We have chosen SAF for its light-weight algorithm and to prevent channel reservation. Future testing may extend flexibility to include WH as well.

2.1.2.2.3 Switching Mechanism

This parameter refers to how connections are made inside a router. Common architectures include fully connected, crossbar matrix, and partial crossbar matrix. We use a partial crossbar scheme to save area as it is the smallest configuration. We have also implemented optimizations based on the chosen routing algorithm which we will discuss later.

2.1.2.2.4 Routing Algorithm

The routing algorithm determines the path the packet will take. There is not much research guidance available on effectiveness of available routing algorithms for NoC implementations. We use XY routing for its simplicity and low area overhead. This scheme also prevents livelock and assures flits and packets arrive in order. Routing schemes can also require congestion control and recovery mechanisms, which can lead to added area overhead. We allow this to be handled by the application layer.

2.1.2.3 Mapping

Mapping determines how to integrate a given application to the NoC platform and includes scheduling and module mapping.

2.1.2.3.1 Scheduling

This is a traditional computer science topic but most work neglects inter-processor communication. Arbitration schemes consider priority of packets when making grants inside the routers among the network. Arbiter schemes can be static or dynamic. Dynamic arbitration makes a decision at run-time and is more flexible, however also requires a larger area. Dynamic Schemes can also prevent starvation which is a downfall of static schemes. Our library provides a few different components to allow for area and latency trade-offs.

2.1.2.3.2 Module Mapping

This parameter aims at selecting IP modules for different locations to minimize traffic. This parameter is application specific and is explored later.

2.1.3 NoC Evaluation Metrics

NoC architectures are designed to meet certain cost and performance constraints, which include, but are not limited to, speed, area, energy/power consumption, Quality of Service (QoS) and flexibility. Through parameter selection, one or more metrics can often be

improved at the cost of other(s). In the following sections we will discuss the evaluation metrics for NoC router architectures and their relevance to this thesis.

2.1.3.1 Latency/Throughput

When using FPGA technologies, evaluating speed can often be as easy as obtaining the synthesized maximum frequency the clock is capable of running at. For NoC routers, this is not the case. Although still important to the overall performance, NoC routers have multiple ports which can send, receive and process simultaneously. Therefore, it is important to observe data transaction times.

Speed can be measured in delay, which is referred to as latency. Latency can be the overall run time, it can be decomposed into several intervals such as packet or flit latency, calculated as an average, along with other creative possibilities. We use the overall application run time measured in cycles, which is converted to time as a function of the maximum clock frequency.

Speed can also be measured in bandwidth, which is referred to as throughput. Throughput is the amount data transferred over a period of time. Throughput can be; the ideal data processing rate (system working under the best possible conditions), it can be decomposed into several intervals such as overall application, packet or flit throughput, measured per system, IP core, router, or port, calculated as an average, along with other creative possibilities. We use the overall application/simulation throughput measured in packet and flits per cycle, which is converted to time as a function of the maximum clock frequency.

Finally, some papers suggest NoC router speed be measured in terms of bottlenecks. Either the number of occurring bottlenecks, or the time in which a router has a bottleneck occurring. This metric was not used in our experiments but is very interesting to note.

It is important to understand that speed characteristics for NoC routers are application specific and do not represent speed characteristics of the router alone. This makes comparing different router performances quite hard.

2.1.3.2 Area

In an FPGA, overall system area is limited and therefore important to keep minimal. Area can be measured as a number or a percent of available resources. Area is a very vague term. In an FPGA, there are many components which occupy area. For our experiments, we use area in terms of logic elements (LE's), memory blocks, and routing resources (direct wires, interconnects, and clocks). This Information is obtained from Altera Quartus II CAD tool after compiling and synthesizing the VHDL code. Altera Quartus II CAD tool gives the option to synthesize for the lowest area or highest speed.

2.1.3.3 Energy/Power Consumption

For FPGA technologies, power consumption is a metric not often evaluated. This is due to the fact that power consumption has a direct relation with area. Also, designing low power circuits for FPGA implementation is based on trial and error. Therefore, most research including ours focuses on area and excludes the use of power estimation tools.

2.1.3.4 Quality of Service (QoS)

Quality-of-Service (QoS) is a networking term that refers to guarantees that the system can make about its performance. In computer networks, certain application such as video streaming are required to give a guarantee of high uninterrupted bandwidth because of the uniqueness of the application. It is difficult to actually predict the behavioral nature of the data in the network, thus making it nearly impossible to guarantee the required bandwidth without some margin of error. PS suffers even more in its ability to predict the timing of its services. To help provide QoS, NoCs must provide service free of the following causes of failure:

1. *Livelock:* data is prevented from reaching its destination because it is in a cyclic path.
2. *Starvation:* data is prevented from reaching its destination because some resource does not grant access.

3. *Deadlock:* data is prevented from reaching its destination because it is blocked at some intermediate resource.

Livelock occurs when the packets are being routed around their destination and are placed in a cyclic holding manner. Livelock can be avoided by allowing the packet to travel the shortest route. XY routing avoids this situation.

Starvation is a common PS problem. It occurs when the packet is discriminated against as low-priority data, thus never getting service. This can be avoided by allocating resources to process all packets equally, automatically dropping and resending packets in the network for too long, or by use of dynamic arbitration insuring all ports receive service.

Deadlock is cause by packet being continuously blocked and it is the hardest problem to solve because packets that are blocked stay blocked while waiting for an event that cannot happen. This problem is solved by restricting channel reservation.

2.1.3.5 Flexibility

Flexibility refers to the number of manipulations the designer can make. Our router design allows that some of the parameters be changed at design time allowing the designer to choose trade-offs. Designs with a high degree of flexibility are the ones that allow more parameters to be changed. Other flexibility characteristics include scalability (ability to add more and more IP cores) and design re-uses (ability to use the same NoC architecture for multiple designs).

2.2 FPGA Technology

A field-programmable gate array (FPGA) is an integrated circuit (IC) which can be reprogrammed many times to implement any desired digital circuit which doesn't exceed the limits of the device. An FPGA contains a two dimensional array of programmable logic components, called logic elements (LEs), a hierarchy of wires and buses with reconfigurable interconnects that allow the LEs to be physically connected and is surrounded by configurable I/O blocks (IOB's). Figure 2.3 shows this two dimensional FPGA architecture. In addition, FPGAs typically include other specialized blocks, such

as block random access memories (BRAMs) and digital signal processors (DSPs) which still provide some degree of configurability. An FPGA is programmed by loading data bits in memory cells which control transistor switches to establish non-permanent connections. An FPGA can support hundreds of thousands of gates of logic operating at speeds of tens of megahertz.

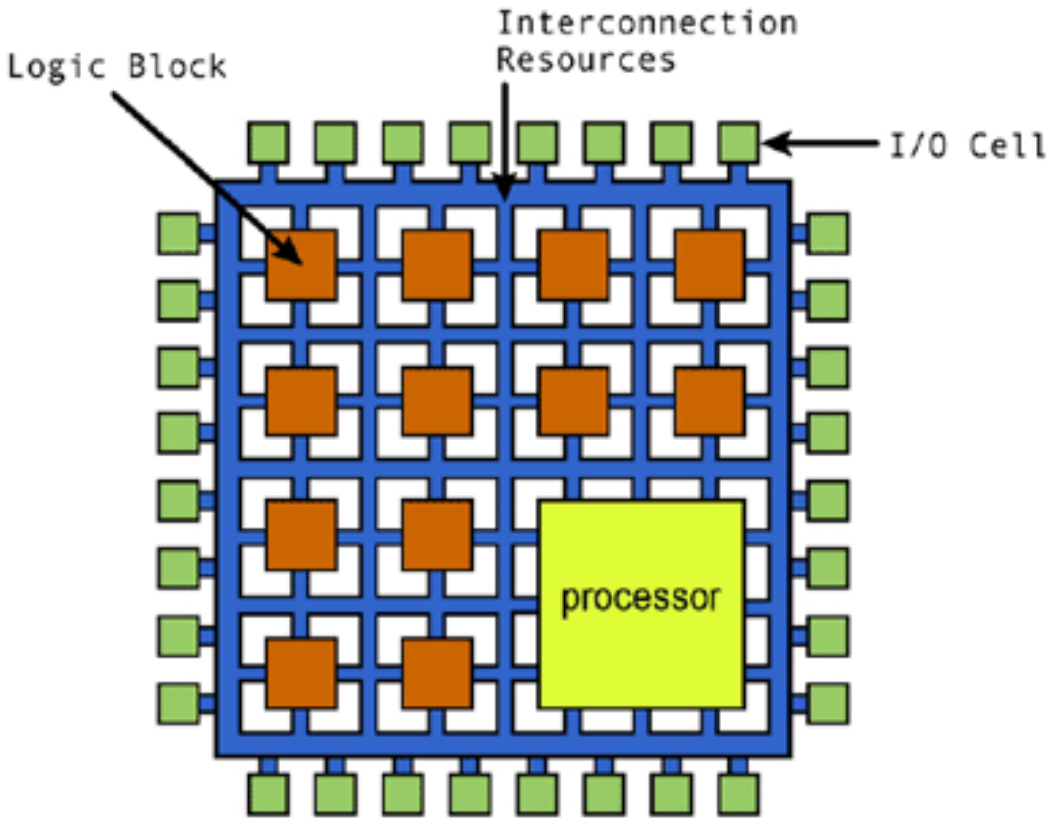


Figure 2.3: FPGA Architecture [25]

We can use an FPGA to implement any logical function that an application-specific integrated circuit (ASIC) could perform, however FPGAs tend to run slower and consume more area and energy compared to application-specific integrated circuits (ASICs). The use of FPGA to implement digital circuits is on the rise over ASIC designs. FPGA allows for faster prototyping, shorter time to market, and lower NRE costs compared to ASIC. The main advantage comes from the ability to fix bugs in the field through reprogramming.

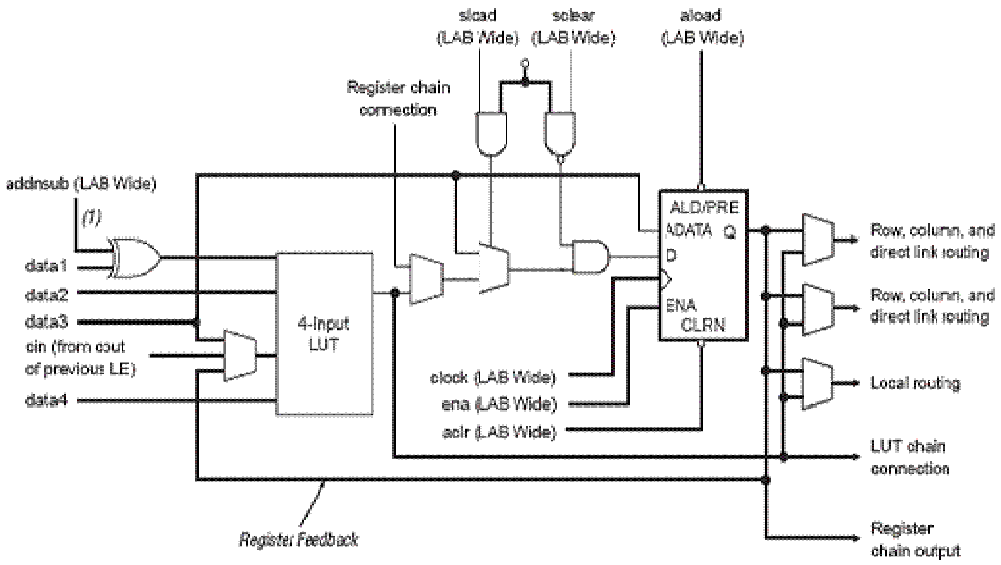


Figure 2.4: Altera Logic Element Architecture [26]

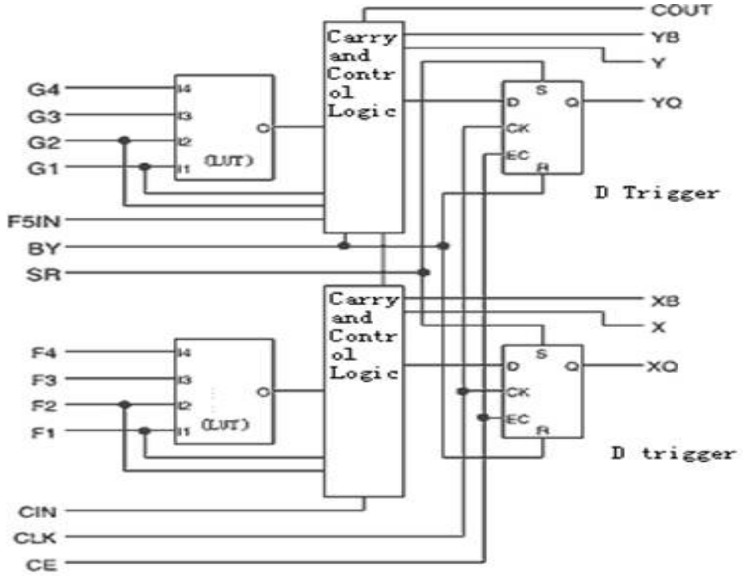


Figure 2.5: Xilinx Slice Architecture [26]

The two largest manufactures of FPGAs are Altera Corporation [20] and Xilinx Incorporated [24]. The basic building block in an Altera FPGA is referred to as a Logic Element (LE) and is shown in Figure 2.4. The basic building block in a Xilinx FPGA is referred to as a slice and is shown in Figure 2.5. The basic building block of the LE is a look-up table (LUT) which is basically a block of programmable memory and is shown in

Figure 2.6. The LUT consists of an array of 1-bit memories which implement a truth table connected to a multiplexed output pin. In short, a Xilinx slice is basically made up of 2 LEs. Altera Stratix II EP1540F1508C5 was selected as the target device for this research. This device contains 41, 250 LE's and was chosen for its popularity and large output pin capability for synthesis of large designs.

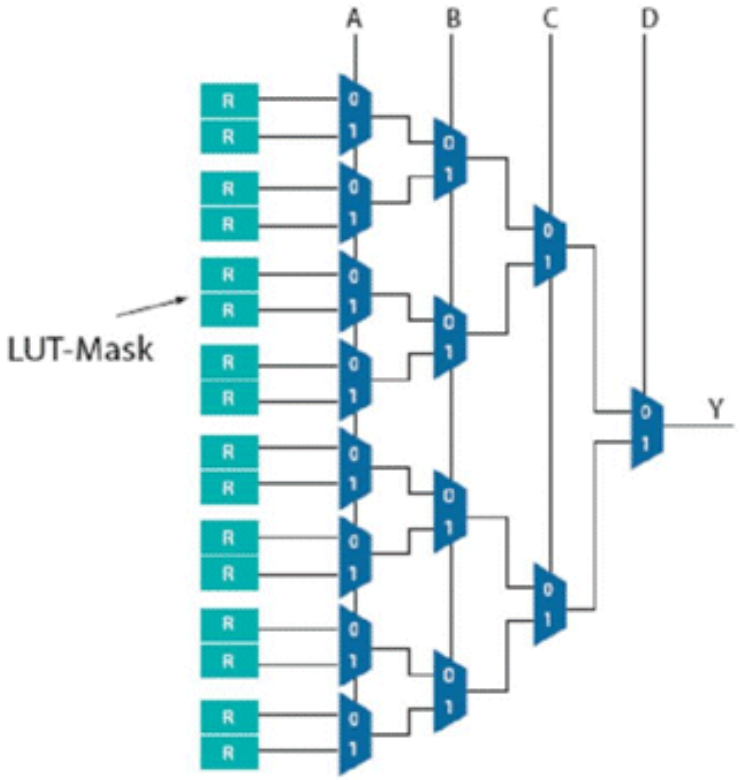


Figure 2.6: Four Input LUT [25]

2.3 Related Work

Our Router has been designed and synthesized on an Altera Stratix II FPGA, therefore although there are a number of ASIC and custom IC implementations, we restrict our discussion of related work to FPGA implementations. This section is intended to provide a comprehensive state of the art for NoCs, although the authors do not pose claims about its completeness. The results of our review are summarized in Table 2.1.

Name, Author, year published, reference []	Novel, useful features	Areas to improve upon	FPGA area synthesis results	Performance features	Testing type and results
N/A, Marescaux, 2002, [6]	first working implementation on FPGA	1-D router configuration leading to same bottleneck problem	611 slices	40MHz clk	no testing
N/A, Bartic, 2003, [7]	improves upon network structure allowing 2-D irregular topology	large routing table logic, setup time, possible blocking due to VCT flow control, without removing buffer requirements	552 slices	50MHz clk	no testing
Hermes, Moraes, 2004, [8][9]	simplifies routing logic, provides parameterizable data width and buffer depth, later work produces CAD tool for auto generation and even traffic analysis (latency)	arbitration and routing logic is centralized creating request bottleneck and restricting simultaneous connection possibilities	5-port/8-bit 278 slices	25MHz clk	simulates 5x5 network with different buffer sizes, latency decrease for larger buffers (perhaps due to central bottleneck)
RASoC, Zeferino, 2004, [10][11] [12]	decentralized routing and arbitration logic, conclusion of shorter network wires leading to higher clock frequencies then in buses.	various packet lengths leads to 2 extra framing bits in each flit and complex routing logic (updates header at each node)	5-port/8-bit Input buffer depth 2 460-570 LE's depth 4 486-795 LE's	55.8-66MHz clk	4 microblaze, master/slave application with shared and local memory, 32-bit bus vs. 32-bit router

Name, Author, year published, reference []	Novel, useful features	Areas to improve upon	FPGA area synthesis results	Performance features	Testing type and results
PNoC, Hilton, 2005, [13]	circuit-switched flow control provides guaranteed throughput, supports any custom topology or number of ports	CS creates setup and teardown latency, blocks channels during idle time, complex routing logic reduces scalability	8-bit 4-port 249 slices 8-port 1113 slices (not including RAM for routing table)	138-151Mhz clk	image binarization application controlled by microblaze, 8 block in total, 1 router vs. shared and locked bus
LiPaR, Sethuraman, 2005, [14][15]	Input and output buffers, dual purpose signals reduces logic, improves crossbar based on XY routing, proposes MLPR idea, later work creates exhaustive CAD tool for mapping IP's and choosing network size	large 5x5 crosspoint matrix, dynamic scheduling? (worth the extra area), slow clock (due to large request/grant process time)	5-port/8-bit 352-432 slices	33.3MHz clk	simulates 1, 1x2, 3x3 router configurations with without blocking and with worst case blocking, takes 24 cycles to process 8-flit packet
GnoC, Vestias, 2006, [17]	supports range of routing, switching, and arbitration algorithms, enables resource sharing to lower area where higher bandwidth isn't needed, CAD tool decides sharing based on injection rate	fully connected switch, only supports output buffering	5-port/8-bit (4 flit buffers) 230 slices extreme case => all ports sharing output logic 88 slices	107 MHz clk	simulation restricted to SAF, RRA, 4 flit packets, 36 nodes, results saw the CAD tool significantly reduce area for lower injection rates
MoCres, Janarthanan, 2007, [18]	takes crossbar reduction (based on XY routing) idea further by designing custom ports	VCT creates setup and teardown latency, blocks channels during idle time, while still using buffers, also goes back to central arbitration which creates performance bottlenecks	5-port/8-bit 282 Slices common clk 5-port/8-bit 332 slices multiple clk domains	286MHz clk (common) 357MHz clk (multiple clk domains) eliminate pad delays	simulates a 3x3 network by implementing a wrapper to inject packets at various rates and reports average latencies

The first working implementation of FPGAs was presented by Marescaux et al. [6]. It has many faults mainly large size, and a one dimensional architecture which fails to provide a high degree of scalability. They extend their work in [7], allowing a more flexible architecture, but still suffering from large area overhead. They use VCT flow control which is now considered too area-intensive for FPGA platforms because of complex routing logic without eliminating any buffer constraints.

Moraes et al, present Hermes, a router with parameterizable data width and buffer depth. They perform simulations of a 5 x 5 mesh while varying buffer depth. They conclude with the notion that increased buffer size reduced latency, but only to a saturation point. Their design uses centralized arbitration and routing units, which decreases area but stalls performance as routing requests are queued to be handled one at a time. Their design also suffers from a very low clock speed. They later extend their work to provide an automatic router generation and traffic analyzer [9].

A comparable router, RASoC [10], was presented by Zeferino et al. The main difference being they use a WH flow control. Performance differences are yet to be compared and may be considered for future work as a WH downfall is that it reserves channels which can cause blocking. However, WH also requires complex routing logic as well as extra bits in the datapath for framing. They also used Altera to synthesis their 5-port, 8-bit router which occupies 486 LE's and has a clock frequency of approximately 57MHz. This area is quite large for a router whose buffers are limited to 4 per port.

PNoc, proposed by Hilton et al in [13], gives us a router with circuit switched flow control. They test their router against bus based approaches to show improvements. However, routing complexity grows as the number of ports, or number of routers increase and therefore reduces scalability. It also suffers typical CS setup and teardown latencies and possible blocked idle time.

Sethuraman et al. propose LiPaR in [14], which was a starting point of our design, but significant improvements were added by us. They use SAF, input and output buffering, and decentralized components. Optimizations are made in the crossbar matrix to reduce area through careful analysis of the XY routing algorithm. However, we extend

these optimizations to the arbitration unit. They use a single 5x5 crossbar matrix for switching rather then 5 5x1 partial crossbars leading to a larger area. Their complex crossbar design results in a slower clock speed and increased area.

They later propose multi-local port routers (MLPR) in [15], which have the potential of improving area and performance metrics. However, the authors fail to provide any synthesis results to support their proposal. Another extension the authors propose is Optimap [16], an exhaustive CAD tool for mapping IP's and choosing network size.

Vestias et al. propose GNoC in [17], a generic router which supports a range of routing, switching and arbitration protocols. They create a tool for exploring the sharing of some decentralized components to reduce area that is based on the injection rate of ports. Unfortunately, they lock all protocols to certain values and do not explore them further. Their tool shows how they can save area when injection rates are low but does not test to see if performance is degraded.

MoCres, designed by Janarthanan et al. in [18], uses complex VCT flow control and attempts to reduce area by sacrificing area through centralizing components. They create multi-clock domain to enable high clock frequencies during transfers. Optimizations from XY routing in the crossbar matrix have been extended to the routing algorithm, and gave us the idea for a further arbitration unit extension. We have also used their idea of creating VHDL wrappers to simulate the stand-alone router or routing configurations to compare parameters.

Our paper attempts to zero in on all the best router characteristics from the above to make as many optimizations in area as possible while concentrating on system performance. We notice a lack of evaluation and comparison of network parameters on FPGAs and try to test accordingly. Most work has focused on dynamic arbitration schemes, mainly round robin (RRA), which may be too area consuming when implementing decentralized components. We see that the data width size is often set to 8-bit flits as many papers assume a size without analysis. Most importantly, we agree with the opportunity to optimize data traffic through use of MLPR. Our plan is to present area

utilization and performance values for the above network parameters to help future designers make accurate decisions for their computing needs.

2.4 Summary

In this chapter, the relevant background material and related previous work was presented. First, a short collection of concise definitions of NoC building blocks was presented. We then listed relevant concepts and theories about NoC Parameters. Our NoC discussion concluded with the presentation of evaluation metrics. Next, the basic concepts of FPGA technology were discussed. Finally, the Chapter concluded with a discussion of some of the previous work that is closely related to this research, and how it was used to motivate our own research. In Chapter 3, a detailed description of the proposed NoC Router architecture hierarchy and functionality is presented.

Chapter 3

A Parameterizable NoC Router Architecture

In this chapter a detailed description of our proposed NoC router architecture is presented. This chapter begins with a discussion of basic functionality of the NoC router based on protocols chosen. That is followed by a discussion of the NoC router architecture describing the main components used and the data flow. NoC design is presented next, including functionality and how to assemble. We briefly discuss how the NoC router architecture was verified using Altera Quartus II CAD tool and then conclude the chapter.

3.1 Functionality

In section 2.1 we discussed NoC router parameters and gave some insight into the choices we have made for our router design. In the following sections, we will discuss those parameters in which directly affect the functionality of the router which include protocols and algorithms.

3.1.1 Protocols and Algorithms

NoC router protocols and algorithms govern the flow of data through the NoC network. They make decisions on where data flows, at what speed, in what order, how it is stored, ect. Therefore they directly affect performance. These parameters are hard to create flexible due to how they often control the router design as a whole. Therefore, their affect on area can also be significant. Careful selection is crucial and there is much work to be done in creating new or testing existing protocols and algorithms for NoC router design.

The following sections describe our protocol and algorithm choices to provide working knowledge of our NoC router. These parameters include flow control, switching mode, routing algorithm, and scheduling.

3.1.1.1 Flow Control

We have chosen a packet switched flow control. In PS networks, data is separated into small blocks called packets at the core. This packet includes a header which has information about its destination. Upon creation of the packet, IP cores simply release the packet into the network where a series of interconnected routers forward the packet to its destination. PS is referred to as connectionless as there is no direct connection between communicating cores. This is an attractive choice as it allows multiple IP cores to communicate without contention.

3.1.1.2 Switching Mode

Switching mode can often be confused with flow control as it plays a large part on the flow of the packet. Switching mode is only a parameter of PS networks. This parameter is in charge of allocating buffers and channels to the packet and deciding when it will receive service. A packet is broken down into flow control units (flits). We have chosen to break the packet into 8 flits. Each flit is the size of the channel. We have chosen a store & forward (SAF) scheme. In this scheme, packets are buffered at each router, and the router waits for the full packet to arrive before forwarding. This prevents a single packet from blocking more than one channel at a time. The downfall is that it increases the buffering requirements of each router. Testing this parameter would be great future work as there are a few other alternatives. However, designing a router with different switching modes is very complex and was omitted from the scope of this research.

3.1.1.3 Routing Algorithm

The routing algorithm is implemented within the router and is in charge of choosing the next hop toward the packets destination. We have chosen XY routing for its simplicity allowing for the implementation of a low area router. XY routing prevents livelock from

occurring. Since all packets leaving the same source and headed for the same destination will travel the same path, it also prevents having to deal with complex scenarios like packet reordering. Unfortunately, using the same logic, XY routing cannot provide any type of congestion control.

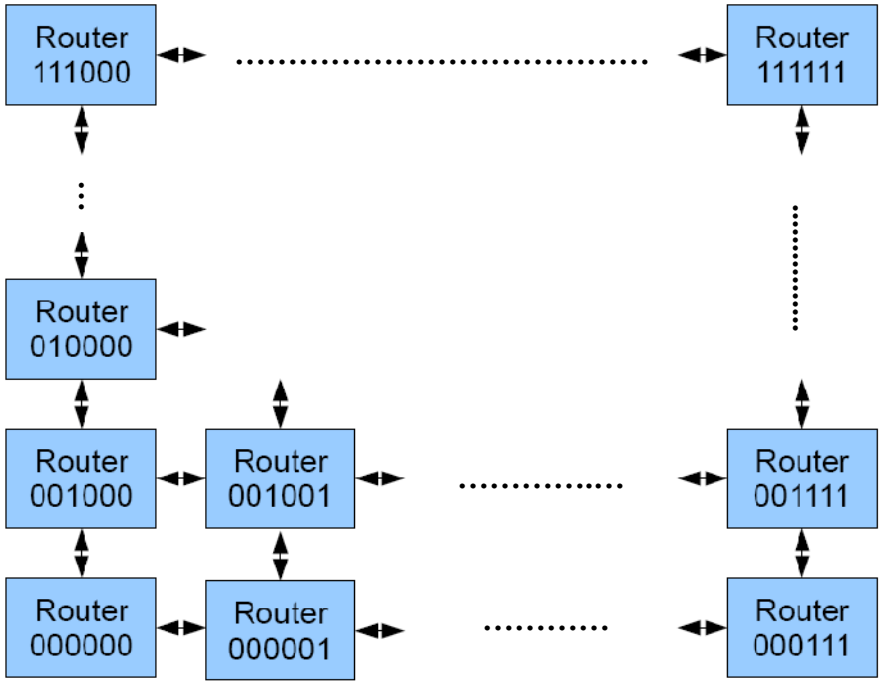


Figure 3.1: Coordinate Configuration for XY Routing

In XY routing, each router is given a coordinate based on its position in the network. We restrict our mesh size to 8X8 and therefore our coordinate is 6 bits. The most significant 3 bits portrays the routers vertical displacement with 000 being the lowest (southern) router and 111 being the highest (northern) router. The least significant 3 bits portrays the routers horizontal displacement with 000 being the left most (western) router and 111 being the right most (eastern) router. Figure 3.1 shows router coordinate configuration within a mesh. A packet arrives at the router with an 8 bit header. This header contains the destination of the packet. The vertical displacement is checked first. If the destination is greater then the coordinate, the packet is forward north. If the destination is lesser then the coordinate, the packet is forward south. If the destination is equal to the coordinate, then its vertical displacement is ok. The same process then

occurs for the horizontal displacement. Eventually, the packet arrives at the router with the proper coordinate. At this point the packet is at the proper port and must now be forwarded to the correct destination port. Since routers in our mesh can have up to 4 ports, the least significant 2 bits of the header are used to distinguish among local ports. Figure 3.2 shows the configuration of local ports within the router.

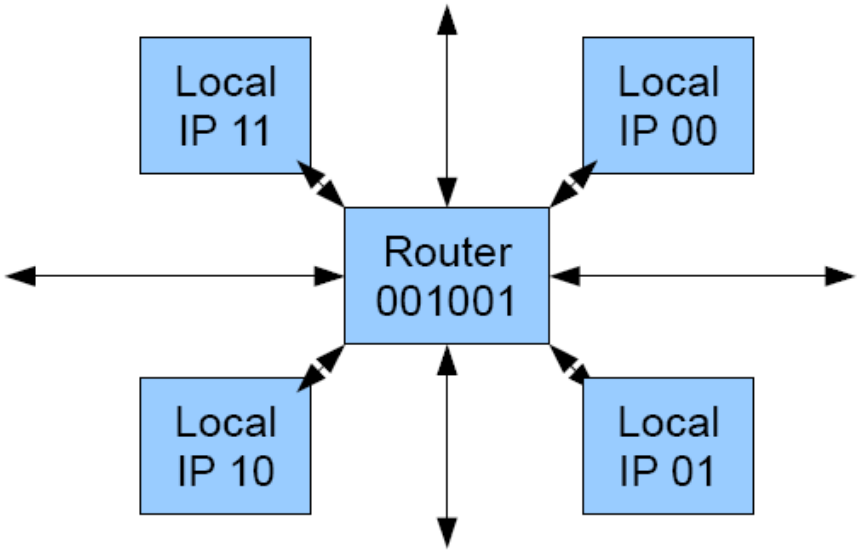


Figure 3.2: Configuration of Local Ports for XY Routing

An important note can be made about this algorithm. Since the vertical displacement is always found first, a packet coming in from the east or west ports must already be in its proper vertical position. Therefore, a packet coming in from the east or west ports cannot be forwarded north or south. This observation is exploited later to optimize the area selected components.

3.1.1.4 Scheduling

Scheduling of data depends largely on IP cores. However, scheduling can occur within the network. If two or more packets request the same port at the same time or while it's busy, the requested (output) port will have to make a decision on which to grant access first. This is called arbitration. Our router allows for some flexibility in choosing

arbitration schemes that consider priority of packets in routers among the network and are classified into static and dynamic schemes.

In static arbitration schemes, the priority of each port is chosen during design. First, we use a generic fixed scheme where priority is given to the north first, and degrades clockwise. We use two other static arbiters, both based on the fixed scheme. Both schemes were designed during the evaluation phase. Custom scheme was designed based on the setup of the simulation. Custom each port scheme included a different fixed priority in each port based on the setup of the simulation.

Dynamic arbitration makes a decision at run-time and is more flexible, however also requires a larger area. However, dynamic schemes can avoid deadlock. We include 3 counting schemes and a coin passing scheme. The counting schemes all have similar area results, but their performance depends on the application. The first scheme gives priority to the port that has been busiest (sending the most requests). The Next scheme gives priority to the port that has been waiting the longest. Here, the arbitration unit counts cycles after a request has been received for all ports. The last counting scheme gives priority to the port that sends the least packets (opposite to the first scheme). Finally, in coin passing scheme, one input port is assigned the coin. The port assigned with the coin, has priority, until it has been granted. Then the coin is passed to the next port, clockwise. If the port with the coin is not making a request, the unit grants the request of the port closes to it, again clockwise. This scheme is much like round robin used in many FPGA NoC router implementations.

Scheduling is one of the parameters we wish to test. Interesting results may show static arbiters latency is quite reasonable considering its area savings. This is especially a concern in decentralized routers, where each port has its own control logic.

3.2 Router Implementation

The router was designed with 4 ports for communication with neighboring routers, North, East, South, and West and anywhere from 0 to 4 local ports for communication to IP cores. A router with no local ports would be used just to complete a mesh and help with congestion control within the network. Generic port and component design was used,

therefore and input port has the ability to forward to its own output port, although this situation could never occur. Figure 3.3 shows the port architecture and it's interaction with the switch. Packet size has been set to a depth of 8. Flit size is parameterizable, with 8 bits being the smallest possible size for routing information purposes. Our implementation does not include High Level Protocols (HLP) but could easily be implemented on an application level. The router is decentralized meaning each port runs its own control logic and hence can request and set up concurrent connections. Below we will include details on inter-router data transfers, the I/O channels and the crossbar switch designs.

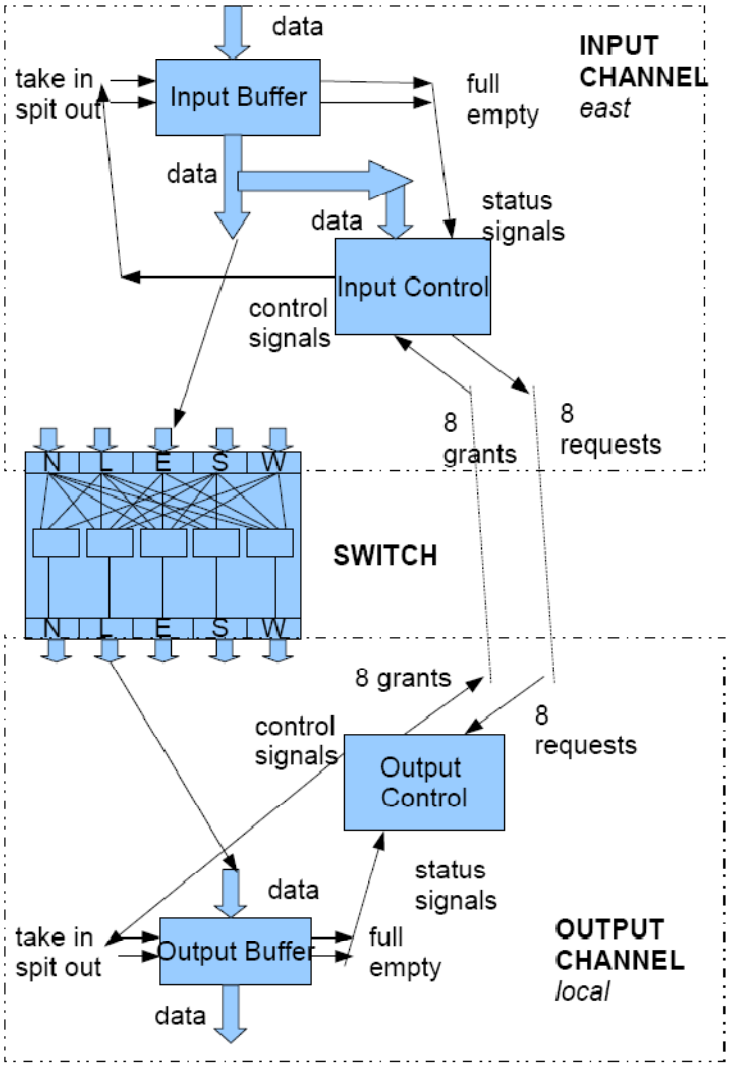


Figure 3.3: Architecture of Port: I/O Channels and Switch

3.2.1 Data Transfer between Input and Output Ports

Communication between ports is established by use of a two-way handshake of request/grant signals. Figure 3.4 shows a handshake scenario between local and west ports. Upon packet arrival, local sends a request for west's output port. Once local receives a grant from west it can drive its request line back to low and it is free to send the packet, one flit at a time. West will hold its grant line high until the full packet has been received. Any other ports which have high request lines to west, will keep them high until they also receive a grant.

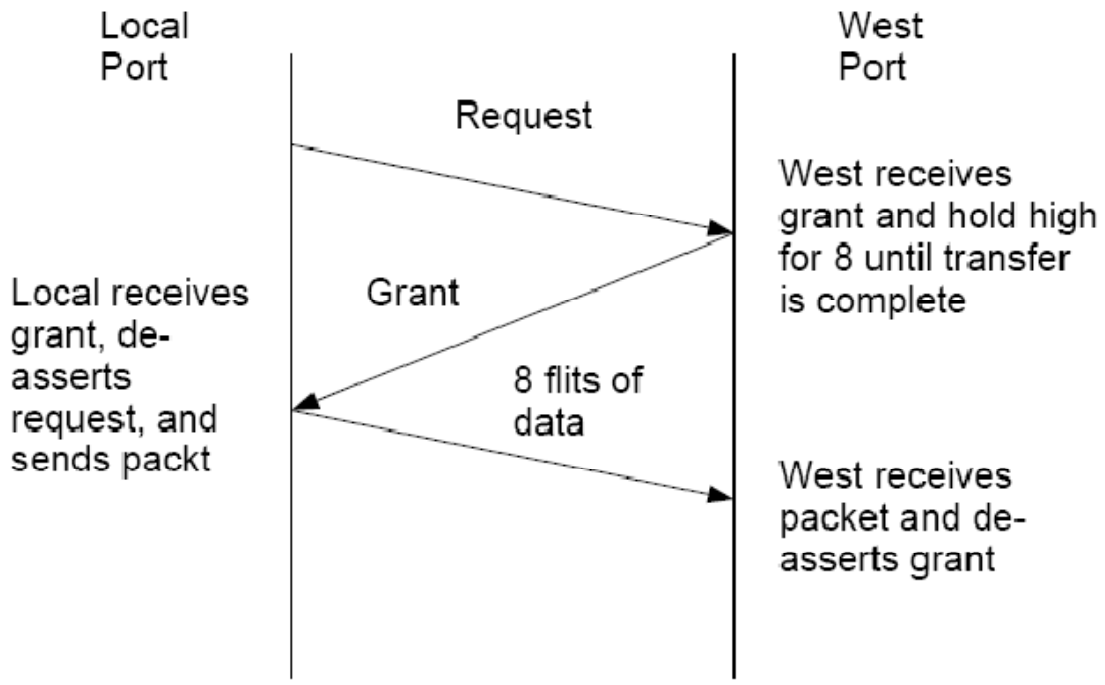


Figure 3.4: Handshake Scenario between I/O Ports

3.2.2 Input Channel

All input channel modules include a buffer unit of depth 8 and a logic controller. This module grants access to input buffers, accepts and stores packets, performs routing algorithm, issues requests to appropriate output ports, and sends data to the switch.

3.2.2.1 Input Buffer

The input buffer is shown in Figure 3.5. It is capable of storing the whole 8 flits of the packet. It has 2 status signals letting the input controller know if it is full and ready to be forwarded or it is empty and ready to accept a new packet. It also has 2 control signals allowing the input controller to store or forward its contents.

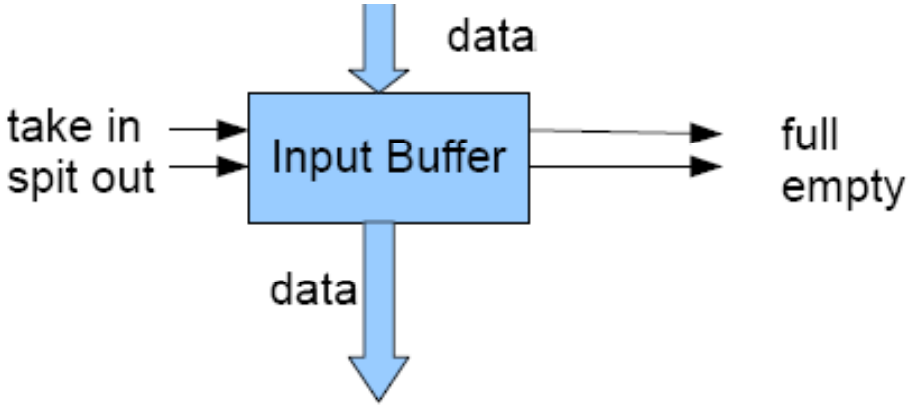


Figure 3.5: Architecture of Input Buffer

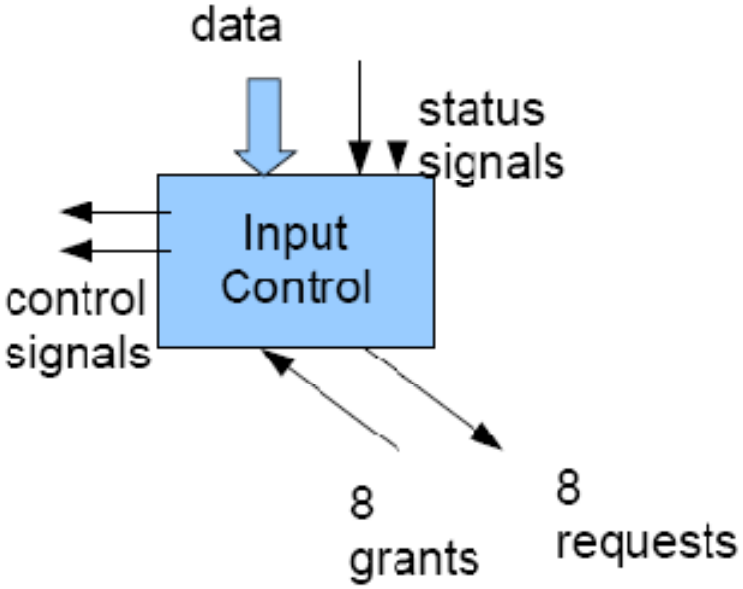


Figure 3.6: Architecture of Input Controller

3.2.2.2 Input Controller

The input controller is shown in Figure 3.6. This unit is responsible for running the routing algorithm. It continually monitors the header flit and determines its next hop. When the buffer becomes full, the controller issues a grant to the appropriate output port. It then waits for the grant, when it can prompt the transfer. When the buffer becomes empty, the input controller can prompt transfers from the outside.

3.2.3 Switching Mechanism

The crossbar switch is shown in Figure 3.7. It is a set of multiplexers having an interconnection allowing all possible connections between input and output channels. Three optimizations have been made in the crossbar switch. First, it uses a partial scheme, which includes one 5 by 1 unit for each output rather then one 5 by 5 unit for all outputs, for a 5 port router. Initial design included 2 switching options, full and partial switch. Early synthesis results eliminated the full switch design because it was larger in area and slower in clock frequency. Each output is connected to a different port. Next, there are no demultiplexers in the design. The input data is connected to all partial crossbar units which will choose the appropriate data for the output. The fact that at a time, the output channel can only serve one input request is exploited here. The final optimizations are made in the partial units of the north and south. Though analysis of the XY routing algorithm, we can conclude that these units will never receive data from the east or west. This reduces the inputs of all of these units by two. All optimizations reduce the area without effecting latency of the router.

3.2.4 Output Channel

All output channel modules include a buffer unit of depth 8 and a logic controller. This module grants access to output buffers, accepts and stores packets, performs arbitration, issues requests to the next hop, and sends data to the next hop.

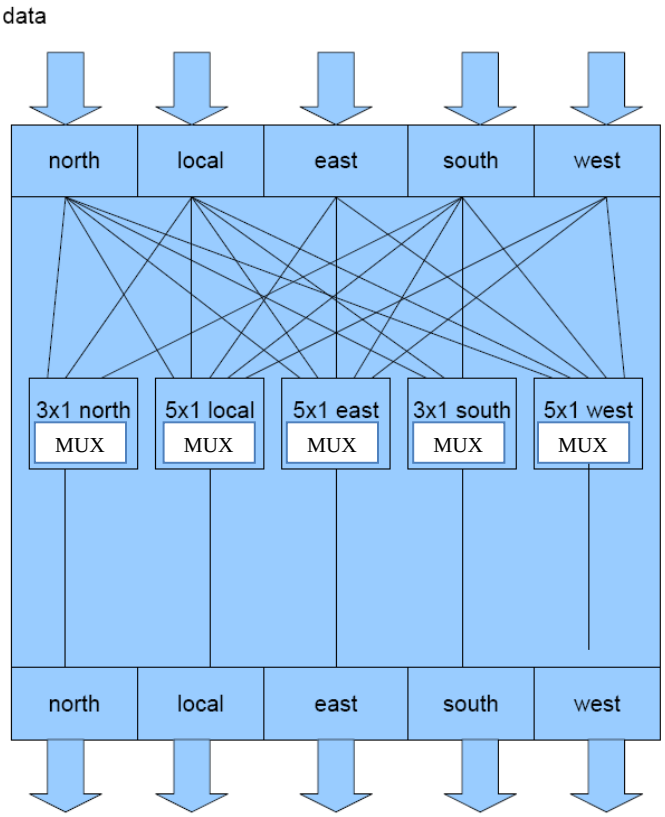


Figure 3.7: Architecture of Switching Fabric

3.2.4.1 Output Buffer

The output buffer is shown in Figure 3.8. It is capable of storing the whole 8 flits of the packet. It has 2 status signals letting the output controller know if it is full and ready to be forwarded or it is empty and ready to accept a new packet. It also has 2 control signals allowing the output controller to store or forward its contents.

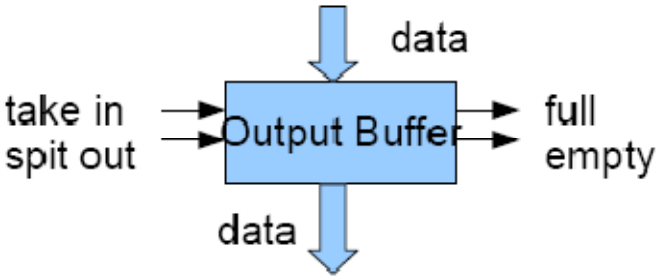


Figure 3.8: Architecture of Output Buffer

3.2.4.2 Output Controller

The output controller is shown in Figure 3.9. This unit is responsible for running the arbitration algorithm and making grants. It continually monitors request line. When one or more become high, the controller issues a grant to the prioritized input port. It then waits for the packet, when it can prompt the transfer outside. When the buffer becomes empty, the output controller can continue issuing grants.

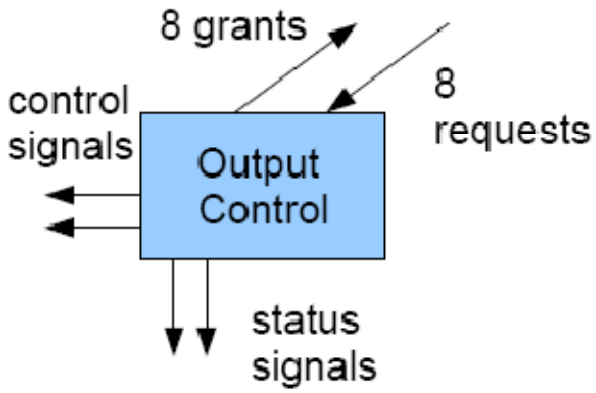


Figure 3.9: Architecture of Output Controller

3.3 NoC Architecture

Although we propose the design of a stand-alone router with the purpose of testing its parameter, the router can of coarse be used in the building of a NoC network. The router protocols will only work for two dimensional mesh architecture, with properly positioned coordinates. The following sections discuss the intra-router data transfers, along with how to build a NoC using the proposed router and accompanying components.

3.3.1 Data Transfer between Routers

Figure 3.10 shows the proposed router with its external signals. For each port, there are 4 generic control signals, and 2 data paths. One data path is for incoming packets while the other is for outgoing packets. Similarly, 2 control signals are input controlled and two are output controlled. Figure 3.11 shows interaction among 2 adjacent routers, mainly the

transaction of data from a routers output port to the other routers input port. The output port will let the adjacent routers input port know when its buffer is empty, and therefore ready to receive a packet, through use of emptyout/emptyin signals. Once the output port has received a packet and the emptyin signal has been driven high by the adjacent router, it can begin sending. To start, it drives the sendingout signal high for just one clock cycle, which prepares the output port that all 8 flits will begin transferring upon the next clock cycle. When data begin to flow, the input port will send the emptyout signal to a low state, and stay that way until the packet has been forwarded within that specific router.

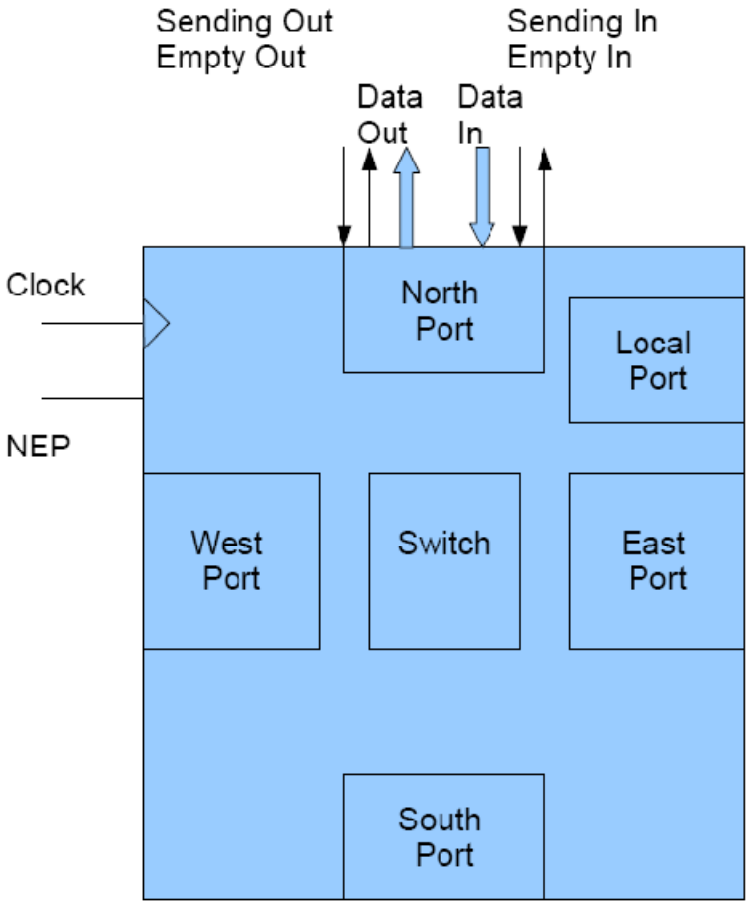


Figure 3.10: Architecture of Proposed Router

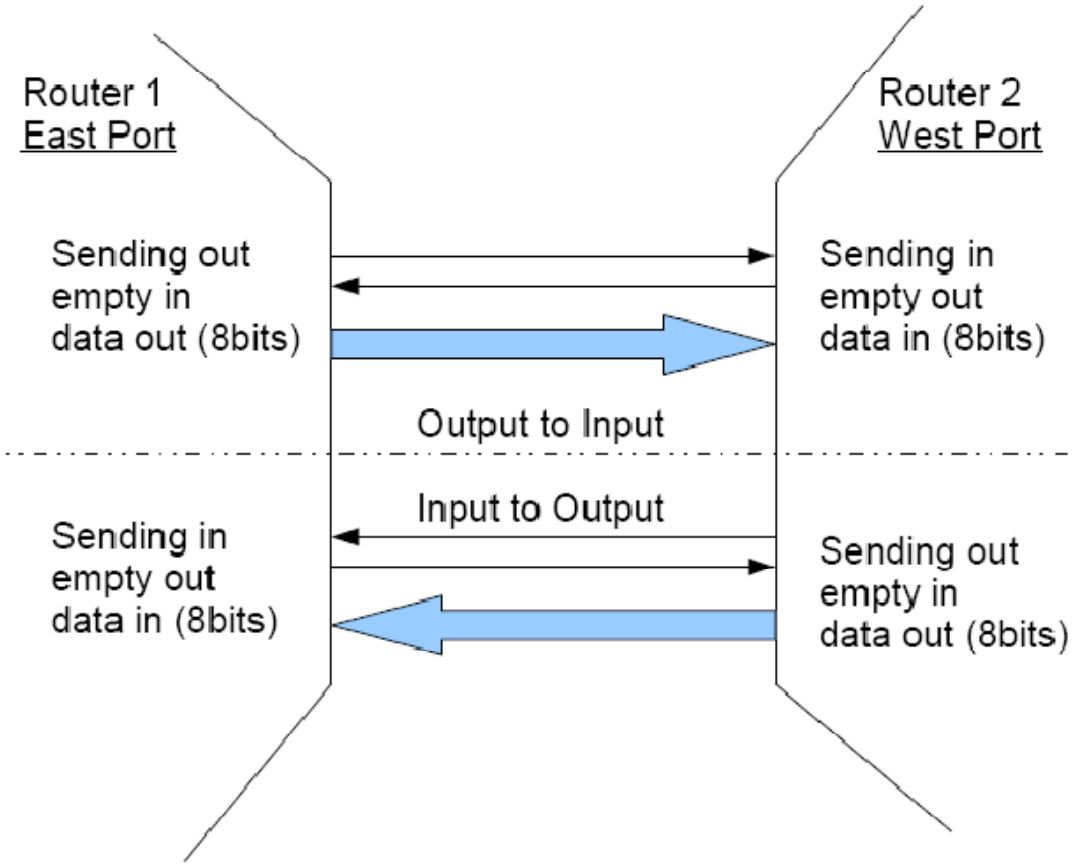


Figure 3.11: Connections between Adjacent Routers

Since are router provides a variable number of ports, and extra signal was added, NEP to make sure that there are no requests for a non-existent local port. This signal is driven high if the header of a packet requests a local port that is not included in that particular router. This signal is common on all routers and could be sent to a central processor (CPU), where it could stop the program and re-assess where the IP cores are in terms of NoC network position.

Future re-design/improvements of this router architecture would see the output control logic increase. This would be to provide an option to skip the output buffer if the receiving router is ready. Although the output buffer is useful in preventing HOL blocking, it adds un-needed delay in cases where the receiving router is ready.

3.3.2 Building a NoC Network

With our parameterizable router design, building a router takes careful placement of components. The design makes choices on channel width, number of local ports per router, and arbitration type. Also, when building a NoC, designers must configure the coordinates of each router based on its network position. Figure 3.12 shows a flowchart outlining how to build a NoC network. It is recommended to start with the lower left corner of the NoC and maintain a pattern when designing each router. This is to make integration of the coordinate a little easier.

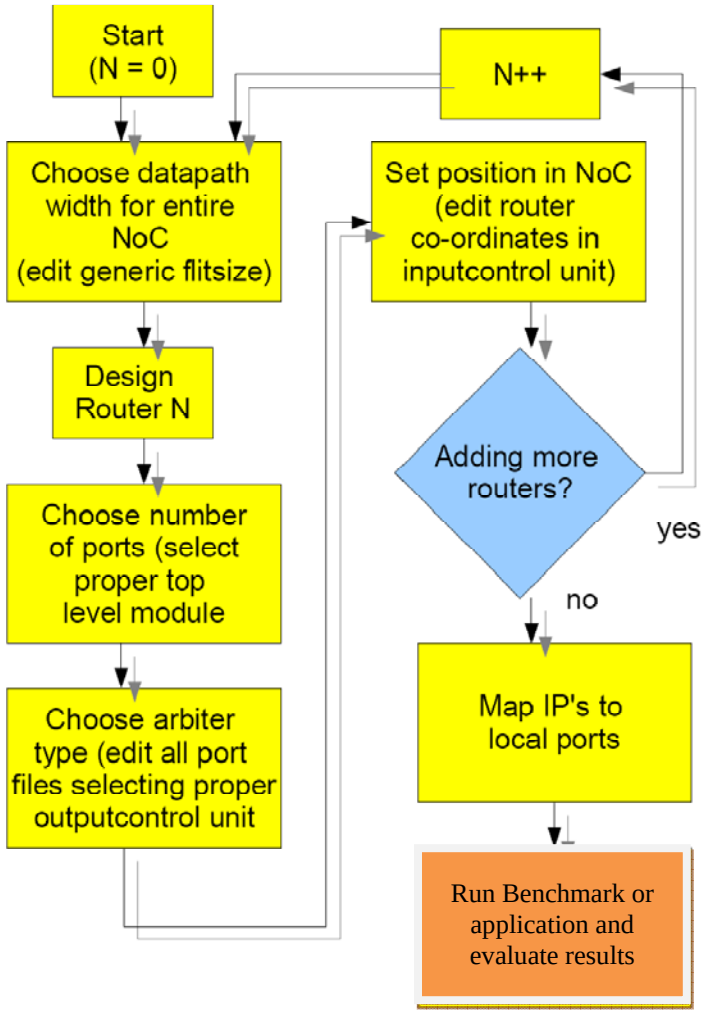


Figure 3.12: NoC Router Design Flow

First is the choice of channel width. This cannot be made smaller then 8 bits for routing purposes. Once chosen, it must remain the same for all routers within the NoC network. This can be modified in the top level module by changing the generic parameter called flit size.

Next is the choice over how many local ports. Each router in the NoC network can have anywhere from 0 to 4 local ports independent of the other routers. This option is as simple as choosing the correct top level module for router design. Each top level router module is named after the number of local ports it has (router_fs_xx). Here the xx should be chosen to be 0l, 1l, 2l, 3l, or 4l implying the number of local ports each router has.

The final choice to be made is on which arbitration unit is to be used. The number of changes made here depends on the number of ports chosen. Each port must be opened separately, to change the name of the output controller used. Ports are named based on numbers (port_ns_x). Here the x is a number and is based on the number of ports in the design. So if you build a 6 port router, ports with numbers 0 to 5 should be opened and changed. Output controllers are named after their arbitration unit (outputcontrol_xx). Here the xx is replaced by select options listed in Table 3.1.

Table 3.1: Coding Scheme for Different Arbiters

Arbiter code	Arbiter Type
fa	Static – fixed scheme
c1, c2, c3	Dynamic – various counting schemes
cp	Dynamic – coin passing (RRA) scheme
cap	Static – custom application fixed scheme
ca0, ca1, ca2, ca3, ca4	Static – custom each port for application fixed scheme

Lastly, the router must be configured with its proper coordinates. Router positioning must be done carefully. Coordinates are configured in the input controller (inputcontrol) and are called router_coordinates.

Future re-design/improvements of this NoC router would see the use of corner routers which would eliminate the need for a couple ports further reducing area. Since we simply wish to test the parameters, this omission is fine. This reduction in area and power would be crucial for use of this router in a NoC system for real world implementation.

3.4 Verification

ModelSim [21] was used to create tests and obtain latency results, but before that phase testing on the routers functionality was needed. After choosing the parameters of the router, design began with the creation of components (buffers, controllers). As each component was finished, it needed to be tested for functionality. Each component would have to be verified before moving on in the design. There is no point in creating a router with components that do not work correctly. This phase was completed in Altera Quartus II CAD tool using the waveform editor and simulation tool. The output controller needed extra testing as the priority of each port needed to be tested for different arbiter types

First each component was tested to ensure its control signals were working and data was flowing through the components with proper timing. Then came creation of a generic port composed of the components. Testing was done to ensure each port could receive and send data properly. Finally a 5-port router was designed. As it is nearly impossible to verify every possible bit stream scenario in a router, a few situations were simulated. First was a common transfer. The north port obtained a packet and was sent to the south. Next was a concurrent test where the local port was requested and provided service to the west and east ports. Finally, a test was developed where all five ports obtained a packet and five simultaneous connections were established.

Figure 3.13 shows the output waveform for east to north transfer. Only the east and north ports signals are shown. Note that the sending in, si, signal of the east port go high. East begins to receive the packet. Once it has been received and routed, we see the

north output buffer begins to receive it. This figure shows all sending and empty signals and how they act during transfer. The input data was held steady at 11001001 for input to the east port.

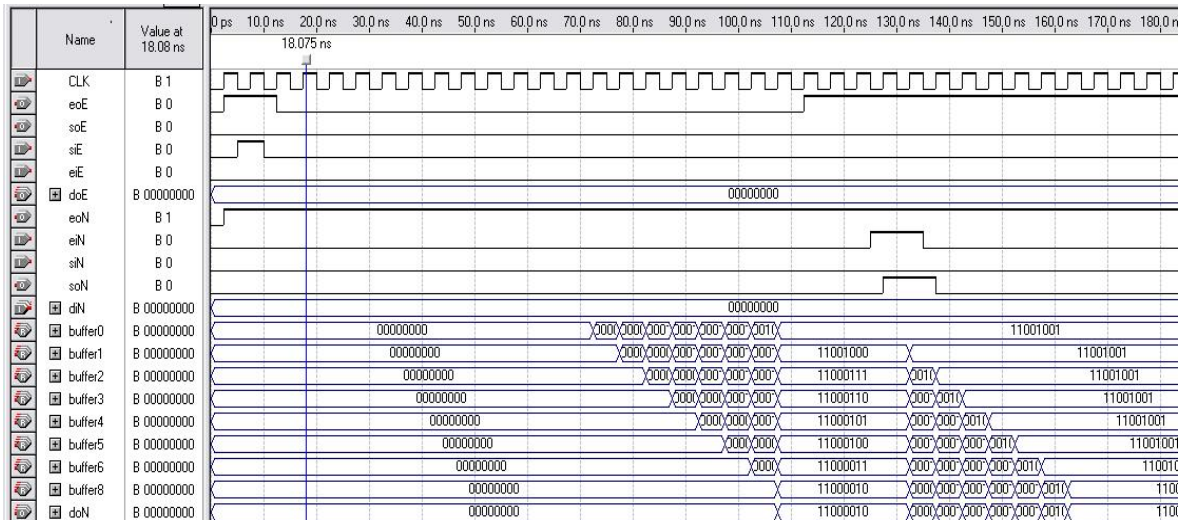


Figure 3.13: East to North Transfer Simulation Output in Altera Quartus II CAD tool

Other routers were created containing different number of local ports. Since each port is generic, building a bigger router should be similar. Therefore verification was as easy as creating a transfer which utilizes the new port to make sure it has been wired correctly.

For router with less the 4 local ports, another quick test was developed to ensure the functionality of the NEP signal. Here, a packet sent with the heading for a local port that was not there.

3.5 Summary

In this chapter, we discuss functionality of the router protocols and how a packet flows through the network. We provide a detailed description of the proposed NoC Router architecture hierarchy and design. The NoC network design and functionality was then presented. The chapter concluded with a description of the design and verification process. In Chapter 4, experimental evaluation results, and their analysis are presented.

Chapter 4

Experimental Evaluation Results

This chapter starts with a discussion of the design methodology for implementing a Network-on-Chip (NoC) system. This methodology also facilitates rapid prototyping and exploration of various aspects of NoC implementation. This is followed by a brief description on the difficulties involved in testing NoC components. Then, we describe how we choose to test each parameter of our NoC router. Lastly, the synthesis and simulation results are presented and analyzed.

4.1 Design Methodology

As previously discussed, NoC routers have many parameters. These parameters can often be flexible for allowing different choices for different target applications. Choosing parameters can often be a difficult task for embedded SoC system designers. In the following sections, we aim to provide insight into the latency and area trade-offs to allow designers to make informed decisions.

4.2 Synthesis Results

We use Altera Quartus II CAD tool [20] to synthesize the system to obtain area utilization and clock frequency values. We chose to target a popular Stratix II FPGA family, device EPIS40F1508C5. All components and modules have been implemented in VHDL. Components were originally tested for functionality in Altera Quartus II CAD tool environment. The router coordinates are set during synthesis before simulation takes

place. We have performed experiments on the parameters arbitration type, flit size, and configuration. Table 4.1 shows synthesis results for our 5-port, 8-bit router using fixed arbitration. Here area utilization is broken down into usage by each component. For full fitter utilization results, refer to Appendix A. The router consumes only 598 (1.45%) LEs, making it one of the most competitive NoC routers with standard features (I/O buffers, decentralized routing and arbitration logic) targeted for FPGAs. This shows a very small input control due to the simple routing algorithm. We notice a large output control unit, motivating our arbitration test as this unit is already large. We notice the buffers taking up most of the space. This may prompt future experiments involving other flow control protocols which do not require such large buffer sizes.

Table 4.1: Area Utilization for Router Components

Component	**Sub-Component**	**Area (LE's)**
Input Channel	Input Buffer	157
	Input Control	54
Output Channel	Output Buffer	156
	Output Control	124
Switch	---	107
Router	All	598

Here we would like to make a comparison with one of the most competitive routers in current research, LiPaR. Table 4.2 shows LiPaRs synthesis results. All our components are comparable in total area, if not smaller. Consideration should be taken for the fact that they used Xilinx to synthesize the design and that Xilinx slices are much larger then Altera LE's. Also, Xilinx has embedded FIFO buffers which were used in the design. We created our own FIFO buffers and accompanying signals which may be the reason for the increased area.

Table 4.2: Area Utilization for LiPaRs Router Components

Component	Sub-Component	Area (Slices)
Input Channel	Input Buffer	105
	Input Control	65
Output Channel	Output Buffer	105
	Output Control	132
Switch	---	78
Router	All	485

4.2.1 Arbitration

Our first experiment was performed on Arbitration type. Here, a 5-port, 8-bit router was synthesized many times swapping in different output controllers. Each output controller contained a unique arbitration unit as described in Section 3.1.1.4. Table 4.3 and 4.4 show the synthesis results for this experiment optimized for area and speed, respectively. We can conclude that static fixed arbitration schemes are the least area expensive components. The clock speed seems to get worse for more complex designs. We cannot yet analyze the latency as clock speed is just a small aspect of the overall speed. We will use these results to obtain accurate latency metrics in Section 4.5.

Table 4.3: Effect of Arbiter Choice on FPGA Utilization, Optimized for Area

Arbiter Type	Area (LE's)	Memory		Clock Speed (ns)
		M4k's	M512's	
Fixed	598	0/183	8/384	9.71
Counter	1191	0/183	8/384	10.89

scheme 1				
Counter Scheme 2	1173	0/183	8/384	10.13
Counter Scheme 3	1191	0/183	8/384	11.4
Coin passing (RRA)	746	0/183	8/384	9.58
Custom Fixed	598	0/183	8/384	10.14
Custom Fixed Each Port	598	0/183	8/384	9.79

Table 4.4: Effect of Arbiter Choice on FPGA Utilization, Optimized for Speed

Arbiter Type	**Area (LE's)**	**Memory**		**Clock Speed (ns)**
		M4k's	**M512's**	
Fixed	1036	0/183	0/384	9.58
Counter scheme 1	1619	0/183	0/384	8.72
Counter Scheme 2	1606	0/183	0/384	10.56
Counter Scheme 3	1618	0/183	0/384	8.85

Coin passing (RRA)	1182	0/183	0/384	11.23
Custom Fixed	1036	0/183	0/384	10.52
Custom Fixed Each Port	1036	0/183	0/384	10.3

4.2.2 Flit Size

Our next experiment was performed on the flit size or data path size. Here, a 5-port, 8-bit router was synthesized 4 times increasing the size of the data path each time. Table 4.5 and 4.6 show the synthesis results for this experiment optimized for area and speed, respectively. We can conclude that larger flit sizes lead to more area intensive components. The clock speed also seems to get worse for more complex designs. We cannot yet analyze the latency as clock speed is just a small aspect of the overall speed. We will use these results to obtain accurate latency metrics in Section 4.5.

Table 4.5: Effect of Flit Size on FPGA Utilization, Optimized for Area

Flit Size	**Area (LE's)**	**Memory**		**Clock Speed (ns)**
		M4k's	**M512's**	
8	610	0/183	8/384	9.71
16	738	0/183	8/384	9.57
32	994	8/183	0/384	10.16
64	1505	16/183	0/384	10.87

Table 4.6: Effect of Flit Size on FPGA Utilization, Optimized for Speed

Flit Size	Area (LE's)	Memory		Clock Speed (ns)
		M4k's	M512's	
8	1036	0/183	0/384	9.58
16	1596	0/183	0/384	9.45
32	2716	0/183	0/384	8.91
64	4956	0/183	0/384	11.91

4.2.3 Configuration

Our last experiment was performed on the configuration. Here, multiple versions of our router were synthesized in many different topologies. These topologies are discussed in greater detail in the following section. During simulation, different mappings were also tested, but this did not affect the synthesis results and is not shown here. Table 4.7 and 4.8 show the synthesis results for this experiment optimized for area and speed, respectively. We can conclude that NoC network topologies with a smaller number of routers are less area expensive, even though the routers tehmselves are larger. The clock speed also seems to get better with fewer routers involved in the NoC topology. We cannot yet analyze the latency as clock speed is just a small aspect of the overall speed. We will use these results to obtain accurate latency metrics in Section 4.5. We have also included some routing resource information in Table 4.9. For full fitter routing resource utilization, please refer to appendix A. It also seems adding more ports or routers to the topology will increase wire and interconnect usage.

Table 4.7: Effect of Configuration on FPGA Utilization, Optimized for Area

Configuration	Area (LE's)	Memory		Clock Speed (ns)
		M4k's	M512's	
Single Router	530	0/183	6/384	8.25
1x2 Mesh	986	0/183	10/384	9.19
1x2 Mesh Extended	1107	0/183	12/384	10.02
2x2 Mesh	1847	0/183	22/384	10.25

Table 4.8: Effect of Configuration on FPGA Utilization, Optimized for Speed

Configuration	Area (LE's)	Memory		Clock Speed (ns)
		M4k's	M512's	
Single Router	829	0/183	0/384	7.5
1x2 Mesh	1486	0/183	0/384	8.97
1x2 Mesh Extended	1722	0/183	0/384	9.36
2x2 Mesh	2926	0/183	0/384	9.05

Table 4.9: Effect of Configuration on Routing Resource Utilization

Configuration	Direct Links /163,680	Global Clocks /16	Local Routing Interconnects
Single Router	167	4	405
1x2 Mesh	295	6	687
1x2 Mesh Extended	323	7	777
2x2 Mesh	486	12	1291

4.3 Router Performance

NoC Systems are still in the research phase and not many implementation results are available in the literature. For some FPGA synthesized designs, testing speed is as easy as maximizing the clock speed. An example of such design is a Microprocessor. This unit retrieves commands and data, performs operations and stores answers. There is only one logical path in the design, in which each of these phases is completed. These things take place one after the other and the speed at which the application can finish depends on how fast each stage can be completed (clock speed).

A NoC router involves multiple ports receiving data, a central switch which can be configured to send data multiple ports for transmission. Although clock speed plays a role in the overall latency, it is not the only factor. The authors of Hermes [8], attempt to compare recent router designs using the calculated maximum (or best-case) throughput. This is when all input ports can request simultaneous connections with different output ports. They use simple mathematics involving flit size, clock speed, and number of ports to determine latency results.

For our in-depth test of parameters, this was not acceptable. For instance, when varying arbitration type, area will grow for more complex designs. Although no difference would be seen in maximum throughput calculations, a more complex arbiter may schedule transactions to prevent blocking which could lead to increase performance. Therefore, our tests involved some form of experimentation.

The recent increase in SoC system implementation has lead to research for new communication architectures. Unfortunately, there is a lack of commonly accepted methodology for performance analysis amongst NoC design research for FPGAs. Some of the best results have come for simple applications being run on both a standard bus and a NoC system. For our research of a NoC router, implementing an application would involve use of IP cores and NoC interfaces. It would seem much more time efficient to simply inject packets into a stand-alone router.

To create a test, we once again looked to academic research. Since these NoC router parameters have never been tested, no benchmarks could be found. One idea is the use of a traffic generator to provide traffic patterns which allow comparison of router parameters as well as comparison among other routers. Researchers are working on a model which generates and absorbs traffic that simulates the behaviour of a real IP core [27]. This project is not yet finished. They have also proposed the use of a theoretical model to calculate performance in [28]. Another research group at the Royal Institute of Technology is working on a simulator that uses synthetic workloads and models real applications [30]. This simulator is designed specifically for use in simulating a two dimensional mesh, and the tool itself is designed for testing their router only, Nostrum [29]. The research group from the University of Rostock designed E-core [31]. E-core is a traffic source and/or sink which is modeled in VHDL. We explored this option as a possible test, since we could use it as an IP core. The problem was that this module had much different control signals then our router and the VHDL comments were all in German. We also recognized that although we would be using another research groups' work, we could not perform a comparison as they have no published results.

We found it would be easier and just as useful to create our own test benches. Benefits to this include creating traffic flow tailored to test each parameter, creating reactive traffic scenarios, having proper control signals. The following sections discuss the detailed descriptions of the performed simulations.

4.4 Experimental Evaluation Framework

For this research, it is important to define a framework that helps guide this research for exploring the design space for NoC routers. As discussed earlier in Chapter 2, NoC router architectures have a vast design space. Figure 4.1 illustrates the design space for NoC architectures, including choices for our router design. The following sub-sections describe how each parameters simulation evaluation was set up.

Parameter Class	Adjustable Parameters	Set Parameters
Infrastructure	Topology =different size meshes Channel width =different flit sizes	Buffering =8 Input & 8 Output Floorplanning =N/A
Communication Mechanism		Flow control =packet switched Switching mode =s&f Switching mech. =partial crossbar Routing algoithm =XY routing
Mapping	Scheduling =different arbitration units Module mapping =different map schemes	

Figure 4.1: Proposed NoC Router Design Space

4.4.1 Arbitration

We implemented a wrapper around our stand-alone router with different arbitration units embedded within. This wrapper focused on sending to the local node to create arbitration dilemmas, although packets were sent and received by all ports. In total 111 packets were sent out from various ports in groups from as small as 1 to as large as 10.

4.4.2 Flit Size

We implemented various wrappers around our stand-alone router with different datapath sizes (flit size). This wrapper was based on the traffic in the arbiter type test, but with larger packet sizes. Two tests were created for testing flit size.

In test 1, a total 544 packets were sent out from various ports in groups from as small as 16 to as large as 64. The number of packets stayed the same as the flit size was increased. Here the amount of data transferred was also increased with flit size.

In test 2, a total of 544 packets were sent out for a flit size of 8. Each time the flit size doubled, the size of the packet groups sent out was cut in half. Here, the amount of data transferred stayed the same as flit size was increased. For example, if 544 8-bit packets were sent, only half that (272 packets) would be needed for 16-bit flits.

4.4.3 Configuration

We implemented various wrappers around different configurations of mesh size, number of local ports and mapping. This wrapper was designed to model a 4 IP core application. The 3 basic configurations are shown in Figures 4.2, 4.3 and 4.4. In total 201 packets were injected into the mesh through local ports. IP core 1 acted as the central processing node sending a total of 160 packets to IPs 2 and 3. IP cores 2 and 3 acted as custom logic blocks receiving 20 packets at a time and responding 5 packets to IP 4. IP core 4 acted as an output display of some sort, receiving the resulting 5 packets from IPs 2 and 3 each stage in the application. The application ended with IP core 4 sending a final packet to IP core 1.

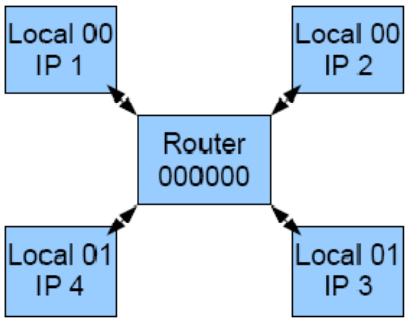
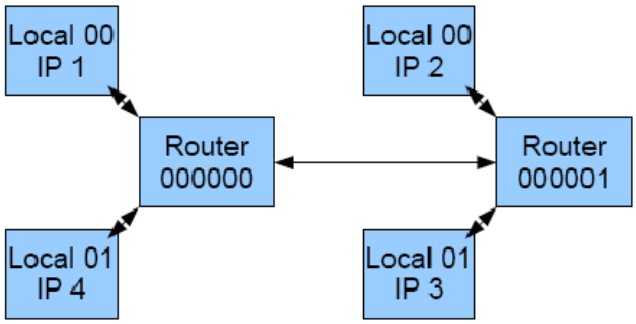
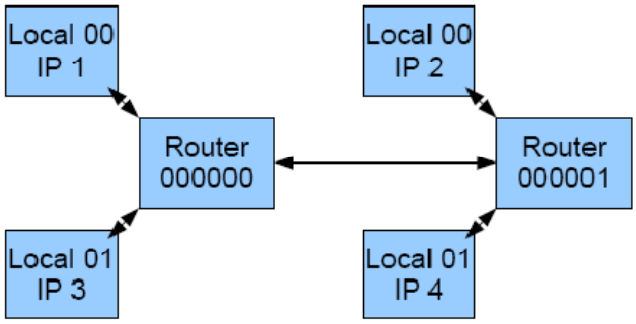


Figure 4.2: Single Router Architecture

a)



b)



c)

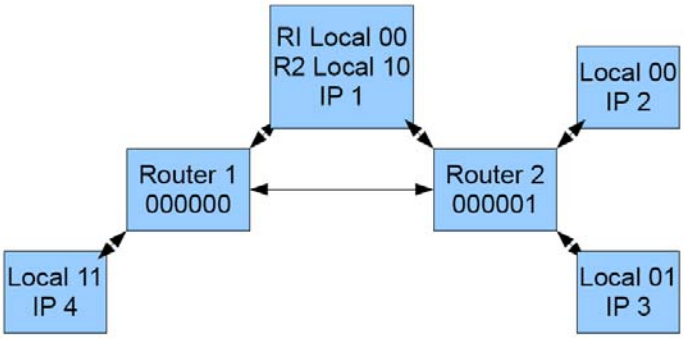


Figure 4.3: 1X2 Mesh Architecture a) Map 1 b) Map 2 c) Map 2 extended

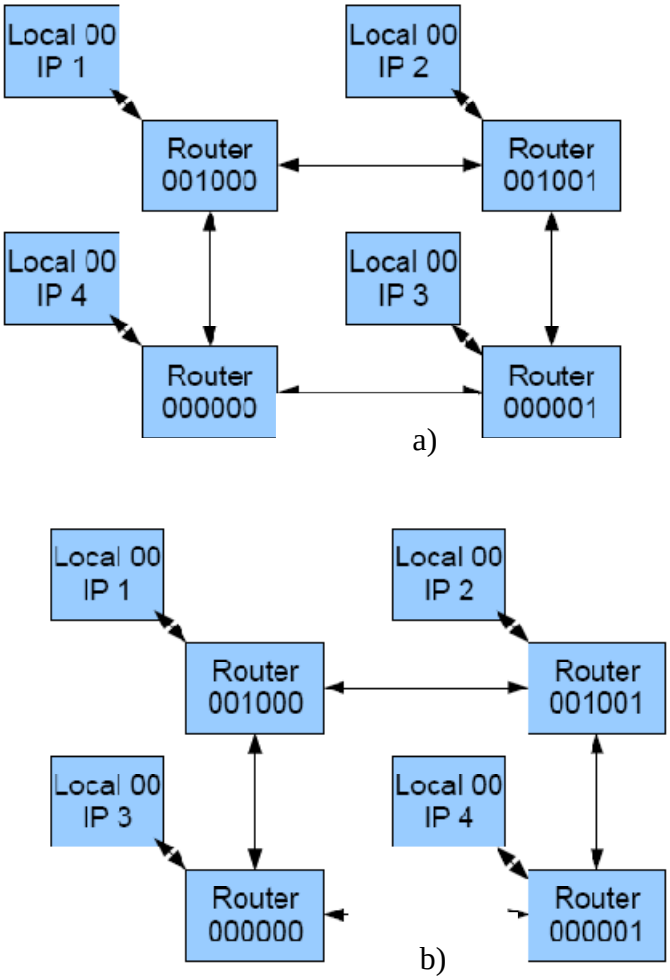


Figure 4.4: 2X2 Mesh Architecture a) Map 1 b) Map 2

4.5 Experimental Results and Analysis

We use Mentor Graphics ModelSim [21], to model IP traffic and simulate activity. All test benches wrappers have been implemented in VHDL. Results of simulations focused on overall latency in terms of cycles. Using synthesis results, latency was later obtained in terms of time. Average throughput was also calculated using total number of packets/flits sent.

4.5.1 Arbitration

Our first experiment was performed on Arbitration type. Here, a 5-port, 8-bit router was synthesized many times swapping in different output controllers. Each output controller contained a unique arbitration unit as described in Section 3.1.1.4. Table 4.10 shows the simulation results for this experiment. Combining with synthesis results, we can obtain throughput for a more accurate performance measure. Figure 4.5 presents throughput results, well Figure 4.6 re-iterates area results for accurate analysis. We can conclude that static fixed arbitration schemes are the least area expensive with very competitive latency. Static schemes can even out-perform dynamic schemes when optimized for the specific application. Dynamic schemes can be useful when QoS is the first priority, mainly preventing starvation, and that the RRA or coin passing scheme has the best metrics. This promotes the use of a flexible component library.

Table 4.10: Simulation Results for Arbitration

Arbiter Type	**Latency (cycles)**
Fixed	1476
Counter scheme 1	1452
Counter Scheme 2	1341
Counter Scheme 3	1347
Coin passing (RRA)	1266
Custom Fixed	1254
Custom Fixed Each Port	1237

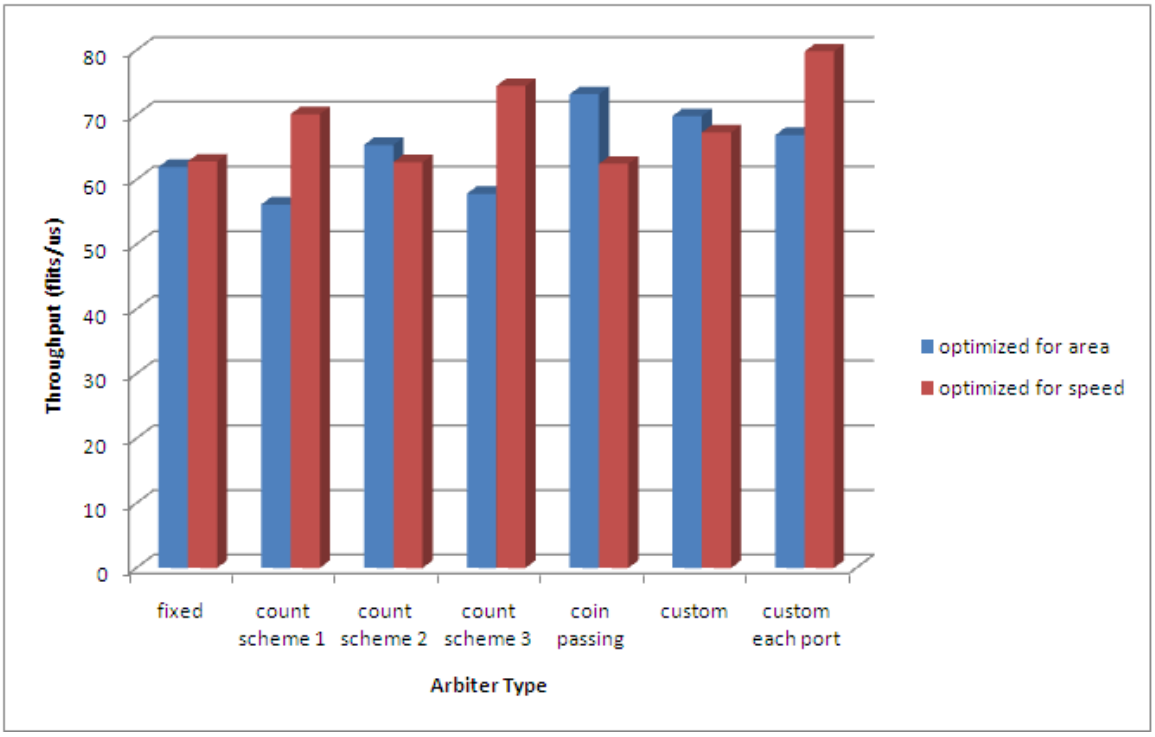


Figure 4.5: Effect of Arbiter Choice on Throughput

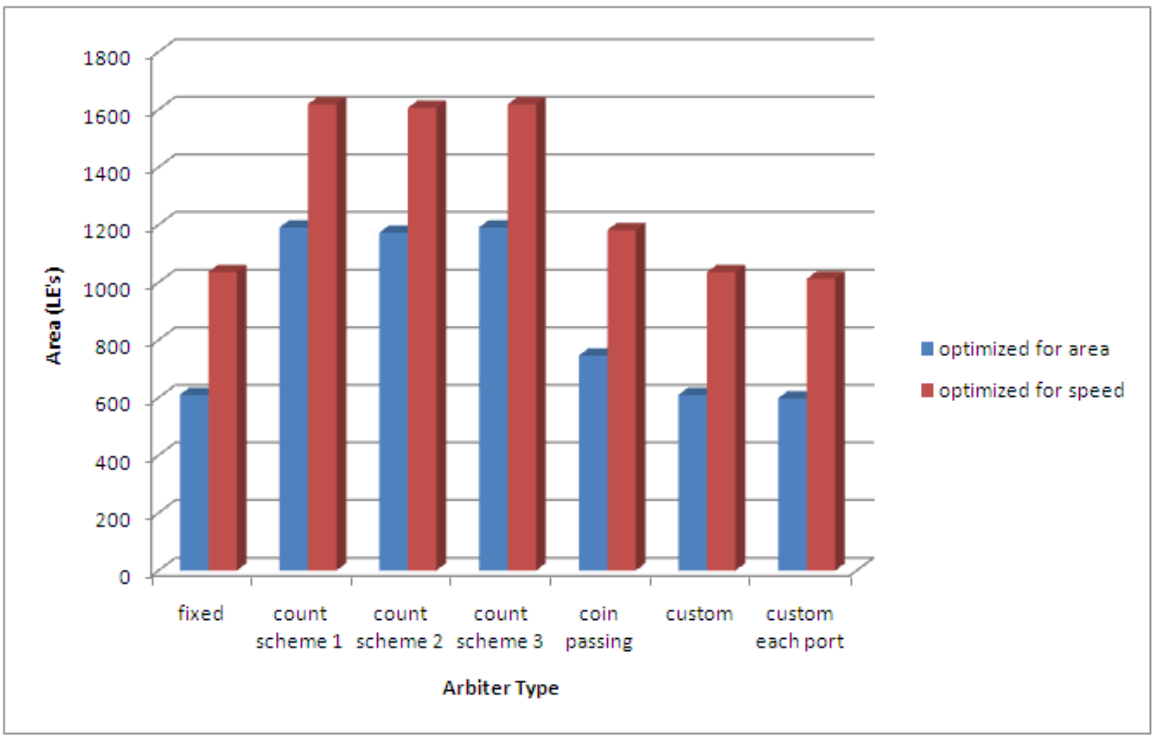


Figure 4.6: Effect of Arbiter Type on FPGA Area Utilization

4.5.2 Flit Size

Our next experiment was performed on the flit size or data path size. Here, a 5-port, 8-bit router was synthesized 4 times increasing the size of the data path each time. We were unable to synthesis designs larger do to pin restrictions among current FPGA architectures. Table 4.11 shows the simulation results for this experiment which consists of 2 tests explained in Section 4.4.2. For test 1, the number of packets stayed the same and therefore there was no effect on latency. Here, the amount of data in each packet increased with flit size. Test 2 differences were seen in latency as the amount of data was kept equal by lessening the number of packets sent. Combining with synthesis results, we can obtain throughput for a more accurate performance measure. Figures 4.7 and 4.8 present throughput results, while Figure 4.9 re-iterates area results for accurate analysis. We conclude that larger flit sizes lead to more area expensive components, although throughput is dramatically increased. With this parameter, we would recommend designers to make the flits size as large as possible subject to their area constraints. Designs optimized for speed are much too large with little to no gain in throughput.

Table 4.11: Simulation Results for Flit Size

Flit Size	**Latency (cycles)**
All flit sizes for test 1	4296
8	4296
16	2136
32	1136
64	556

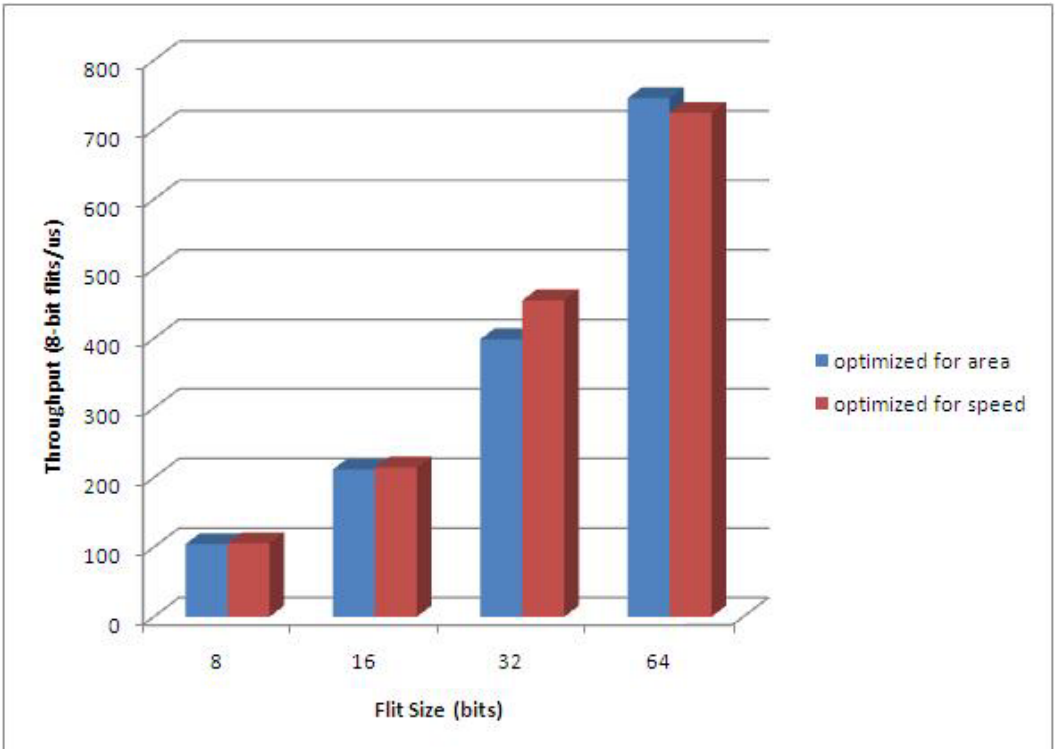


Figure 4.7: Effect of Flit Size on Throughput for Test 1

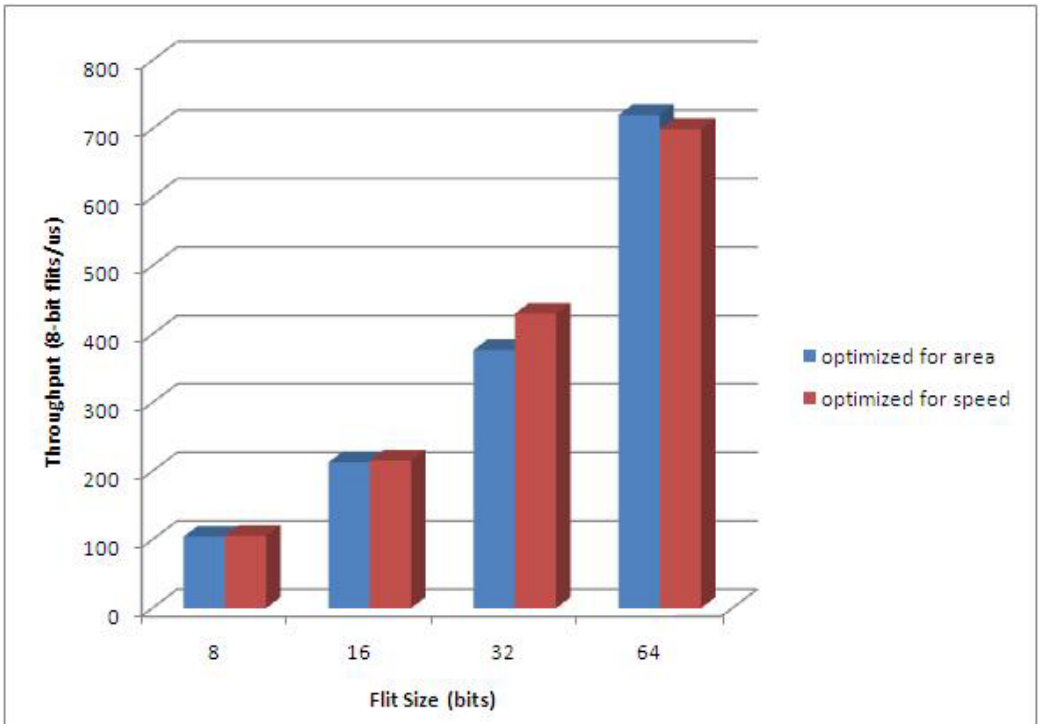


Figure 4.8: Effect of Flit Size on Throughput for Test 2

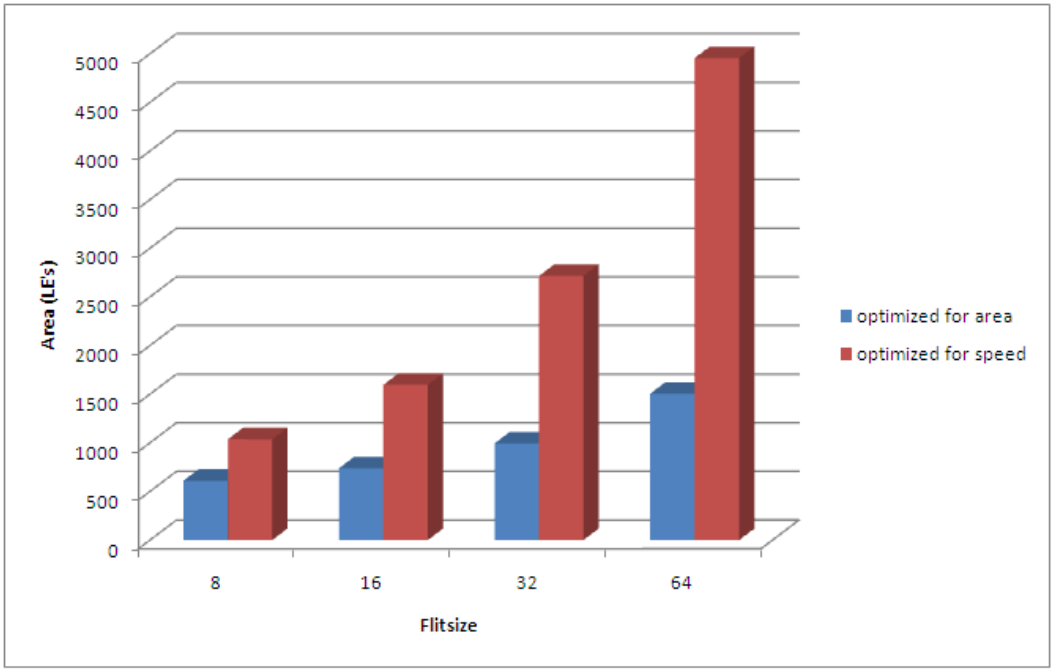


Figure 4.9: Effect on Flit Size on FPGA Area Utilization

4.5.3 Configuration

Our last experiment was performed on the configuration. Here, multiple versions of our 8-bit router were synthesized in many different mesh sizes by varying the number of local ports. During simulation, different mappings were also tested. Table 4.13 shows the simulation results for this experiment. Combining with synthesis results, we can obtain throughput for a more accurate performance measure. Figures 4.10 and 4.11 present throughput results, while Figure 4.9 re-iterates area results for accurate analysis. We conclude that NoC network topologies with a smaller number of routers are less area intensive, and provide better throughput making them superior. From the 1x2 extended configuration, we are able to see the real benefit to MLPR. For our router with SAF flow control, connecting the main processing IP core to multiple routers has an incredible impact. Interesting future work could involve testing this theory against other flow control protocols. With the flit size kept low, optimizations involving speed could turn out beneficial. Different module mapping shows how designers must not overlook this final stage in the design process.

Table 4.12: Simulation Results for Configuration

Configuration	Latency (cycles)
Single Router	3837
1x2 Mesh Map 1	4112
1x2 Mesh Map 2	4017
2x2 Mesh Map 1	4210
2x2 Mesh Map 2	4065
1x2 Mesh Extended	2126

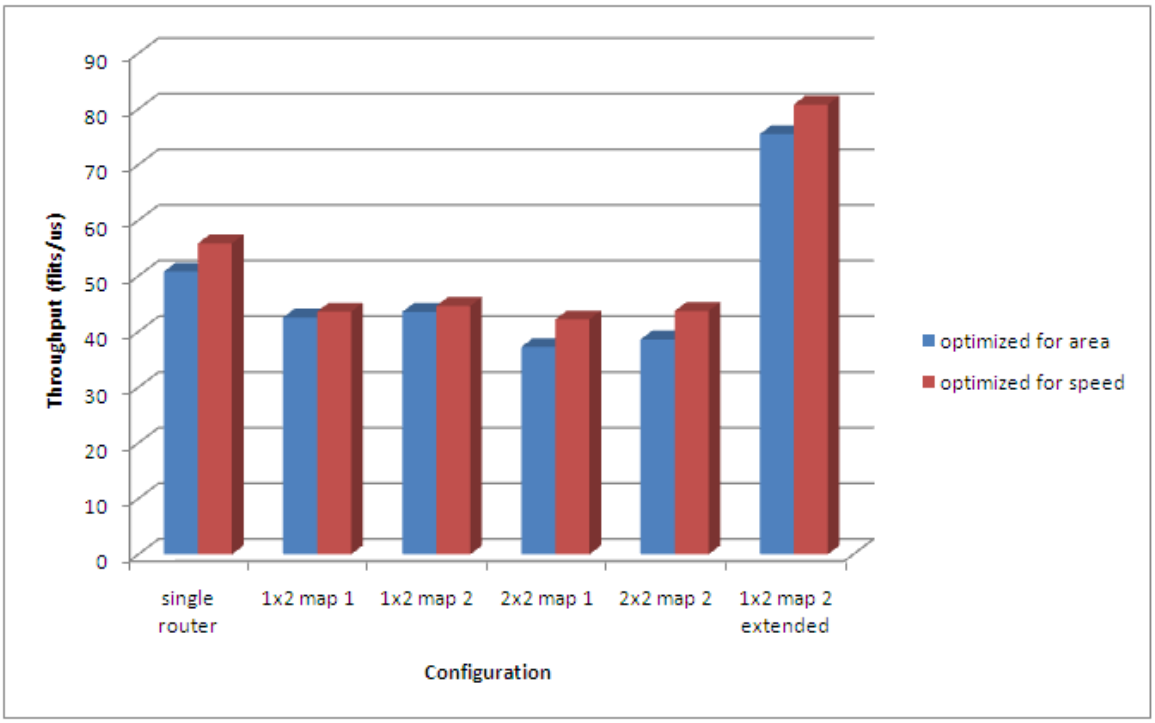


Figure 4.10: Effect of Configuration on Throughput

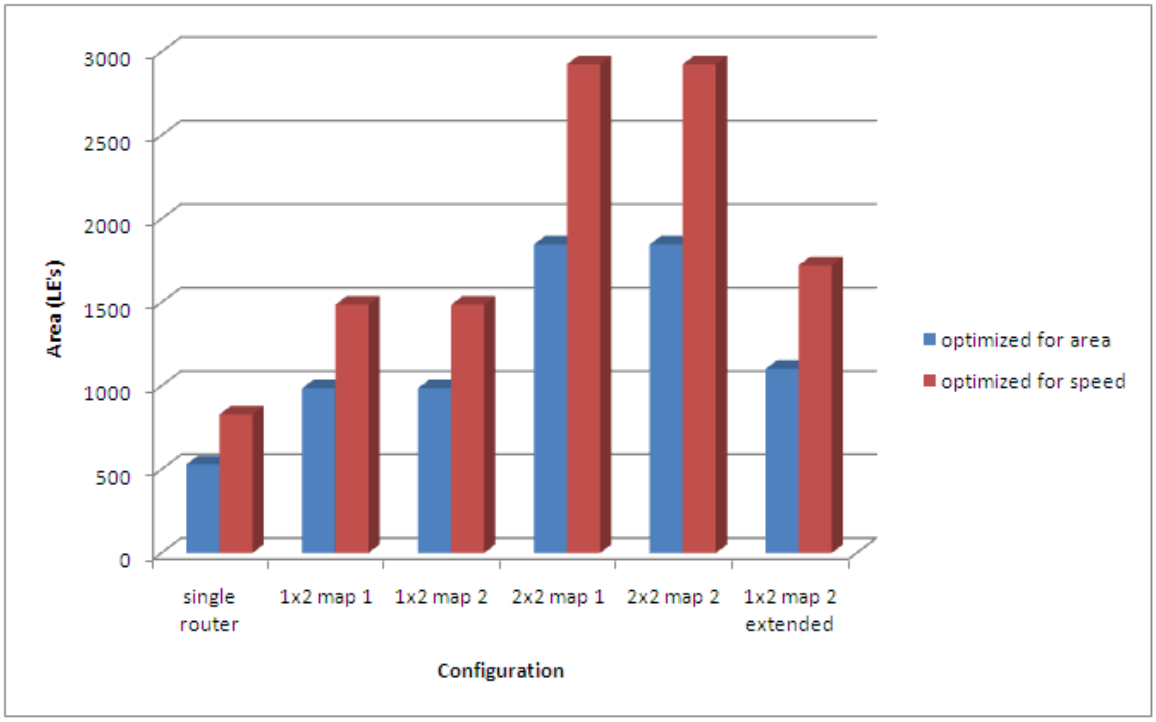


Figure 4.11: Effect of Configuration on FPGA Area Utilization

4.6 Summary

In this chapter, we briefly present the design methodology for NoC implementation. This is followed by a detailed description of the synthesis results for the proposed router under various parameter tests. Next, we discuss the details of the difficulties in evaluating NoC router performance. The experimental framework for all experiments is discussed before presenting simulation results. Finally, synthesis and simulation results are then analyzed to provide insight on parameter selection to future designers of NoC-based systems. We make case for the use of fixed arbitration, especially on design with serious area restrictions. We show the significance flit size plays on both area and throughput, making recommendations for routers with larger channel widths. We conclude our analysis with the importance of optimizing the system through use of multi-local ports, especially for multiple router connections.

The next chapter concludes this thesis by providing a summary of the research contributions and a discussion of future work.

Chapter 5

Conclusions and Future Work

The design of computer hardware is entering a new phase. Typically, designers focused on the computation aspect and simply used ad hoc mixture of direct links and buses as communication architecture. As the number of transistors that can be placed on a single chip increases, designers are forced to concentrate on communication aspect as well. It is evident that a new design methodology is required, with the adoption of NoC being the prime candidate for future SoC systems.

NoCs are seen as a solution to provide concurrent transactions among IP cores, leading to higher performance at reasonable area costs. NoCs re-useable architecture allows designers to once again concentrate on the computation aspect. Although future applications of NoC are still unknown, their flexibility provides vast potential. Recent practical evaluations of NoCs versus standard communication architectures and significant amounts of theoretical work, points to the need for future research in this area. Importance must be placed on the evaluation of standard parameters to make proper design choices.

This thesis explored the design of an NoC router for FPGA implementation. We compared metrics such as area utilization, routing resource utilization, and speed for various router parameters. Using literature review, we constrain certain parameters to prune a vast design space, to make our research feasible. In Chapter 3, the implementation of NoC protocols and design of router components was discussed in

detail. In Chapter 4, experimental evaluation results for different values of router parameters were presented and analyzed.

5.1 Summary of Research Contributions

The following contributions were made over the course of this research:

1. A preliminary case study was conducted in which the feasibility of designing and testing an NoC router for FPGA implementation was investigated.
2. We succeeded in creating a NoC router platform in VHDL with flexible parameters such as number of local ports, channel size, and arbitration type. The simple mesh topology can significantly reduce network complexity while still providing reasonable area utilization and reduced data latency. The implementation of a packet-switched protocol allows for parameter flexibility, low complexity of network control, high degree of scalability.
3. Multiple experiments were conducted that evaluated and compared the area utilization and throughput of a NoC-based system using different NoC router parameters. The results will be useful to future designers of NoC-based systems to help optimize NoC router design.

5.2 Future Work

Through the progression of this research, many interesting topics continue to surface during the development of the NoC router. Because of time constraints, these topics are out of the scope of this research but they can provide an excellent opportunity for future work to further the design space exploration of NoC. Follow-up research can use the router components that were developed, for implementing and evaluating different NoC architectures.

First, our design of output buffers to prevent HOL blocking, slowed the system. A possible path for bypassing of the output buffers could be created for cases when the next router or IP core is ready and waiting.

Second, many parameters are yet to be thoroughly explored. As the component area usage indicated, buffers occupy the largest area in NoC routers. Different flow

control modes, such as VCT and WH should be evaluated. Although XY routing provides low complexity, it makes no attempt to avoid blocked or busy routers. New routing algorithms should be explored, as there is a current lack of research results on this topic. Also, the parameters we explored were subject to specific communication protocols. Further research is needed in exploring these parameters with other communication protocols.

Third, in order to produce more accurate results, the parameter exploration experiments must be done using real world benchmarks/applications.

Fourth, a CAD tool could be developed to synthesize different variants of the proposed NoC router, based on specific values of parameters.

Finally, increased theoretical research in NoC systems has shed light on their potential. These future router designs need standardized methods of evaluation to allow comparisons between existing router designs. This leads to the need for commonly accepted benchmark applications or traffic generators to allow researchers to spend more time on the design process and less time preparing experiments.

In our lab, research is being done on a network interface for our router to connect to a standard IP core running protocols such as Wishbone. This will allow for practical testing of applications to further analyze the NoC router parameters.

.

Appendix A

Detailed Synthesis Results

These are the synthesis results from our 5-port, 8-bit router using fixed arbitration. It contains the closest comparable configuration to related work. Results are obtained from Altera Quartus II CAD tool. This is a copy of the fitter's resource usage chart.

Total logic elements 598 / 41,250 (1 %)
-- Combinational with no register 300
-- Register only 40
-- Combinational with a register258

Logic element usage by number of LUT inputs
-- 4 input functions 307
-- 3 input functions 147
-- 2 input functions 93
-- 1 input functions 31
-- 0 input functions 20

Logic elements by mode
-- normal mode 574
-- arithmetic mode 24
-- qfbk mode 8

-- register cascade mode 0

-- synchronous clear/load mode 51

-- asynchronous clear/load mode 0

Total LABs 91 / 4,125 (2 %)

Logic elements in carry chains 32

User inserted logic elements 0

Virtual pins 0

I/O pins 102 / 831 (12 %)

-- Clock pins 7 / 20 (35 %)

Global signals 4

M512s 8 / 384 (2 %)

M4Ks 0 / 183 (0 %)

M-RAMs 0 / 4 (0 %)

Total memory bits 320 / 3,423,744 (< 1 %)

Total RAM block bits 4,608 / 3,423,744 (< 1 %)

DSP block 9-bit elements 0 / 112 (0 %)

Global clocks 4 / 16 (25 %)

Regional clocks 0 / 16 (0 %)

Fast regional clocks 0 / 32 (0 %)

SERDES transmitters 0 / 90 (0 %)

SERDES receivers 0 / 90 (0 %)

Maximum fan-out node CLK

Maximum fan-out 306

Total fan-out 2488

Average fan-out 3.51

These are the synthesis results for the configuration test showing an in-depth look at routing resource information for each configuration. They are presented in order of single router, 1x2 mesh, 1x2 extended mesh, and 2x2 mesh. Once again, results are obtained from Altera Quartus II CAD tool. This is a copy of the fitter's routing resource chart.

Interconnect Resource Type / Usage for Single Router

C16 interconnects 148 / 7,039 (2 %)

C4 interconnects 338 / 109,820 (< 1 %)

C8 interconnects 98 / 24,220 (< 1 %)

DIFFIOCLKs 0 / 32 (0 %)

DQS bus muxes 0 / 76 (0 %)

DQS-16 I/O buses 0 / 8 (0 %)

DQS-32 I/O buses 0 / 4 (0 %)

DQS-8 I/O buses 0 / 20 (0 %)

Direct links 167 / 163,680 (< 1 %)

Fast regional clocks 0 / 32 (0 %)

Global clocks 4 / 16 (25 %)

I/O buses 8 / 404 (1 %)

LUT chains 30 / 37,125 (< 1 %)

Local routing interconnects 405 / 41,250 (< 1 %)

R24 interconnects 42 / 7,259 (< 1 %)

R4 interconnects 419 / 222,840 (< 1 %)

R8 interconnects 99 / 36,138 (< 1 %)

Regional clocks 0 / 16 (0 %)

Interconnect Resource Type / Usage for 1x2 Mesh

C16 interconnects 96 / 7,039 (1 %)

C4 interconnects 521 / 109,820 (< 1 %)

C8 interconnects 182 / 24,220 (< 1 %)

DIFFIOCLKs 0 / 32 (0 %)

DQS bus muxes 0 / 76 (0 %)

DQS-16 I/O buses 0 / 8 (0 %)

DQS-32 I/O buses 0 / 4 (0 %)

DQS-8 I/O buses 0 / 20 (0 %)

Direct links 295 / 163,680 (< 1 %)

Fast regional clocks 0 / 32 (0 %)

Global clocks 6 / 16 (37 %)

I/O buses 1 / 404 (< 1 %)

LUT chains 37 / 37,125 (< 1 %)

Local routing interconnects 687 / 41,250 (1 %)

R24 interconnects 32 / 7,259 (< 1 %)

R4 interconnects 515 / 222,840 (< 1 %)

R8 interconnects 108 / 36,138 (< 1 %)

Regional clocks 0 / 16 (0 %)

Interconnect Resource Type / Usage for 1x2 extended Mesh

C16 interconnects 127 / 7,039 (1 %)

C4 interconnects 600 / 109,820 (< 1 %)

C8 interconnects 151 / 24,220 (< 1 %)

DIFFIOCLKs 0 / 32 (0 %)

DQS bus muxes 0 / 76 (0 %)

DQS-16 I/O buses 0 / 8 (0 %)

DQS-32 I/O buses 0 / 4 (0 %)

DQS-8 I/O buses 0 / 20 (0 %)

Direct links 323 / 163,680 (< 1 %)

Fast regional clocks 0 / 32 (0 %)

Global clocks 7 / 16 (43 %)

I/O buses 8 / 404 (1 %)

LUT chains 47 / 37,125 (< 1 %)

Local routing interconnects 777 / 41,250 (1 %)

R24 interconnects 47 / 7,259 (< 1 %)

R4 interconnects 658 / 222,840 (< 1 %)

R8 interconnects 146 / 36,138 (< 1 %)

Regional clocks 0 / 16 (0 %)

Interconnect Resource Type / Usage for 2x2 Mesh

C16 interconnects 113 / 7,039 (1 %)

C4 interconnects 1,067 / 109,820 (< 1 %)

C8 interconnects 334 / 24,220 (1 %)

DIFFIOCLKs 0 / 32 (0 %)

DQS bus muxes 0 / 76 (0 %)

DQS-16 I/O buses 0 / 8 (0 %)

DQS-32 I/O buses 0 / 4 (0 %)

DQS-8 I/O buses 0 / 20 (0 %)

Direct links 486 / 163,680 (< 1 %)

Fast regional clocks 0 / 32 (0 %)

Global clocks 12 / 16 (75 %)

I/O buses 1 / 404 (< 1 %)

LUT chains 69 / 37,125 (< 1 %)

Local routing interconnects 1,291 / 41,250 (3 %)

R24 interconnects 40 / 7,259 (< 1 %)

R4 interconnects 1,022 / 222,840 (< 1 %)

R8 interconnects 196 / 36,138 (< 1 %)

Regional clocks 0 / 16 (0 %)

References

[1] W. Dally, B. Towles, “Route packets, not wires: On-chip interconnection networks,” IEEE, Proc. of Design Automation Conference, 2001, Page(s): 684-689.

[2] R. Marculescu et al, “Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives,” IEEE Trans. on CAD of ICs and Systems, vol. 28, no. 1, January 2009.

[3] M. Saldana, L. Shannon, P. Chow, "The routability of multiprocessor network topologies in FPGAs," ACM, Proc. of International Workshop on System-Level Interconnect Prediction, 2006, Pages: 49-56.

[4] Narasimhan and O. Kumaravelu and R. Sridhar, "An investigation of the impact of network parameters on performance of network-on-chips," IEEE, Proc. of Circuits and Systems, Aug. 2005, Page(s): 1617-1620.

[5] T. Bjerregaard, S. Mahadevan, "A survey of research and practices of Network-on-chip," ACM, Proc. of Computing Surveys, 2006, Issue 1.

[6] T. Marescaux, T. Bartic, D. Verkest, S. Vernalde, R. Lauwereins, "Interconnection Networks Enable Fine -Grain Dynamic Multi-Tasking on FPGAs," ACM, Proc. of Field-Programmable Logic and Applications, Sep. 2002, Page(s): 795-805.

[7] T. Marescaux, T. Bartic, D. Verkest, S. Vernalde, R. Lauwereins, Mignolet, J.-Y.; Nollet, V.; "Highly scalable network on chip for reconfigurable systems," IEEE, Proc. of International Symposium on System-on-Chip, Nov. 2003, Page(s): 79-82.

[8] F. Moraes, A. Mello, L. Möller, L. Ost, N. Calazans, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," ACM, Integration -the VLSI Journal, 2004, Volume 38, no. 1, Page(s) 69-93.

[9] L. Ost, A.Mello, J. Palma, F. Moraes, N. Calazans, "MAIA -a framework for networks on chip generation and verification," IEEE, Proc. of Design Automation Conference, Jan. 2005, Page(s): 49-52.

[10] C. A. Zeferino, M.E. Kreutz, A.A Susin, “RASoC: a router soft-core for networks-on-chip,” IEEE, Proc. of Design Automation and Test in Europe Conference, Feb. 2004, Page(s): 198-203.

[11] C. A. Zeferino, A.A Susin, "SoCIN: a parametric and scalable network-on-chip," IEEE, Proc. of Integrated Circuits and Systems Design, Sept. 2003, Page(s): 169-174.

[12] H. Freitas, D. Colombo, F. Kastensmidt, P. Navaux, "Evaluating Network-on-Chip for Homogeneous Embedded Multiprocessors in FPGAs," IEEE, Proc. of International Symposium on Circuits and Systems, May 2007, Page(s): 3776-3779.

[13] C. Hilton and B. Nelson, "PNoC: a flexible circuit-switched NoC for FPGA-based systems," IEEE, Proc. of Computers and Digital Techniques, May 2005, Page(s): 181-188.

[14] B. Sethuraman and P. Bhattacharya and J. Khan and R. Vemuri, "LiPaR: A light-weight parallel router for FPGA-based networks-on-chip," ACM, Proc. of Great Lakes Symposium on VLSI, 2005, Page(s): 452-457.

[15] B. Sethuraman, "Novel Methodologies for Performance & Power Efficient Reconfigurable Networks-on-Chip," IEEE, Proc. of International Conference on Field Programmable Logic and Applications, 2006, Page(s): 1-2.

[16] B. Sethuraman and R. Vemuri, “OptiMap: a tool for automated generation of NoC architectures using multi-port routers for FPGAs," IEEE, Proc. of Council on Electronic Design Automation, March 2006, Page(s) 6-11.

[17] M. P. Vestias, H. C. Neto, "Area and performance optimization of a generic network-on-chip architecture," ACM, Proc. of Integrated circuits and systems design, 2006, Page(s) 68-73.

[18] Janarthanan, V. Swaminathan, K. A. Tomko, “MoCReS: an Area-Efficient Multi-Clock On-Chip Network for Reconfigurable Systems,” IEEE, Proc. of Symposium on Computer Society, March 2007, Page(s): 455-456.

[19] Ling Wang, Jianye Hao, Feixuan Wang, "Bus-Based and NoC Infrastructure Performance Emulation and Comparison," IEEE, Proc. of International Conference on Information Technology: New Generations, 2009, Page(s): 855-858.

[20] Altera Corp. http://www.altera.com last accessed, September 2009.

[21] MentorGraphics Inc. http://mentorgraphics.com last accessed, September 2009.

[22] T. Le, "Implementation and Evaluation of an NoC Architecture for FPGAs," M.S. Thesis, University of Windsor, 2009.

[23] EIT, NoC slides, http://www.eit.lth.se/fileadmin/eit/courses/eti135/slides/NoC.pdf last accessed, September 2009.

[24] Xilinx Inc. http://www.xilinx.com.

[25] Article from embedded development site, http://www.embedded.com/showArticle.jhtml?articleID=166403161?_requestid=134990 last accessed, September 2009.

[26] Article from electronics engineering site, http://www.element-14.com/community/docs/DOC-12235;jsessionid=0792E50BDA4C9090757AAB360A8C05DF last accessed, September 2009.

[27] U. Ogras, R. Marculescu, "Analytical Router Modeling for Networks-on-Chip Performance Analysis", IEEE, Proc. of Design, Automation, and Test in Europe, April 2007, Page(s): 1-6.

[28] Integrated Microsystems Group, University of Victoria, http://www.ims.ece.uvic.ca last accessed, September 2009.

[29] Z. Lu, R. Thid, M. Millberg, E. Nilsson, A. Jantsch. "NNSE: Nostrum network-on-chip simulation environment," Proc. of Design Automation and Test in Europe, March 2005

[30] Department for Electronics, Computer and Software Systems at KTH, Stockholm, http://www.ict.kth.se/nostrum last accessed, September 2009.

[31] S. Kubisch, H. Widiger, C. Cornelius, D. Timmermann, A. Strzeletz, "E-Core – A Configurable IP Core for Application-specific NoC Performance Evaluation," Proc. of Design Automation and Test in Europe, Workshop on Diagnostic Services in Network-on-Chips, Nice, France, April 2007.

VITA AUCTORIS

Mike Brugge was born in Windsor, Ontario, Canada in August 1985. He received his B.A.Sc. degree in electrical engineering in 2007 from the University of Windsor in Windsor, Ontario, Canada. He is currently a candidate in the electrical and computer engineering M.A.Sc. program, at the University of Windsor. His research interests include field programmable-related technologies, hardware and software development for embedded system, and digital computing.