



**DESIGN OF A RECONFIGURABLE NETWORK ON
CHIP FOR NEXT GENERATION FPGA USING
DYNAMIC PARTIAL RECONFIGURATION**

By

Ramy Ahmed Ali Mohamed

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

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2019

**DESIGN OF A RECONFIGURABLE NETWORK ON
CHIP FOR NEXT GENERATION FPGA USING
DYNAMIC PARTIAL RECONFIGURATION**

By

Ramy Ahmed Ali Mohamed

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

Under the Supervision of

Prof. Ahmed Hussein Mohamed

Dr. Hassan Mostafa Hassan

Professor

Assistant Professor

Electronics and Communications
Engineering Department
Faculty of Engineering, Cairo University

Electronics and Communications
Engineering Department
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2019

**DESIGN OF A RECONFIGURABLE NETWORK ON
CHIP FOR NEXT GENERATION FPGA USING
DYNAMIC PARTIAL RECONFIGURATION**

By

Ramy Ahmed Ali Mohamed

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

Approved by the Examining Committee:

Prof. Ahmed Hussein Mohamed (Thesis Main Advisor)
Electronics Professor, Faculty of Engineering, Cairo University

Prof. Amin Mohamed Nassar (Internal Examiner)
Electronics Professor, Faculty of Engineering, Cairo University

Dr. Amr Talaat Abdel Hamid (External Examiner)
Associate Professor, Electronics/Networks Departments, German
University in Cairo

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT

2019

Engineer's Name: Ramy Ahmed Ali Mohamed
Date of Birth: 23/06/1989
Nationality: Egyptian
E-mail: ramy.ahmed.ali@gmail.com
Phone: +201007994483
Address: 17 Building, 13B street, Maadi,
Cairo, Egypt
Registration Date: 1/10/2012
Awarding Date: / /2019
Degree: **Master of Science**
Department: **Electronics and Communications Engineering**



Supervisors:
Prof. Dr. Ahmed Hussein Mohamed
Dr. Hassan Mostafa Hassan

Examiners:
Prof. Dr. Ahmed Hussein Mohamed (Thesis main advisor)
Electronics Professor, Faculty of Engineering,
Cairo University
Prof. Amin Mohamed Nassar (Internal examiner)
Electronics Professor, Faculty of Engineering,
Cairo University
Dr. Amr Talaat Abdel-Hameed (External examiner)
Associate Professor, Electronics/Networks Departments,
German University in Cairo

Title of Thesis:

Design of a Reconfigurable Network on Chip for next generation FPGA using Dynamic Partial Reconfiguration

Key Words:

Partial Dynamic Reconfiguration; Network on Chip; FPGA

Summary:

The main goal of this thesis is to present the runtime configurability support to CONNECT Network-on-Chip (NoC). Additionally, the thesis studies the reconfigurability impact on the network performance with its different configuration parameters. In comparison with the fixed NoCs, the runtime configurable NoCs save area by reusing a part of the network when it is not required during runtime. A reconfiguration tool is developed helping the user to decide the optimum network structure for every used benchmark.

Disclaimer

I hereby declare that this thesis is my own original work and that no part of it has been submitted for a degree qualification at any other university or institute.

I further declare that I have appropriately acknowledged all sources used and have cited them in the references Section.

Name: Ramy Ahmed Ali Mohamed

Date:

Signature:

Acknowledgments

I would like to thank Dr. Hassan Mostafa and Prof. Dr. Ahmed Hussein a lot for the opportunity to work on such topic under their supervision and allowing me to add to my skills and experience. Moreover, this work would not have been possible without the flexible and healthy work environment offered by my manager Haytham Ashour. Additionally, I would like to thank my teammate Ahmed Kamal for his continuous encouragement and patience. Furthermore, I would like to express my sincere thanks and gratitude to my workmate Ahmed Akl who helped me a lot with his knowledge. And, I would like to thank my workmate Maha Beheiry who helped in the thesis work review. Furthermore, a lot of thanks goes to my fiancée Samah Mamdouh who provided very helpful comments and has not stopped supporting me till this work is finished. Finally, I would like to thank my family members for their lifetime support.

Table of Contents

DISCLAIMER	I
ACKNOWLEDGMENTS	II
TABLE OF CONTENTS	III
LIST OF TABLES	V
LIST OF FIGURES	VI
NOMENCLATURE	IX
ABSTRACT	X
CHAPTER 1 : INTRODUCTION	1
1.1. MOTIVATION	1
1.2. CONTRIBUTION	2
1.3. ORGANIZATION OF THE THESIS	2
CHAPTER 2 : LITERATURE REVIEW	3
2.1. INTRODUCTION	3
2.2. FPGA ARCHITECTURE AND CAPABILITIES	3
2.2.1. FPGA over ASIC.....	3
2.2.2. FPGA structure	4
2.3. DYNAMIC PARTIAL RECONFIGURATION (DPR)	5
2.3.1. SRAM cells configuration topologies	5
2.3.2. Multi context configuration	7
2.3.3. Dynamic Partial Reconfiguration Advantages	7
2.4. NETWORK ON CHIPS	8
2.4.1. NoC parameters	8
2.5. CHALLENGES OF RECONFIGURABLE NOCs	11
2.5.1. Previous efforts.....	11
2.5.2. DyNoC.....	12
CHAPTER 3 : PERFORMANCE EVALUATION OF DYNAMIC PARTIAL RECONFIGURATION TECHNIQUES FOR SDR ON FPGA	19
3.1. XILINX FPGA CONFIGURATION TECHNIQUES	20
3.1.1. JTAG	20
3.1.2. Serial Mode	21
3.1.3. SelectMAP.....	21
3.1.4. ICAP	22
3.2. SOFTWARE DEFINED RADIO	23
3.3. RESULTS AND DISCUSSION	24
CHAPTER 4 : APPLYING DYNAMIC RECONFIGURATION TO CONNECT NOC	29

4.1.	CONNECT NoC ARCHITECTURE	29
4.2.	RECONFIGURATION CHANGES TO CONNECT	31
4.3.	TEST ENVIRONMENT STRUCTURE	34
4.4.	RECONFIGURATION TOOL	35
CHAPTER 5 : IMPACT OF DYNAMIC RECONFIGURATION ON NETWORK ON CHIP PERFORMANCE		39
5.1.	ENVIRONMENT SETUP.....	39
5.2.	NETWORK TOPOLOGY DPR EVALUATION.....	39
5.3.	BUFFER DEPTH DPR EVALUATION.....	46
5.4.	VIRTUAL CHANNEL DPR EVALUATION.....	52
5.5.	BUFFER DEPTH VS VIRTUAL CHANNEL.....	58
5.6.	FLOW CONTROL DPR EVALUATION	62
5.7.	AREA EVALUATION	66
5.7.1.	Case Study	66
5.8.	DESIGN RECOMMENDATIONS	69
CONCLUSION.....		70
FUTURE WORK		71
REFERENCES		72
APPENDIX A: RECONFIGURATION TOOL USER MANUAL		75
APPENDIX B: RECONFIGURATION TOOL SOURCE CODE		79

List of Tables

Table 3.1: Reconfiguration speed for different Reconfiguration techniques [28].....	24
Table 4.1: Reconfiguration tool output for different user benchmarks	35
Table 5.1: Estimated Virtex 5 Xilinx FPGA resource area [26, 27, 33]	66
Table 5.2: Case study benchmarks with Static NoC and Reconfigurable NoC approaches	66
Table 5.3: Area resources for different networks corresponding to different benchmarks	68

List of Figures

Figure 2.1: FPGA versus ASIC with respect to Cost and Number of units [1]	3
Figure 2.2: Mesh-based FPGA internal structure [2]	4
Figure 2.3: SRAM configuration topologies [3]	6
Figure 2.4: Tabula FPGA internal structure [4]	7
Figure 2.5: Dynamic Partial Reconfiguration illustration [7]	8
Figure 2.6: Star, Mesh, and Ring network topologies [8]	9
Figure 2.7: Credit based flow control structure [9]	10
Figure 2.8: Feasible (a) and infeasible (b) component placement [11]	13
Figure 2.9: Obstacle surrounding in the horizontal (a) and vertical (b) directions [12].	15
Figure 2.10: Placement leading to extremely long routing path (a); Router guiding in the DyNoc (b) [12]	17
Figure 2.11: Merging bit-stream for a 2D partial reconfiguration on Xilinx FPGAs [12]	18
Figure 3.1: Reconfiguration techniques of convolutional encoder inside communication chain	20
Figure 3.2: SelectMAP structure in Xilinx Virtex 5 FPGAs [24]	22
Figure 3.3: 8-bit ICAP and SelectMAP with Serial mode and JTAG [28]	25
Figure 3.4: 16-bit ICAP and SelectMAP with Serial mode and JTAG [28]	26
Figure 3.5: 32-bit ICAP and SelectMAP with Serial mode and JTAG [28]	27
Figure 4.1: CONNECT NoC RTL configurator [8]	30
Figure 4.2: CONNECT NoC router core internal structure	31
Figure 4.3: Reconfigurable CONNECT NoC structure	32
Figure 4.4: Reconfigurable examples of CONNECT NoC during runtime [29]	33
Figure 4.5: Environment structure of CONNECT NoC during runtime [29]	34
Figure 4.6: Reconfiguration tool structure [29]	35
Figure 4.7: Reconfiguration tool interface in Batch mode	36
Figure 4.8: Reconfiguration tool interface in GUI mode	37
Figure 4.9: Reconfiguration tool output in Batch and GUI modes	37
Figure 5.1: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=2 – BD=4	40
Figure 5.2: CONNECT Performance all PDR configurations – Credit-based flow control – VC=4 – BD=4	41
Figure 5.3: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=8	41
Figure 5.4: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=16	42
Figure 5.5: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=32	42
Figure 5.6: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=64	43
Figure 5.7: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=4	43
Figure 5.8: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=8	44
Figure 5.9: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=16	44

Figure 5.10: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=32	45
Figure 5.11: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=64	45
Figure 5.12: CONNECT throughput of all BD configurations – Mesh 4x3 – Credit-based flow control – VC=2	46
Figure 5.13: CONNECT throughput of all BD configurations – Mesh 2x2 – Credit-based flow control – VC=2	47
Figure 5.14: CONNECT throughput of all BD configurations – Mesh 2x1 – Credit-based flow control – VC=2	48
Figure 5.15: CONNECT throughput of all BD configurations – Mesh 4x4 – Peek-based flow control – VC=2	48
Figure 5.16: CONNECT throughput of all BD configurations – Mesh 4x3 – Peek-based flow control – VC=2	49
Figure 5.17: CONNECT throughput of all BD configurations – Mesh 3x3 – Peek-based flow control – VC=2	49
Figure 5.18: CONNECT throughput of all BD configurations – Mesh 3x2 – Peek-based flow control – VC=2	50
Figure 5.19: CONNECT throughput of all BD configurations – Mesh 2x2 – Peek-based flow control – VC=2	50
Figure 5.20: CONNECT throughput of all BD configurations – Mesh 2x1 – Peek-based flow control – VC=2	51
Figure 5.21: CONNECT throughput of all VC configurations – Mesh 4x4 – Peek-based flow control – BD=4	52
Figure 5.22: CONNECT throughput of all VC configurations – Mesh 2x1 – Peek-based flow control – BD=4	53
Figure 5.23: CONNECT throughput of all VC configurations – Mesh 4x4 – Peek-based flow control – BD=64	53
Figure 5.24: CONNECT throughput of all VC configurations – Mesh 4x4 – Credit-based flow control – BD=8	54
Figure 5.25: CONNECT throughput of all VC configurations – Mesh 4x3 – Credit-based flow control – BD=8	55
Figure 5.26: CONNECT throughput of all VC configurations – Mesh 3x3 – Credit-based flow control – BD=8	55
Figure 5.27: CONNECT throughput of all VC configurations – Mesh 3x2 – Credit-based flow control – BD=8	56
Figure 5.28: CONNECT throughput of all VC configurations – Mesh 2x2 – Credit-based flow control – BD=8	56
Figure 5.29: CONNECT throughput of all VC configurations – Mesh 2x1 – Credit-based flow control – BD=8	57
Figure 5.30: CONNECT throughput of BD vs VC configurations – Mesh 4x4 – Peek-based flow control – VC=2, 4, and 8 – BD=4, 8, and 16	58
Figure 5.31: CONNECT throughput of BD vs VC configurations – Mesh 4x4 – Credit-based flow control – VC=2, 4, and 8 – BD=4, 8, and 16	59
Figure 5.32: CONNECT throughput of BD vs VC configurations – Mesh 3x3 – Credit-based flow control – VC=2, 4, and 8 – BD=8, 16, and 32	59
Figure 5.33: CONNECT throughput of BD vs VC configurations – Mesh 3x3 – Peek-based flow control – VC=2, 4, and 8 – BD=8, 16, and 32	60
Figure 5.34: CONNECT throughput of BD vs VC configurations – Mesh 2x2 – Credit-based flow control – VC=2, 4, and 8 – BD=16, 32, and 64	60

Figure 5.35: CONNECT throughput of BD vs VC configurations – Mesh 2x2 – Peek-based flow control – VC=2, 4, and 8 – BD=16, 32, and 64	61
Figure 5.36: CONNECT throughput of Flow Control configurations – Mesh 4x4 – VC=2 – BD=4	62
Figure 5.37: CONNECT throughput of Flow Control configurations – Mesh 2x1 – VC=2 – BD=4	63
Figure 5.38: CONNECT throughput of Flow Control configurations – Mesh 3x3 – VC=4 – BD=16	63
.....	64
Figure 5.39: CONNECT throughput of Flow Control configurations – Mesh 2x2 – VC=4 – BD=32	64
Figure 5.40: CONNECT throughput of Flow Control configurations – Mesh 4x3 – VC=8 – BD=8	64
.....	65
Figure 5.41: CONNECT throughput Flow Control configurations – Mesh 2x1 – VC=8 – BD=64	65
Figure 5.42: Virtex 5 xc5vlx110tff1136-1 Area results of reconfigurable mesh 4x4 CONNECT vs Static mesh 4x4 CONNECT	67
Figure A.1: The Reconfiguration tool required files and sheets	75
Figure A.2: The Reconfiguration tool interface and output	77

Nomenclature

NRC	Non Recurring Cost
ASIC	Application Specific Integrated Circuits
FPGA	Field-Programmable Gate Array
NoC	Network on Chip
DPR	Dynamic Partial Reconfiguration
CLB	Configurable Logic Block
SDR	Software Defined Radio
SoC	System on Chip
JTAG	Joint Test Action Group
ICAP	Internal Configuration Access Port
RTL	Register Transfer Level
BD	Buffer Depth
VC	Virtual Channel
FC	Flow Control
LUT	Look-Up Table

Abstract

With the vast increase in the design densities inside System-on-Chips (SoCs) every year, Network-on-Chip (NoC) design architecture is introduced as a reliable on-chip communication platform facing the challenges of complex design systems. NoC design approach is preferred over the conventional bus communication for its scalability, improved modularity, and better performance.

On the other hand, the advancement in dynamically reconfigurable Field Programmable Gate Arrays (FPGAs) allows the hardware designs to be reconfigured during runtime. Dynamic Partial Reconfiguration (DPR) adds more flexibility to hardware modules and offers better area utilization and more power optimization. Furthermore, using DPR permits the adaptive hardware algorithms to evolve based on the different applications.

Introducing the reconfigurability concept into one of the most ramping and trending design platforms like the NoC is considered a good opportunity for extracting the benefits out of the two concepts. The high flexibility and full customization of the reconfigurable NoC could open the door for a completely adaptive NoC that suits a large number of benchmarks according to the runtime requirements. The importance of reconfigurable NoCs appears with the designs intended to be dynamically reconfigurable. When the NoC is part of the design, its re-configurability gives the opportunity to operate with the best fit network to every user benchmark.

The ability to reconfigure SRAM-based FPGAs is the most powerful feature over Application Specific Integrated Circuit (ASIC) designs. DPR emphasizes this feature by increasing flexibility over runtime phase. Xilinx Virtex family of FPGAs provides four techniques to perform DPR; SelectMAP, Serial mode, JTAG, and ICAP. In this thesis, each technique is reviewed, evaluated, and tested using convolutional encoder module which is an essential block from Software Defined Radio (SDR) system. SDR as a system is chosen as it becomes the most promising application for DPR. Experiments are carried out using Xilinx Virtex 5 to measure the trade-offs between performance and area-overhead by adding reconfiguration controller on/off FPGA fabric. It is shown that the performance of each interface is independent of design resource. However, the performance is proportional only with partial reconfiguration region selection which had been chosen at the Place and Route phase.

The main objective of this thesis is to present the runtime configurability support to CONNECT NoC. Additionally, the thesis studies the impact of this reconfigurability on the network performance with its different configuration parameters. Runtime configurability expands the flexibility of NoCs and allows a full customization to the NoC with the dynamic reconfigurable designs. In comparison with the fixed NoCs, the runtime configurable NoCs save area by reusing a part of the network when it is not required during runtime. A reconfiguration tool is developed to assist the user in constructing the optimal network structure for every used benchmark. The reconfiguration tool requires the minimum needed throughput and the expected traffic load as inputs. The tool inputs are required to recommend the best network configuration according to the minimum area that achieves those requirements.

Chapter 1 : Introduction

This chapter highlights the aim of applying the configurability concept into modern systems based on Network on Chip architectures. A complete study to the reconfigurability impact is provided inside this thesis. The following Sections present the motivation behind this work, the contribution added to this work, and the thesis structure and organization.

1.1. Motivation

Design area and throughput are among the most important metrics that need to be considered while planning an architecture for a SoC. With the complexity of designs, NoC design architecture appeared as an optimal candidate for an on-chip platform that can be customized according to the application requirements. NoC design approach is preferred over the conventional bus communication for its scalability, improved modularity, and better performance.

On the other hand, introducing the reconfigurability concept into the FPGAs unlocks a lot of capabilities in the hardware designs leading to a full hardware customization during runtime. The added flexibility to the hardware by the DPR offers better area usage and more power optimization. In addition, using DPR allows the adaptive hardware algorithms to evolve based on the different applications.

Studying the reconfigurability techniques offered by FPGA providers (Xilinx and Altera) would aid in analyzing the strength and weakness of the different methods with the different design sizes. Besides, taking into consideration the reconfigurability constraints could help in the design phase of any SoC. This maximizes the gain earned from targeting reconfigurable architectures.

Introducing the DPR capability to the NoC creates new opportunities for customizing network topology according to the system requirements during the runtime. Self-adaptive NoCs during runtime are beneficial when used with configurable hardware design. The configurable hardware has multiple benchmarks and different performance requirements, and this flexibility is absent in the fixed NoCs.

1.2. Contribution

This work includes the following contributions:

- Presenting a complete review and a comparison for the different DPR techniques in Xilinx FPGAs including serial and parallel methods. Then, providing a complete evaluation for the Serial techniques (JTAG and Slave Serial mode) against the parallel techniques (SelectMAP and ICAP).
- Studying the CONNECT NoC with all its capabilities and how they are implemented. Then, integrating the RTL into a test environment for evaluation.
- Introducing the Configurability to the CONNECT NoC Register Transfer Level (RTL) to be fully adaptive during runtime. The flexibility is accompanied with switching into a certain topology and updating all the routing tables for creating alternative paths to the routed packets.
- Implementing a configuration tool for analyzing the different benchmarks' requirements. The tool is responsible for choosing the most suitable network topology according to the desired performance. The selection criteria is based on the minimum area and power overhead of the different user benchmarks.
- Providing a complete study for the impact of reconfigurability on the NoC different parameters. The study includes how those parameters can emphasize the value of the dynamically adaptive network

1.3. Organization of the thesis

The remainder of this thesis organized as follows. Chapter 2 provides a detailed survey of the DPR and NoCs with FPGA. Chapter 3 presents a performance evaluation of the different DPR techniques inside Xilinx FPGAs using SDR as an application. Chapter 4 offers a detailed description of CONNECT NoC and the contribution made for adding the reconfiguration capability into it. Chapter 5 shows a complete study for the impact of introducing the configurability to NoCs. Chapter 6 presents a discussion and conclusion for the work in addition to the possible future work for this thesis.

Finally, Appendix A contains a user manual for the Reconfiguration tool and Appendix B contains the Reconfiguration tool source code.

Chapter 2 : Literature Review

2.1. Introduction

This Chapter covers the FPGA technology important concepts like the advantages and disadvantages of FPGA over ASIC, FPGA internal architecture, the Partial Dynamic Reconfiguration, and Network-on-Chip concepts. After those concepts being discussed, the reconfigurable NoCs are reviewed and the related work in this area is presented.

2.2. FPGA architecture and Capabilities

2.2.1. FPGA over ASIC

ASIC and FPGA market constraints and needs are different. The ASIC industry consumes a lot of cost for fabrication which includes the Non-Recurring Cost (NRC) with a much more optimized hardware than the FPGAs. Therefore, it is better for the large number of units or large fabrication volumes. On the other hand, the FPGA's cost with low volumes is much more efficient on the expense of design optimization. Therefore, it is more suitable with the initial prototypes and designs with relaxed constraints (speed, area, power ...).

In addition to that, with the technology advancement, the cross-over point between ASIC cost and FPGA cost is moving forward as shown in Figure 2.1. This gives an advance for the FPGA in the future when it comes to the unit cost, time to market, design cycle, and design reusability.

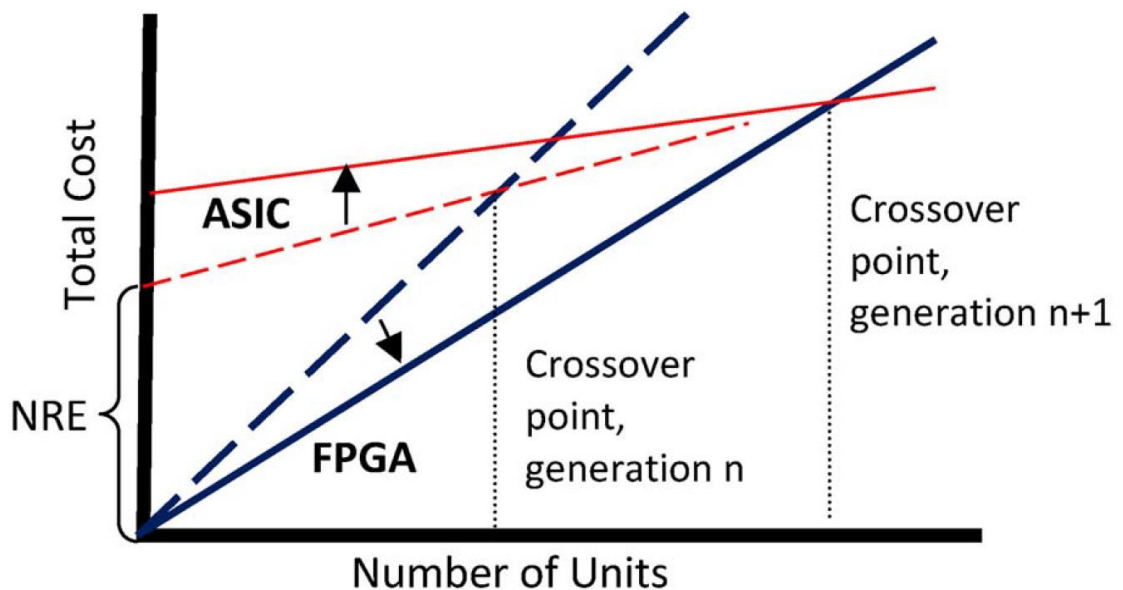


Figure 2.1: FPGA versus ASIC with respect to Cost and Number of units [1]

The FPGA architecture shown in Figure 2.2 is the most common architecture for the FPGAs and is known by the island style architecture. The arrangement of the Configurable Logic Blocks (CLBs) is in a 2D-grid and they are connected to each other by means of programmable routing networks. Moreover, the Input/Output (I/O) blocks are connected to the programmable routing network.

Due to the advancement in the design sizes and complexity, an enormous amount of logic is implemented inside the FPGA chip. This amount leads to the appearance of many routing and communication issues. These communication issues were challenging the new FPGA architecture advancements. Ideas like NoCs and DPR are offered by the FPGA providers as an option for specific design requirements. This thesis reviews the NoC and DPR concepts and discusses the opportunity of the reconfigurable NoCs in the next generation FPGAs.

2.3. Dynamic Partial Reconfiguration (DPR)

The configuration of the FPGA is the action of downloading a design into the programmable blocks of the FPGA. This is achieved by means of electrical pins and configuration memory. The configuration is categorized into the following different categories:

- **Full reconfiguration vs Partial reconfiguration**
The full reconfiguration is reconfiguring the whole design inside the FPGA, while the partial one is reconfiguring a part of the design maintaining the rest of the design as it is.
- **Static reconfiguration vs Dynamic reconfiguration**
The static reconfiguration is the reconfiguration while the design operation is stalled, while the Dynamic configuration is the configuration while the rest of the implementation is running (at runtime).

This thesis is mainly concerned with the Partial Dynamic Reconfiguration.

2.3.1. SRAM cells configuration topologies

Programming of SRAM cells in FPGA is achieved through different topologies that differ in simplicity and optimization. The SRAM configuration is categorized into the following categories:

- **Coarse grained vs Fine grained**
The coarse grained configuration requires programming of a large block as the design is divided into large ones. While in fine grained configuration, the design is divided into fine elements as the building block of the design configuration. This makes the coarse grained is better in routing simplicity and speed while the fine grained is better in area utilization.

Several SRAM cell configuration techniques are used and differ in the method simplicity, control and programming blocks access. Below is a description of different SRAM configuration topology with highlighting the advantage and disadvantage of each.

- Register chain configuration
The SRAM cells are arranged in a chain and the configuration is performed through a single configuration pin outside. This requires a certain configuration data pass by a long chain of cells before reaching its target SRAM cell. This topology has simple configuration wiring and requires a simple configuration controller. However, it is very poor in register access, especially when the number of registers is large. An illustration of the register chain configuration is shown in Figure 2.3 (a).
- Column based configuration
The column based topology provides an access to each column letting a quicker configuration than the register chain configuration topology. However, the column based configuration requires a column decoder for analyzing the target column and cell to be programmed. This way of decoding increases the wiring complexity. Therefore, the column based configuration is better in register access yet more complex in wiring and routing. An illustration of the column based configuration is shown in Figure 2.3 (b).
- Mixed configuration
The mixed configuration is based on both topologies, the register chain and column based ones are merged into this topology. The SRAM cells are divided into regions, each region is accessed like the column based configuration. Inside each region, SRAM cells are organized in a register chain. An illustration of the mixed configuration is shown in Figure 2.3 (c).

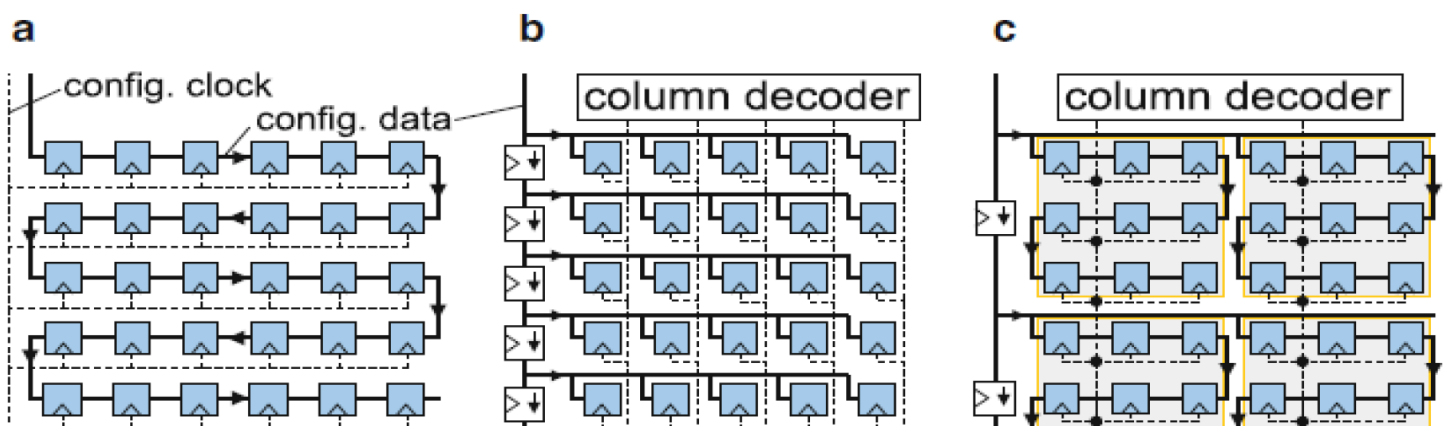


Figure 2.3: SRAM configuration topologies [3]

2.3.2. Multi context configuration

Some modern FPGA architectures go beyond the conventional single context configuration and investigate new areas in context switching. This context switching is achieved with a very high speed between different layers of configuration. This criteria is named as space-time configuration as it allows different plane context switching to be done in a time multiplexed way.

Figure 2.4 shows an example for Tabula FPGA that uses 12 planes and time multiplex between them by a high speed clock of 2 GHz while the user clock is 12 times slower.

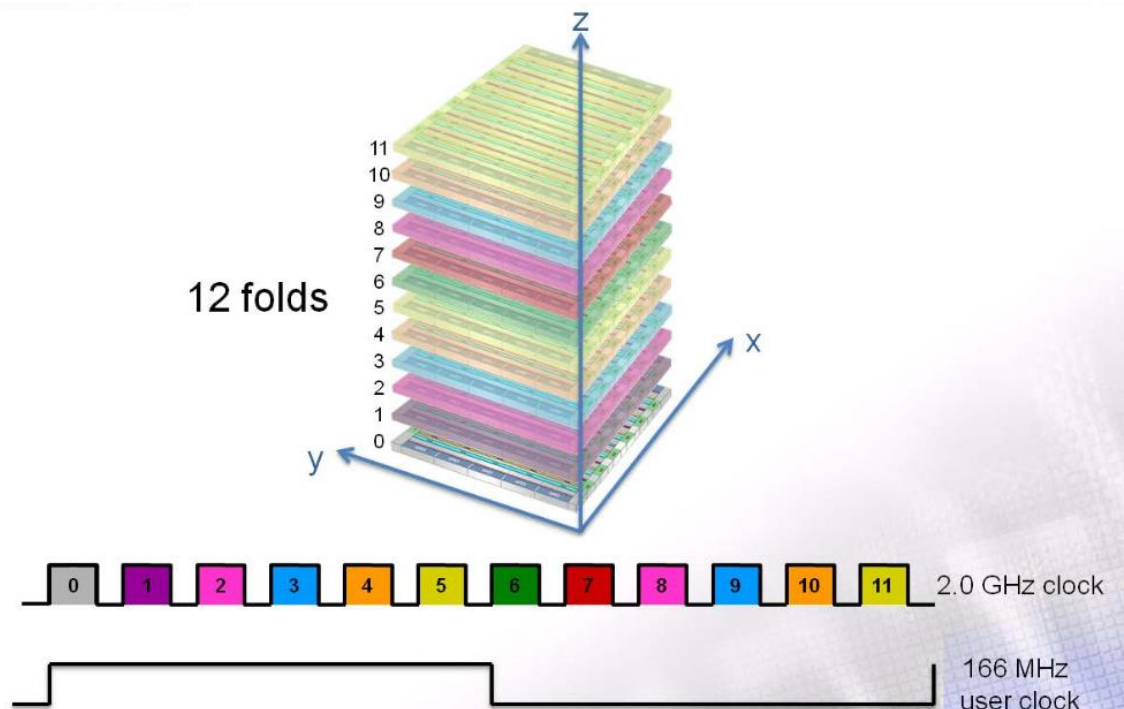


Figure 2.4: Tabula FPGA internal structure [4]

2.3.3. Dynamic Partial Reconfiguration Advantages

DPR is the act of reconfiguring a portion of an FPGA during runtime after its initial configuration.

The most valuable advantage of DPR is that it presents more flexibility to the hardware designs. In addition, DPR allows the implementation of complex circuits within a reasonable area and reduces static power consumption. Thus, it introduces the concept of virtual hardware [5].

DPR is used with applications that require high level of flexibility like SDR and some embedded FPGA applications; video processing, cryptography, and genomic sequence alignment. Moreover, DPR has an important role in implementing adaptive hardware algorithms and improve FPGA fault tolerance [6].

Most of Xilinx FPGAs (i.e. Virtex series) support DPR. Additionally, Altera FPGAs support the DPR. Figure 2.5 shows a high level illustration of the DPR within an SDR system.

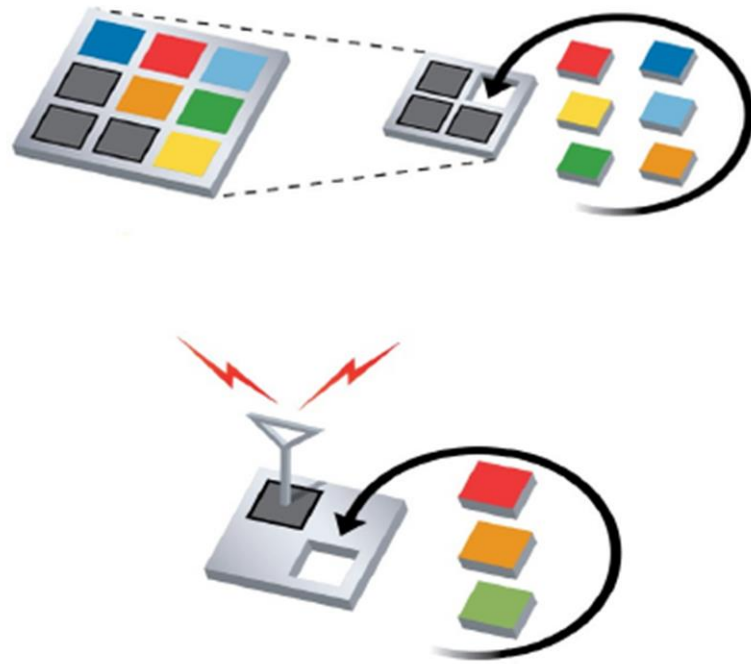


Figure 2.5: Dynamic Partial Reconfiguration illustration [7]

2.4. Network on Chips

Network on Chip (NoC) is a communication platform for connecting different apart elements in the design with each other. NoC is considered as an alternative for the conventional bus communication systems. Besides, NoC evolves due to the massive size of designs and its complexity.

The true value of NoC in FPGA appears in its scalability, modularity, and its power optimization compared to the conventional bus communication. The NoC mainly is composed of routers, links, and network interface. In general, the NoC is defined with certain parameters which are described below in the next subsection.

2.4.1. NoC parameters

Any NoC has some parameters that need to be covered as it impacts the network characteristics differently. Those parameters are as follows:

- **NoC topology**

The NoC topology is the arrangement and connection of the NoC router elements. There are a lot of NoC topologies such as Mesh, Ring, Star, Line, Fat Tree, and others.

The topology selection depends mainly on the application targeted from using the network. Figure 2.6 shows some examples for the different network topologies.

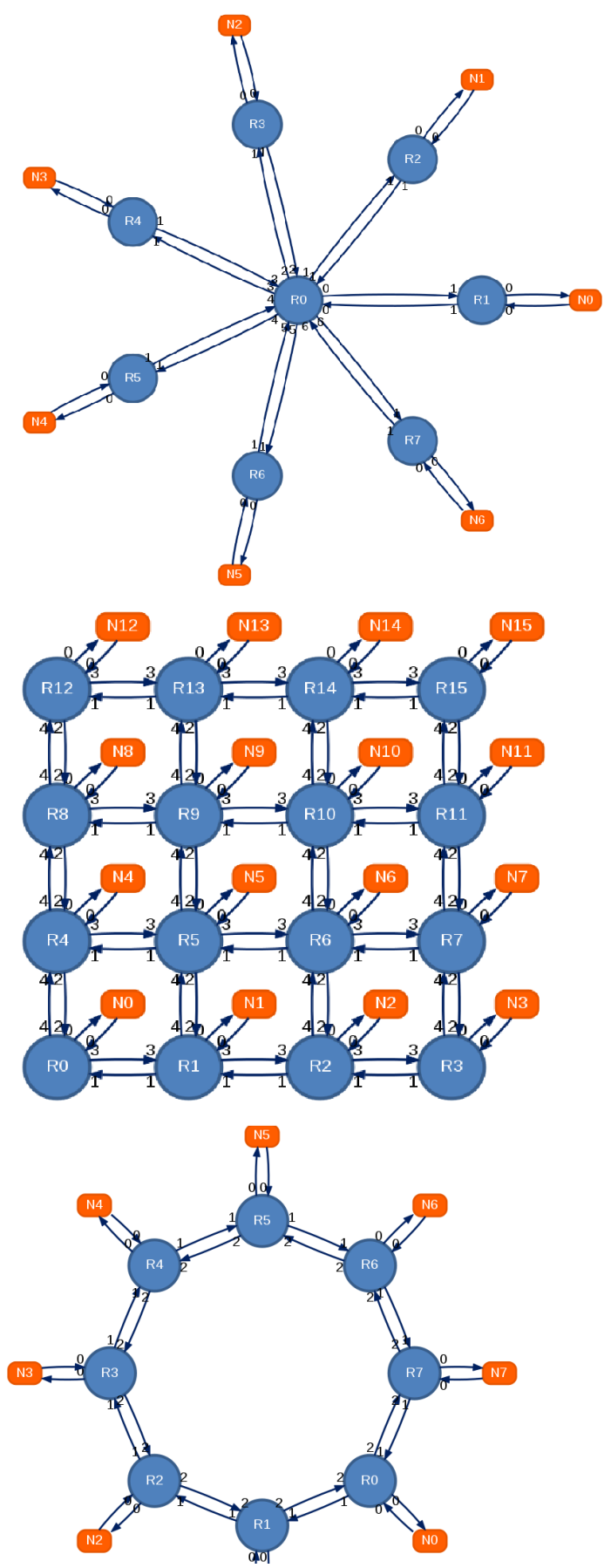


Figure 2.6: Star, Mesh, and Ring network topologies [8]

- **Flow control**

The flow control mechanism defines the way NoC is handling its own resources. The flow control is responsible for broadcasting the availability of the network nodes, buffer space, and virtual channel allocation.

Two main important types of flow control are: Peek flow control and Credit based flow control.

The peek flow control is a simple communication handshake informing the neighbors the availability of resource in each node without more detailed information about the remaining space available. On the other hand, the credit based flow control is responsible for sending credits with each resource update available or busy.

Figure 2.7 shows a simple structure for the credit based flow control.

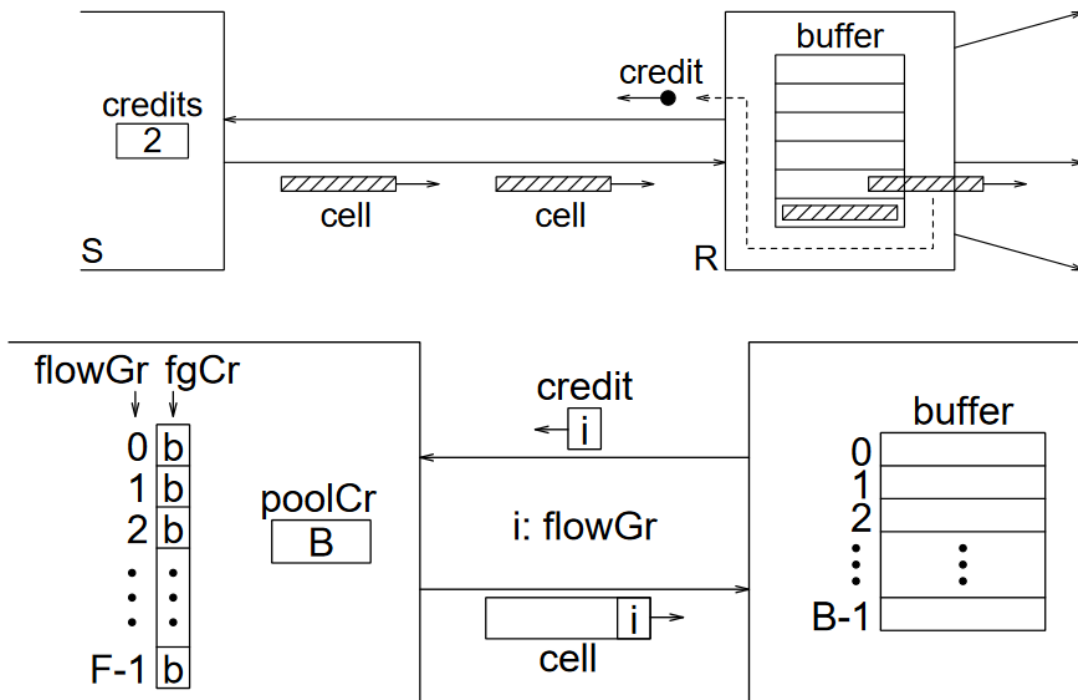


Figure 2.7: Credit based flow control structure [9]

- **Buffer Depth**

The buffer depth is the size of the memory buffer inside each router element. The buffer depth is reflecting directly how much packets can the router store from the source. These stored packets then go through the allocation and the router node pass them to their next station according to the routing path.

- **Virtual Channel**

The virtual channel provides the interface with separate paths for a virtual parallelization at the transaction level. Each interface port has its own buffer depth leading to the sense of separate queues which reduce the latency and enhance throughput. Some virtual channels may have priority over others according to the requirement and the arbitration sequence.

2.5. Challenges of Reconfigurable NoCs

The importance of reconfigurable NoCs appears with the designs intended to be dynamically reconfigurable. When the NoC is part of the design, its re-configurability gives the opportunity to operate with the best fit network to every user benchmark.

Many NOC architectures are proposed offering a high level of configurability. However, most of the work done in configurable NoCs is addressing the design time configurability not the runtime [31]. However, some configurable NoCs handle dynamic communication issues like surrounding or bypassing obstacles during runtime and how routing adaptation could be managed dynamically.

The next subsections discuss the related work in this area with a detailed coverage to the DyNoC as a good example for a dynamic adaptable NoC.

2.5.1. Previous efforts

In [10], the authors proposed a NoC handling circuit routing for dynamic reconfigurable devices and how this is better than the bus based communication architectures. In [11] and [12], DyNoC is proposed offering dynamic capabilities to the routing mechanism in order to guarantee the reachability to all the blocks and pins. Consequently, this is achieved by extending the well-known XY routing algorithm to an S-XY-routing (Surrounding XY routing) which is based on surrounding obstacles during runtime for a deadlock-free routing mechanism. The DyNoC is discussed in the next subsection.

Moreover, the authors in [13] proposed a CoNoChi NoC with minimal number of switches and area overhead with a similar deadlock-free mechanism. Moreover, they offered two ways of reconfigurations with the NoC stalled and without the NoC stalled.

In [14], the reconfigurability is achieved by placing or removing switches and using dynamic routing tables to guarantee full connectivity and every switch is accessed by the neighbors. In addition, network updates are propagated through special packets from a global control unit.

In [15], the NoC reconfiguration is based on local traffic monitoring and path weight calculations are passed to a global arbiter for selecting the minimum cost path.

2.5.2. DyNoC

The solution proposed by DyNoC is based on the communication between modules dynamically placed at runtime on a 2D NoC-based reconfigurable device. This is because of DyNoC advantages in performance, modularity, and structure. An advantage of the packet-based approach is that changing the network does not block communication, because packets are always routed in a strongly connected network. A set of components is strongly connected if and only if, for every pair of components, a path of routers exists connecting the two components. The DyNoC architecture achieves the packet-based approach.

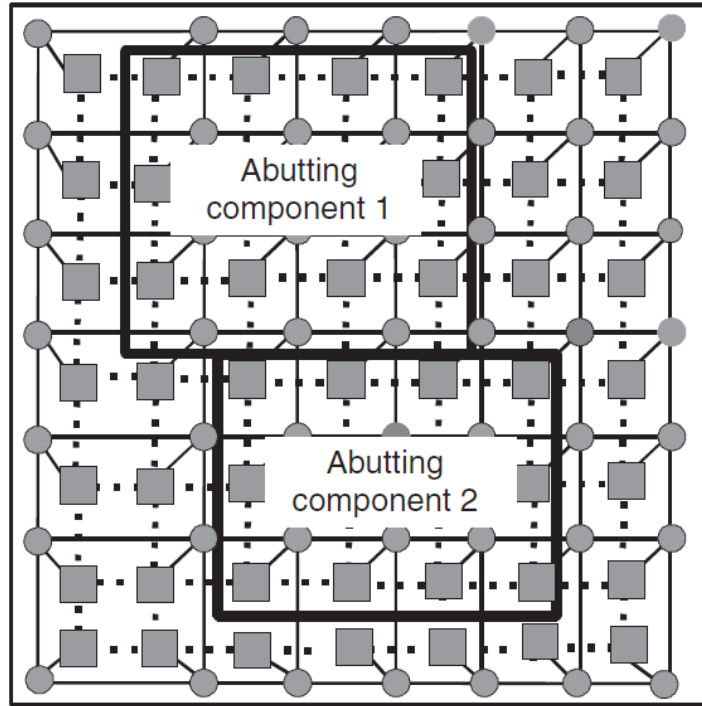
DyNoC implementation goal was not only to ensure module reachability. In addition, the pin reachability was targeted. Therefore, a requirement for the chip architecture is set: A ring of routers should internally surround the device. Figure 2.8 shows this architecture. Each task is implemented as a component, represented by a rectangular box and stored in a database. Because the synthesis is time-consuming, component synthesis must be done during the compile time. A box encapsulates a circuit implemented with the resources in a given area (router logic and PEs).

A component (or pin) at a given time on a reconfigurable device is reachable only if every packet sent to this component can reach the component. Because the chip's configuration is unknown in advance and communications among components are established during the runtime, all pins and components on the device must be assured to be reachable at any time during the temporal placement. This condition is met at any time if the pins and components on the device are strongly connected.

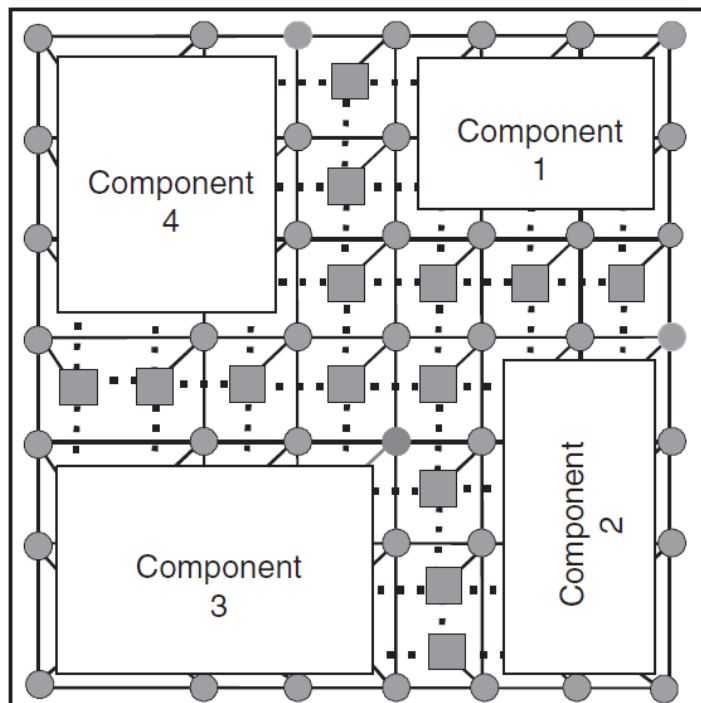
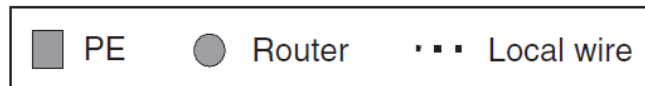
The one way to enforce strong connectivity is to make a ring of routers which always surrounds each component on the chip as a requirement. This can be achieved by one of two solutions. The first solution is synthesizing components so that they are always surrounded by a ring of routers when they are on the device. However, the second solution is achieved by leaving the job to a temporal placer. Nevertheless, this task increases the placer's complexity. Besides the computing free space in which to place a new component, the module placement must be strongly connected.

Therefore, the first solution is chosen. The following statement holds: If each component is synthesized in a way that it is only surrounded internally by PEs, then each placement on the reconfigurable device is strongly connected.

Figure 2.8 (a) and Figure 2.8 (b) show the different component placement types and how this affects the connectivity and the reachability of each PE.



(a)



(b)

Figure 2.8: Feasible (a) and infeasible (b) component placement [11]

In fact, if a set of components which is developed as required in the preceding statement and placed on the device is not strongly connected, at least one pair of components, or a single component, borders the device boundary. Without loss of generality, if the first case is considered, then either the two components will overlap or at least one component uses some routers on its internal boundary. The first case is impossible because only overlapping-free placements are valid.

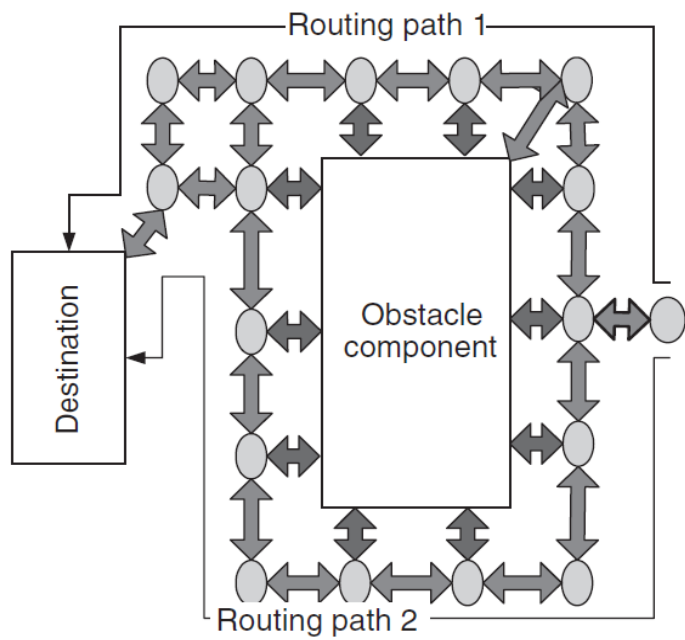
The second case contradicts the preceding statement's requirement. Figure 2.8 (a) shows an impossible placement, in which two components border. Figure 2.8 (b) illustrates a placement in which all components and pins are reachable.

In the static NoC, each router always has its own four active neighbor routers. However, this is not always the case in the Dynamic NoC like DyNoC which is presented here. Whenever a component is placed on the device, it covers the routers in its area. Those routers are deactivated because they cannot be used. Therefore, the component sets a deactivation signal to the neighbor routers in order to notify them not to send packets in its direction. Upon the completion of its execution, the component sets the deactivated routers back to their default state. Due to the obstacles created by the components dynamically placed on the chip, a routing algorithm that was used for common NoCs cannot work on the DyNoC in this case.

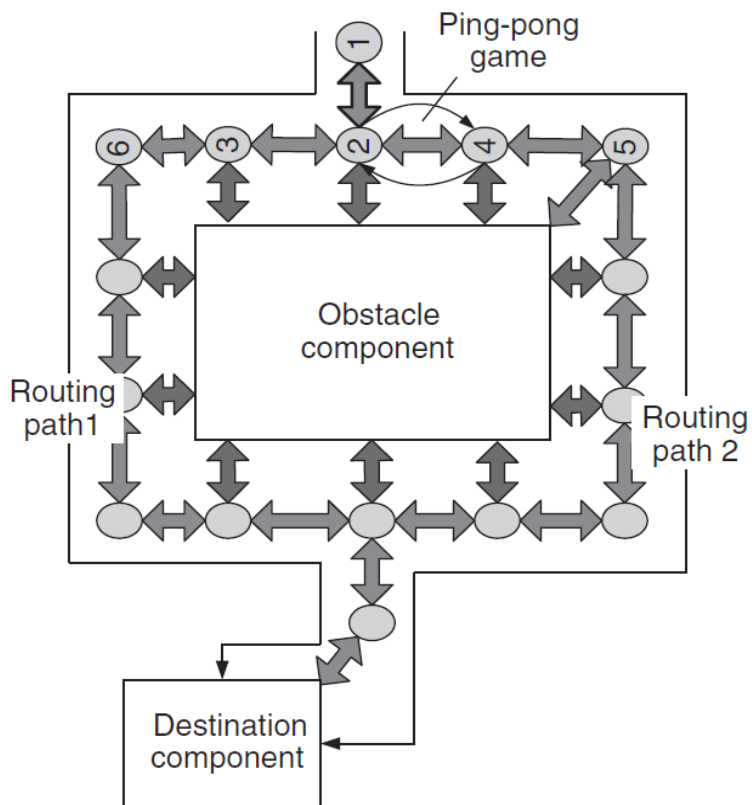
Therefore, a routing algorithm is provided to DyNoC (based on the well-known greedy XY algorithm) that considers network obstacles. As Figure 2.9 shows, the algorithm treats cases in which packets are blocked in the horizontal direction differently from cases in which packets are blocked in the vertical direction.

The proposed routing algorithm is called S-XY (Surrounding XY). As it is an extension of the XY routing algorithm, it still holds the properties of the XY algorithm: locally decisive and deadlock free. This means that each packet reaches its destination after a finite number of steps. Each router operates in three different modes:

- Normal XY (N-XY): A normal XY router behavior, the router sends a packet first horizontally to the right column and then vertically to the right row.
- Surround horizontal XY (SH-XY). The router enters this mode when its left or right horizontal neighbor is deactivated.
- Surround vertical XY (SV-XY). The router enters this mode when its upper or lower vertical neighbor is deactivated.



(a)



(b)

Figure 2.9: Obstacle surrounding in the horizontal (a) and vertical (b) directions [12]

Assume, without loss of generality, that an obstacle blocks a packet moving from the right to the left. As Figure 2.9 (a) shows, there are two alternative paths by which the packet is routed to reach its destination. If the Y-coordinate of the packet's destination is greater than or equal to the local router's Y-coordinate, the local router chooses the first path and sends the packet upwards. Otherwise, the local router chooses the second path and sends the packet downwards. A problem occurs when, for example, a packet with destination Y_{dest} traverses upwards and reaches router R whose coordinate Y_r which is greater than Y_{dest} .

According to the previously defined scheme, the packet traverses downwards to the router with coordinate $Y_r - 1$, which sends it upwards, thus producing a Ping-Pong effect. To avoid this Ping-Pong behavior, the second router stamps the packet by setting a stamp bit to 1 to notify router R not to send the packet back. The stamp bit is removed when the packet reaches the router at the device's upper right, and the packet traverses left until reaching its destination column or encountering another obstacle. The algorithm treats cases in which a packet moving vertically is blocked in the same way, except that all packets must be stamped. Otherwise, the Ping-Pong effect will always occur between router 2 and router 4, as shown in Figure 2.9 (b).

Whenever a packet is blocked in a given direction, it takes the perpendicular direction. This allows the packet to continue until it reaches the last router on the blocking component boundary at one corner of the module that the packet must surround. From this point, the N-XY routing algorithm can resume. Therefore, a packet's looping around a component is not possible.

Obviously, the algorithm creates a placement sequence in which a packet keeps moving around in the device and never reaches its destination. This problem is common at online algorithms. However, only one packet is lost, and all the remaining packets reach their destinations. A packet's probability of being blocked is then less significant. In the S-XY routing, the direction is fixed in advance for all routers in which to send a packet whenever it encounters an obstacle. This can lead to extremely long routing paths like that is shown in Figure 2.10 (a), which is caused by placements for which the routers always choose the right path.

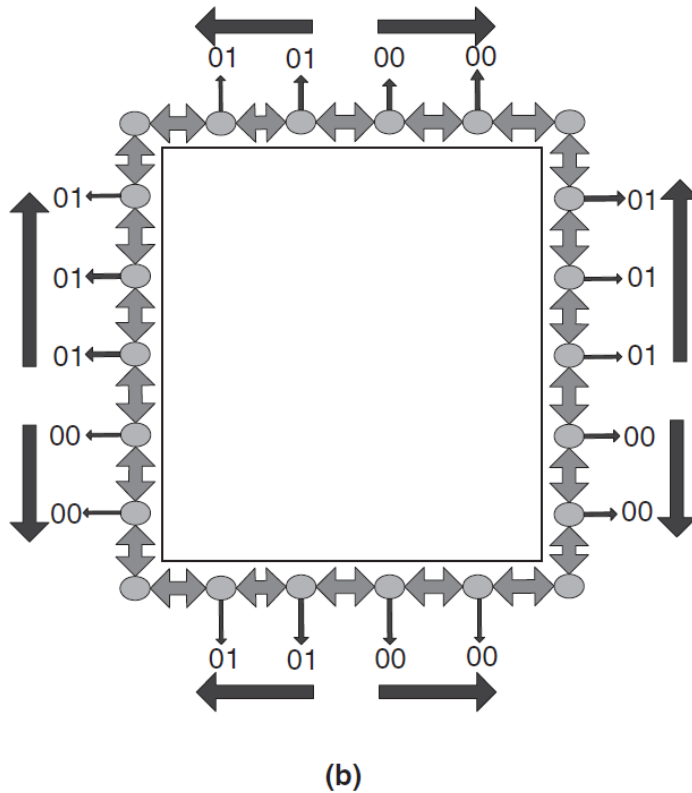
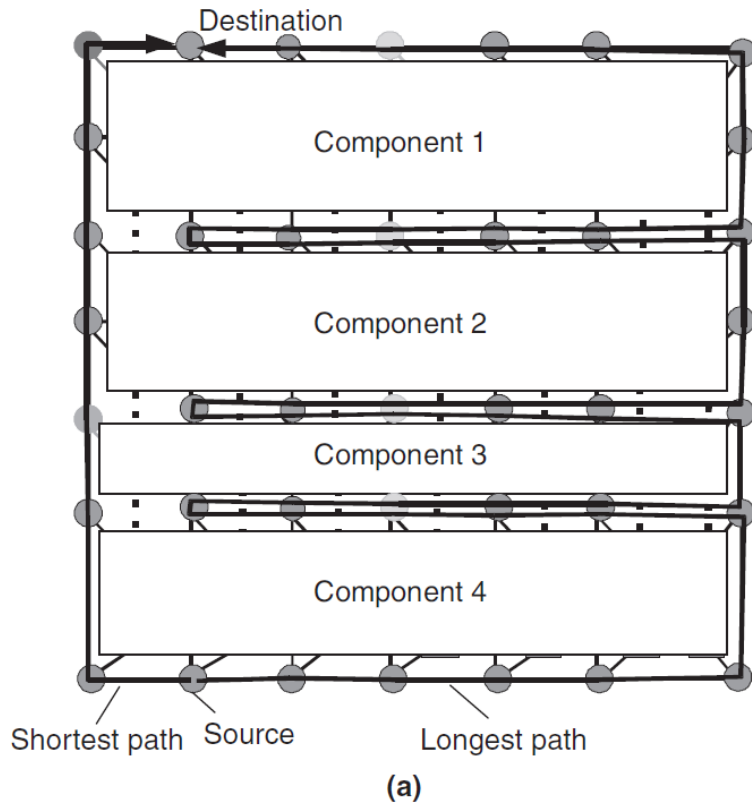


Figure 2.10: Placement leading to extremely long routing path (a); Router guiding in the DyNoc (b) [12]

To avoid this problem, the placed component informs each router which direction to take whenever the component blocks an incoming packet in a given direction. Figure 2.10 (b) illustrates this approach which is called router guiding. Instead of one activation line code, two lines are used: the first is for activation (1 = activate, 0 = deactivate), and the second is for direction (0 = east or south, 1 = west or north). This limits the router complexity considerably and eliminates the need for stamping.

Some component placements lead into making multiple routers has the same direction of routing and decisions, the algorithm is merging of the bit-stream information such as the one shown in Figure 2.11.

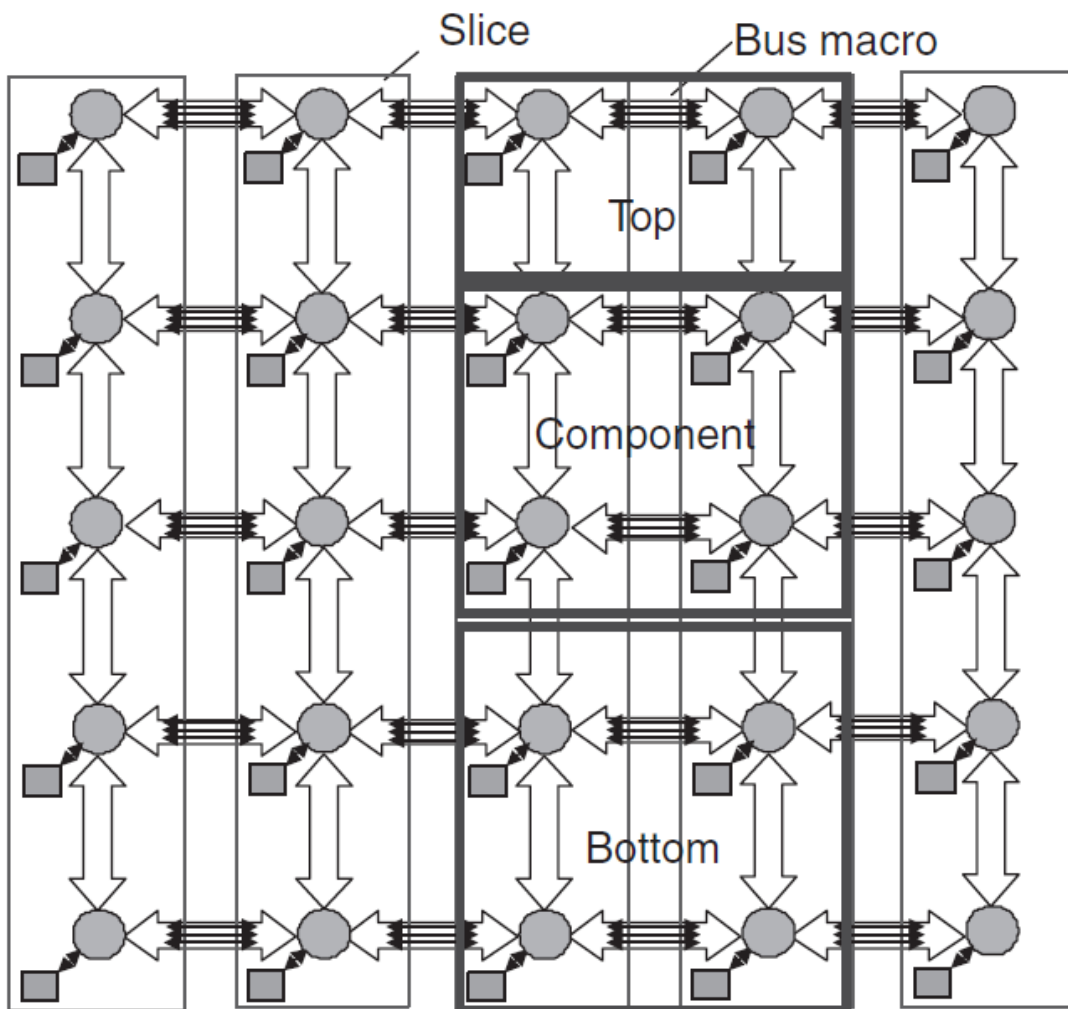


Figure 2.11: Merging bit-stream for a 2D partial reconfiguration on Xilinx FPGAs [12]

Chapter 3 : Performance Evaluation of Dynamic Partial Reconfiguration Techniques for SDR on FPGA

DPR presents more flexibility to the hardware designs which are considered as its most valuable advantage. Additionally, DPR allows the complex circuits implementation within the constant area and works on reducing the static power consumption, thus it introduces the concept of virtual hardware [5]. DPR is recommended with applications that need high level of flexibility like SDR and some embedded FPGA applications; cryptography, video processing, and genomic sequence alignment. Besides, DPR plays an important role in implementing adaptive hardware algorithms and improves FPGA fault tolerance [6].

Many Xilinx FPGAs (i.e. Virtex series) support DPR in a way that allows the user to specify the reconfigured area while maintaining the static logic unaffected. The reconfiguration takes place with different reconfiguration methods like JTAG and ICAP.

Since the partial reconfiguration is implemented at FPGAs and has taken place, more opportunities appeared and take the advantages of dynamic reconfiguration especially in Software Defined Radio (SDR) implementation. McDonlad [16] presents an overview of reconfigures Forward Error Correction (FEC) for partial reconfiguration designs on virtex-4 and comments on the additional overhead necessary for creating this design.

Moreover, Delahaye and Palicot in [17] are targeting the implementation of Convolutional Coder, FIR filter, and a constellation mapper. They implement management architecture based on MicroBlaze interconnected with a NoC which is extended from a 3G wireless communications system.

Tan, DeMara and Ejnoui [18] evaluated (JTAG and SelectMAP) as the only two interfaces of Virtex II in terms of design complexity and performance. The conclusion was that the JTAG design consumes third of the device I/O pins, and from 3 to 7 times fewer logic area. However, the poor throughput of JTAG interface degrades the reconfiguration performance with a factor of 40 than the SelectMAP.

In [19], [20], [21] and [22] the DPR using ICAP interface only is discussed. In these papers, different versions of ICAP interface are exploited starting from OPB_HWICAP which is connected to the On-Chip Peripheral Bus (OPB), then XPS_HWICAP which uses PLB bus, and finally AXI_HWICAP which is connected to AXI bus. All the previous IPs were experimented with various memory configuration setups to measure the reconfiguration time from the system side and evaluate the overhead added by reconfiguring the interconnecting components that are inserted during the reconfiguration process.

This section covers the different configuration techniques in Xilinx FPGAs and presents an evaluation of those techniques based on experimental results using an SDR application.

3.1. Xilinx FPGA Configuration Techniques

DPR in FPGA is achieved by loading a partial bitstream along with the static bitstream. And, this is performed through different interfaces. Xilinx FPGAs offer four interfaces to program a partial bitstream from nonvolatile memory into reconfiguration memory [7]. Some of these interfaces need Partial Reconfiguration controller which is either located externally (in an external device - a PC for example) or internally (inside the FPGA's fabric - like MicroBlaze). Figure 3.1 shows the four techniques applied to a convolution encoder inside a communication system.

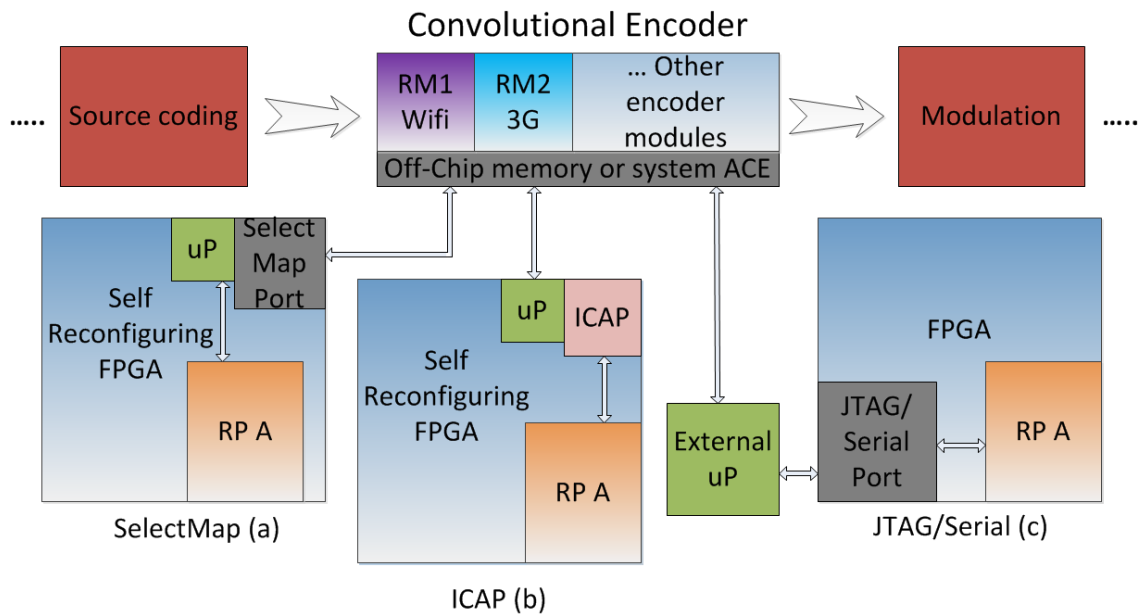


Figure 3.1: Reconfiguration techniques of convolutional encoder inside communication chain

The following sub-sections presents a description of the different reconfiguration techniques in Xilinx FPGAs (JTAG, Serial Mode, SelectMAP, and ICAP) and the differences across them. Additionally, an evaluation of these techniques and their preferred usage model.

3.1.1. JTAG

The JTAG is an acronym for Joint Test Action Group, this group is the one that developed the JTAG standard. JTAG is widely used in testing and as an important debugging tool as it can communicate data out through I/O ports for testing board level connections. In addition, it can internally send signals for testing device behavior, these tests aim for shorts and opens detection at board and device levels. The JTAG configuration is achieved by downloading the bit stream file that is stored on the PC using the iMPACT utility and the Xilinx programming cable as in Figure 3.1 (c). The partial reconfiguration takes place through downloading the partial bit stream the same way.

When JTAG is used for multiple devices configuration, the control signals should be connected in parallel like the TCK pin which is driven by the Xilinx programming cable [18].

3.1.2. Serial Mode

Using slave serial configuration mode, loading the configuration data is achieved one bit per “CCLK” cycle. The “CCLK” in the slave serial mode must be driven externally from an external control logic. The slave serial mode usually is used in configuring a single device from an external microprocessor as in Figure 3.1 (c) or configuring multiple devices in a daisy chain.

The dedicated pins to the slave serial mode that are required for configuration include PROGRAM_B, CCLK, DONE, INIT_B, Din, DOUT_BUSY, and the mode pins M[2:0]. The mode pins should be tied high with the slave mode [3011]. In addition, the single configuration is used to configure multiple devices arranged in a daisy chain. Each device in the daisy chain receives the configuration data through its D_IN pin and is required to pass it to the next device in the chain through its DOUT pin till the last device in the chain is configured and accordingly all the devices release their DONE pins.

3.1.3. SelectMAP

SelectMAP offers an 8-bit, 16-bit, or 32-bit configuration interface with bidirectional data bus interface to FPGA’s fabrication, this interface is used for both configuration and read-back. SelectMAP works in two modes; Master mode which drives the configuration clock, or Slave mode which is driven by an external configuration clock. Read-back is applicable only in case of Slave SelectMAP mode.

There are various setups for the SelectMAP like a Single-device Slave SelectMAP which includes a processor which provides data and clock. Alternatively, a CPLD is used as a configuration manager. Another setup is Multiple-device daisy-chain that is used to configure multiple number of FPGAs in series with different bit-streams buffered from a nonvolatile memory or processor [23].

Slave SelectMAP is the only mode that allows performing partial configuration in all Xilinx FPGA’s as master modes are clearing all FPGA’s configuration memory as in Figure 3.2.

For carrying out the reconfiguration process using Slave mode SelectMAP, 38 pins are required including the DATA pins (D0:D31), DONE, CCLK, BUSY, PROG_B, CS_B, RDWR_B, and INIT_B. Multiple FPGAs are connected to SelectMAP bus and share some pins with others FPGAs.

Keeping the general propose DATA pins as configuration pins, the persist option of BitGen need to be set otherwise, the DATA pins in this case become user pins after configuration. Besides, the SelectMAP is an 8-bit width interface by default unless other SelectMAP width is selected with the CONFIG_MODE constraint.

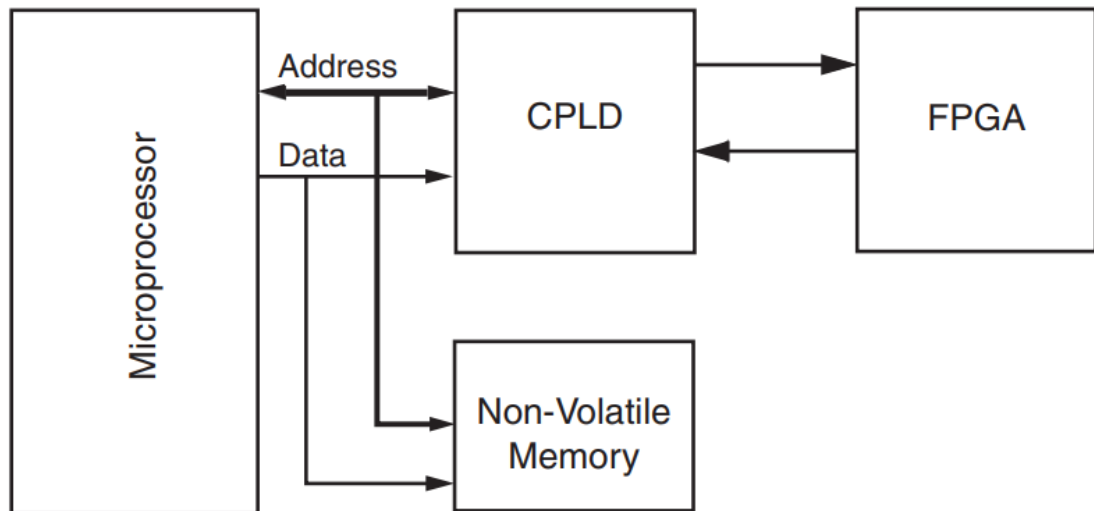


Figure 3.2: SelectMAP structure in Xilinx Virtex 5 FPGAs [24]

3.1.4. ICAP

Internal Configuration Access Port (ICAP) is a Xilinx interface that provides a direct access to the configuration logic at the FPGA fabric. During the run-time, ICAP interface allows the configuration data to be loaded/downloaded into/from the FPGA configuration memory. Moreover, it permits status registers reading of the configuration logic.

The ICAP interface is similar to SelectMAP slave mode interface except with a dual port 8-bit, 16-bit, or 32-bit data bus for reading and writing configuration data. The ICAP interface uses BUSY, CE, WRITE, and CLK signals [23]. Configuration data is written to the device at the rising clock edge and if the ICAP port is enabled. Consequently, the configuration data writing is controlled by the clock as well as by setting the enable signal while connecting the ICAP primitive to a fixed clock.

Even though, there are two available ICAP primitives starting from Virtex-5, the two ports cannot be operated simultaneously. The design must start with the top ICAP, then alternate between the two ports sequentially.

ICAP caches the configuration bits into BRAM before loading to the FPGA configuration memory Figure 3.1 (b). Xilinx provides an IP core called OPBHWICAP that is connected to the OPB bus as a slave peripheral, and enables the processor to access the configuration memory through the ICAP, by using a library and software routines using EDK toolkit. For the Virtex-4 and Virtex-5 FPGAs, the XPSHWICAP then AXI_HWICAP was released which works similarly with the OPBHWICAP. However, it is connected on the PLB and AXI bus respectively. This achieves a lower-latency reconfiguration.

3.2. Software Defined Radio

Software Defined Radio (SDR) is a concept where the different hardware parts can be replaced and controlled by means of software according to the current requirement. Recently, the SDR as an application became a trending one for DPR usage. As wireless technologies gain their growth and development, more advanced standards will be released, thus the demand to implement these entire standards in one device is obligated. On the other hand, hardware designs are required to provide compatibility with the current standards, if even possible, will most likely become obsolete after a short while. Though, SDR system maintains the flexibility to control the same hardware resources via software for these multi-communication devices.

The reconfigurability of FPGAs is considered as a good asset to the SDR systems for loading the desired standard according to the runtime need. Practically, the ability of reconfiguring a specific block provides an opportunity to create an extremely flexible and compact design, while all other blocks are working normally. This permits chip area saving and power reduction.

The advantages of SDR system are noticed clearly after applying DPR techniques on Convolutional Encoder block. This encoder is responsible of generating FEC coding schemes. These coding schemes are used for decreasing the channel noise.

Convolutional encoder outputs are affected by the code schemes used in the current standard, and as well affected by several parameters (n , k , l) which are being used for describing convolutional codes. Where “ n ” represents the input encode elements, “ k ” represents the output encode elements, and “ l ” represents the shift register numbers of convolution encoder.

In the proposed experiment, two encoder schemes; WIFI and 3G communication systems, are used as a benchmark for DPR as shown in Figure 3.1. Following this approach is called a Single-Loaded Encoder Module (SLEM) where DPR is used to implement one encoder at a time on the chip.

3.3. Results and Discussion

The experiment is aiming to apply DPR to the implemented SDR design using the different configuration methods and compare between them with respect to the area and reconfiguration time. This design has been implemented using XUPV5-LX110T kit which includes Virtex-5 xc5vlx110tff1136-1 FPGA, System ACE Compact Flash configuration controller to store bit-stream files of PR regions, and UART interface to interact with MicroBlaze by sending reconfiguring commands.

As mentioned previously, the reconfiguration time of DPR is not directly related to design resources. Nevertheless, it is proportional only with PR region which is translated to number of frames which are the minimum building blocks for PR region.

In the past, in Virtex and Virtex II families, frames that are consisted of the whole column of FPGA. Starting form Virtex 4, frames became a complete tile which includes certain number of CLBs of a whole column, and this number is increasing with each new Xilinx family. Therefore, the total design size (static and PR region) will be the major effect on the reconfiguration time.

Consequently, the SDR design size is varied along the experiment in order to check the variation in performance of each configuration technique. The variation steps are chosen taking into consideration a significant change in the partial bit stream file size which reflects directly in the estimated reconfiguration time.

Table 3.1: Reconfiguration speed for different Reconfiguration techniques [28]

Configuration Mode	Data Width	Max. clock rate	Max. Bandwidth
JTAG	1-bit	66 MHz	66 Mbps
Serial Mode	1-bit	100 MHz	100 Mbps
ICAP	8/16/32 bits	100 MHz	0.8/1.6/3.2 Gbps
SelectMAP	8/16/32 bits	100 MHz	0.8/1.6/3.2 Gbps

The figure of merit chosen for the comparison between these different techniques is the area multiplied by reconfiguration time. This metric is a good indicator to the performance variation from a certain design size to another. Considering the area overhead with the speed makes the comparison more fair between the serial and the parallel techniques.

In addition, the number of occupied LUTs is considered as a significant indicator to the design area as shown at vertical axis and horizontal axis in Figure 3.3.

The theoretically estimated reconfiguration time is calculated according to (1), where “Bssize” is the bit-stream file size of the PR region, “Clkmax” is the maximum clock rate supported by reconfiguration interface, and “Dw” is interface data width. These values are listed in Table 3.1 for each interface.

$$Reconfiguration\ Time = \frac{Bssize}{Clkmax * Dw} \quad (1)$$

The reconfiguration region sizes are chosen in a way to completely occupy a certain number of frames. This is done in order to make use of the whole area without any change in the partial bit stream size and without affecting the estimated reconfiguration time.

Figure 3.3 shows the performance of the JTAG, Slave Serial mode, Slave SelectMAP 8-bit, and ICAP 8-bit data width using the SDR design with different selections of PR regions. It is obvious that at small designs that need PR regions less than ~400 and ~750 LUTs for JTAG and Serial mode respectively, JTAG and Serial mode are better in performance than ICAP and SelectMAP which work with 8-bit width at 50 Mhz.

These values are decreased (~150 and ~300 LUTs) when ICAP and SelectMAP work at 100 MHz taking into consideration that ICAP, due to the others used resources in FPGA fabric, allow maximum frequency less than JTAG and Serial mode. This can be avoided when using SelectMAP as it has external “CCLK” port.

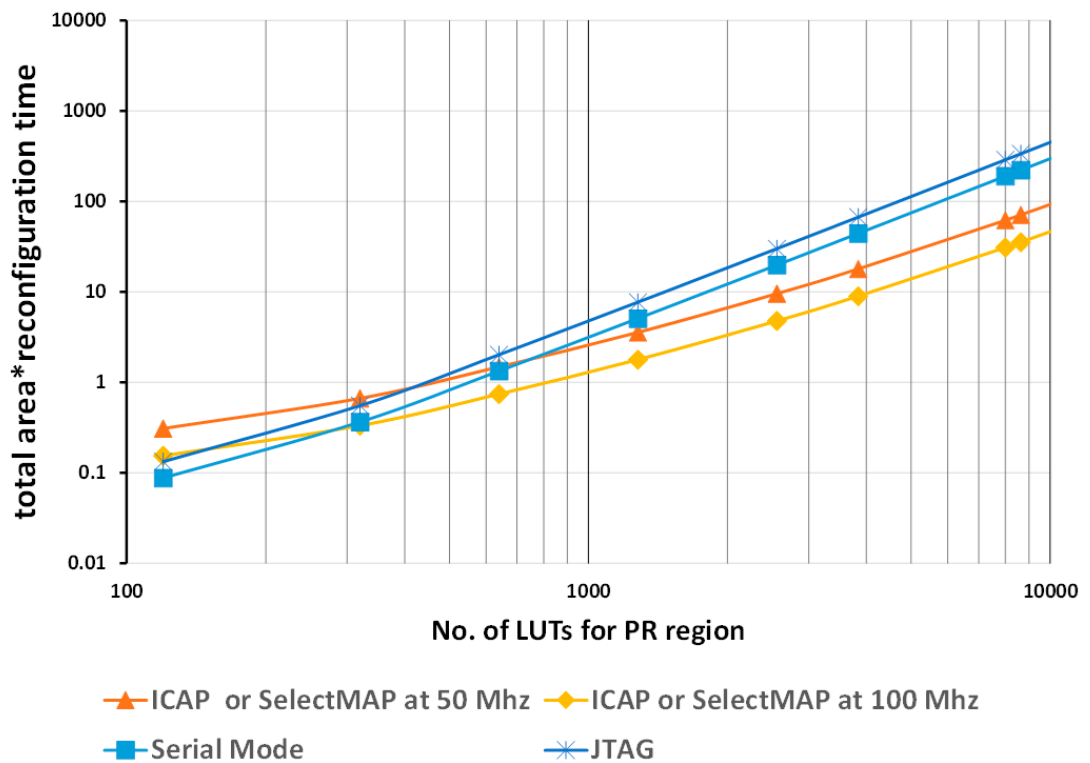


Figure 3.3: 8-bit ICAP and SelectMAP with Serial mode and JTAG [28]

In Figure 3.4, the experiment is repeated with 16-bit data width for ICAP and SelectMAP. It is noted that the intersection points decreased more (~150 and 300 LUTs for JTAG and Serial mode respectively) compared with ICAP working at 50 Mhz. These values decreased because the comparison becomes unfair for serial interfaces like JTAG and Serial mode compared to 16/32-bits ICAP and SelectMAP as the parallel configuration always gives more capability to reach high configuration speed.

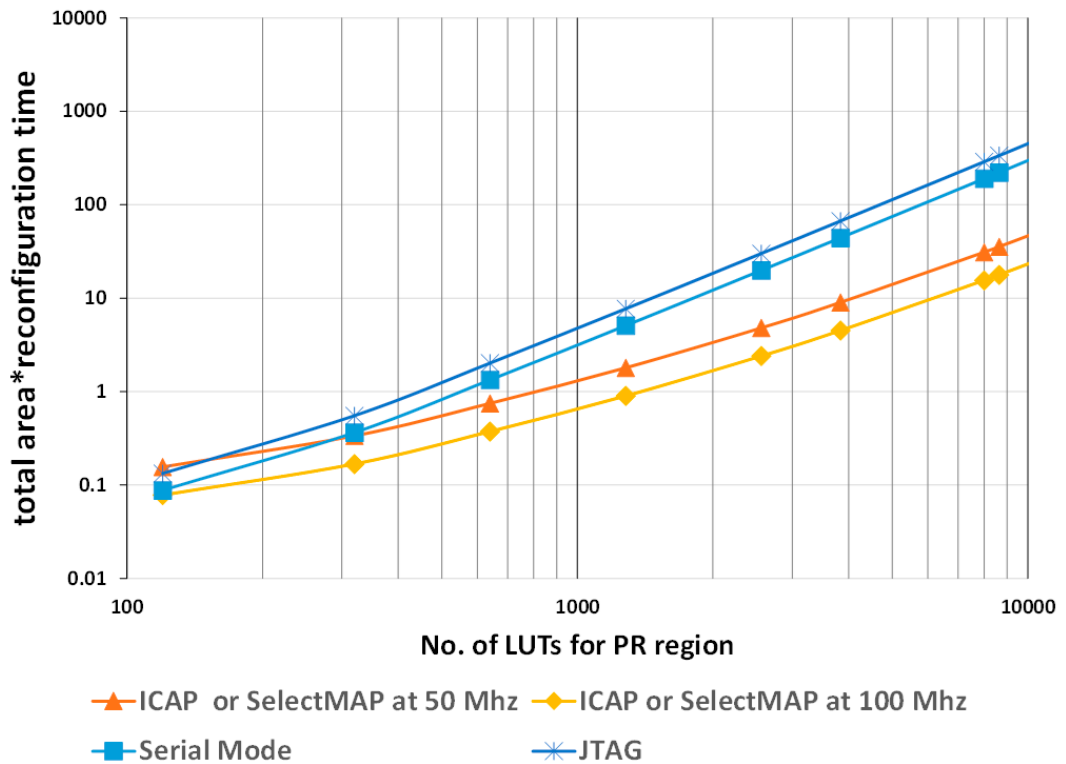


Figure 3.4: 16-bit ICAP and SelectMAP with Serial mode and JTAG [28]

Designs which are using JTAG and Serial mode can save ~2400 LUTs compared to ICAP and SelectMap, this overhead is significant with small area designs. However, the reconfiguration speed of 16-bit ICAP and SelectMap is better with factor of 24.2 and 16 than JTAG and Serial mode respectively.

ICAP and SelectMAP are always recommended if they worked with full data width 32-bit over other reconfiguration techniques as shown in Figure 3.4. However, the drawback of this scheme is the used I/O pins. In SelectMAP, these pins have to be reserved the whole time for reconfiguration purpose only. While in ICAP, these pins are used as I/O general after the reconfiguration has been done.

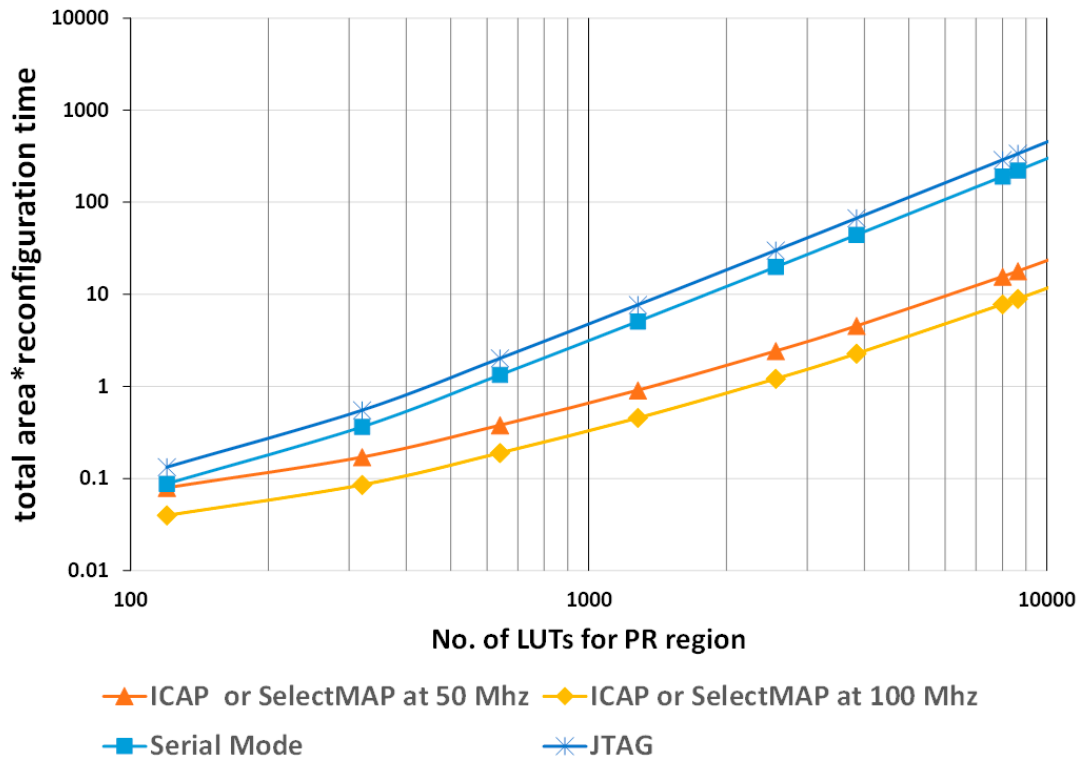


Figure 3.5: 32-bit ICAP and SelectMAP with Serial mode and JTAG [28]

Chapter 4 : Applying Dynamic reconfiguration to CONNECT NoC

4.1. CONNECT NoC Architecture

Different NoCs are reviewed and compared in [31] while NoC parameters are reviewed in [32]. However, this work is based on using CONNECT network for NoC-based FPGA [30]. CONNECT achieves an efficient network performance by means of its consistent lower latencies for FPGA-based designs. Moreover, CONNECT offers a fully parameterizable router design and flexible network routing, allocation and flow control mechanisms. The user generates the desired NoC through an RTL configurator as in Figure 4.1 where different network configurations are supported, such as network topology, network size, number of virtual channels, buffer depth, data width, allocation type, and flow control mechanism.

The internal structure of CONNECT is based on optimized RTL implementation to different modules in a scalable manner in order to satisfy its customizable requirements. Figure 4.2 shows the main internal structure for CONNECT router core. Each router core can communicate with four neighbour routers in addition to the outside user port which makes them five open communication channels. The core contains input handler for routing input packets and input queues for internal storage. Additionally, output port FIFOs are required for dealing with neighbor routers availability [30].

Parameter	Value
Network Topology	
Topology (i)	Mesh (v)
Routers per Row	4 (v)
Routers per Column	4 (v)
Expose Edge Ports (i)	<input type="checkbox"/>
Network and Router Options	
Router Type (i)	Virtual Channel (VC) (v)
Number of VCs (i)	2 (v)
Flow Control Type (w)	Credit-Based Flow Control (v)
Flit Data Width (i)	64 (v)
Advanced Options (click to expand)	
Flit Buffer Depth (i)	8 (v)
Allocator (i)	Separable Input-First Round-Robin (v)
Use Virtual Links (i)	<input type="checkbox"/>
Debug Symbols (w)	None (v)

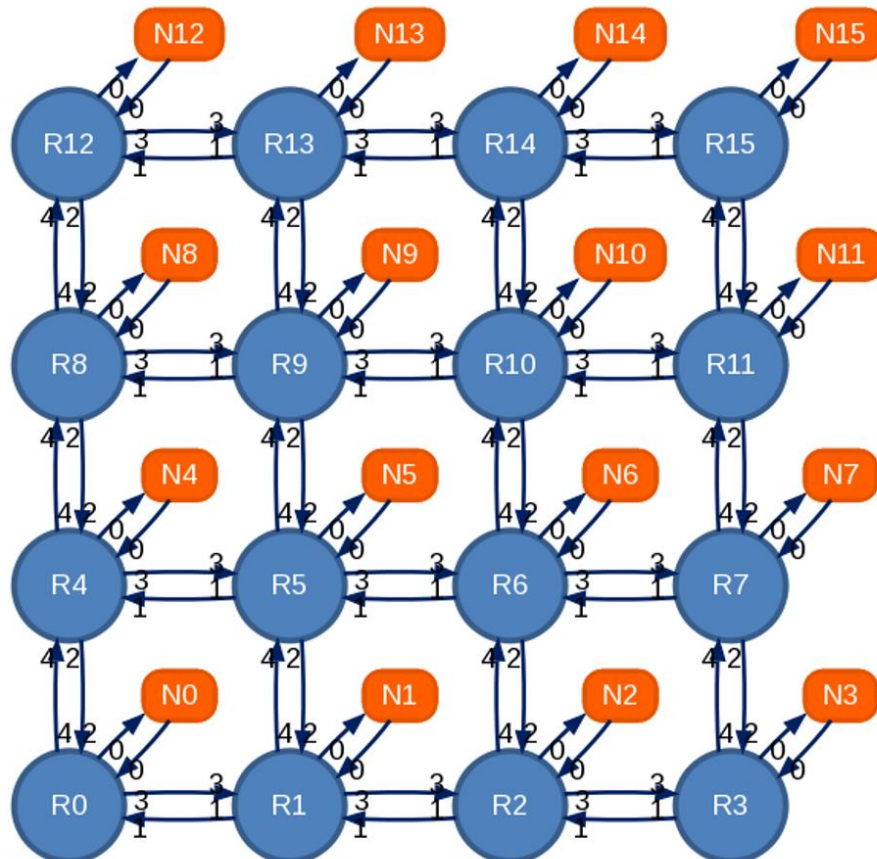


Figure 4.1: CONNECT NoC RTL configurator [8]

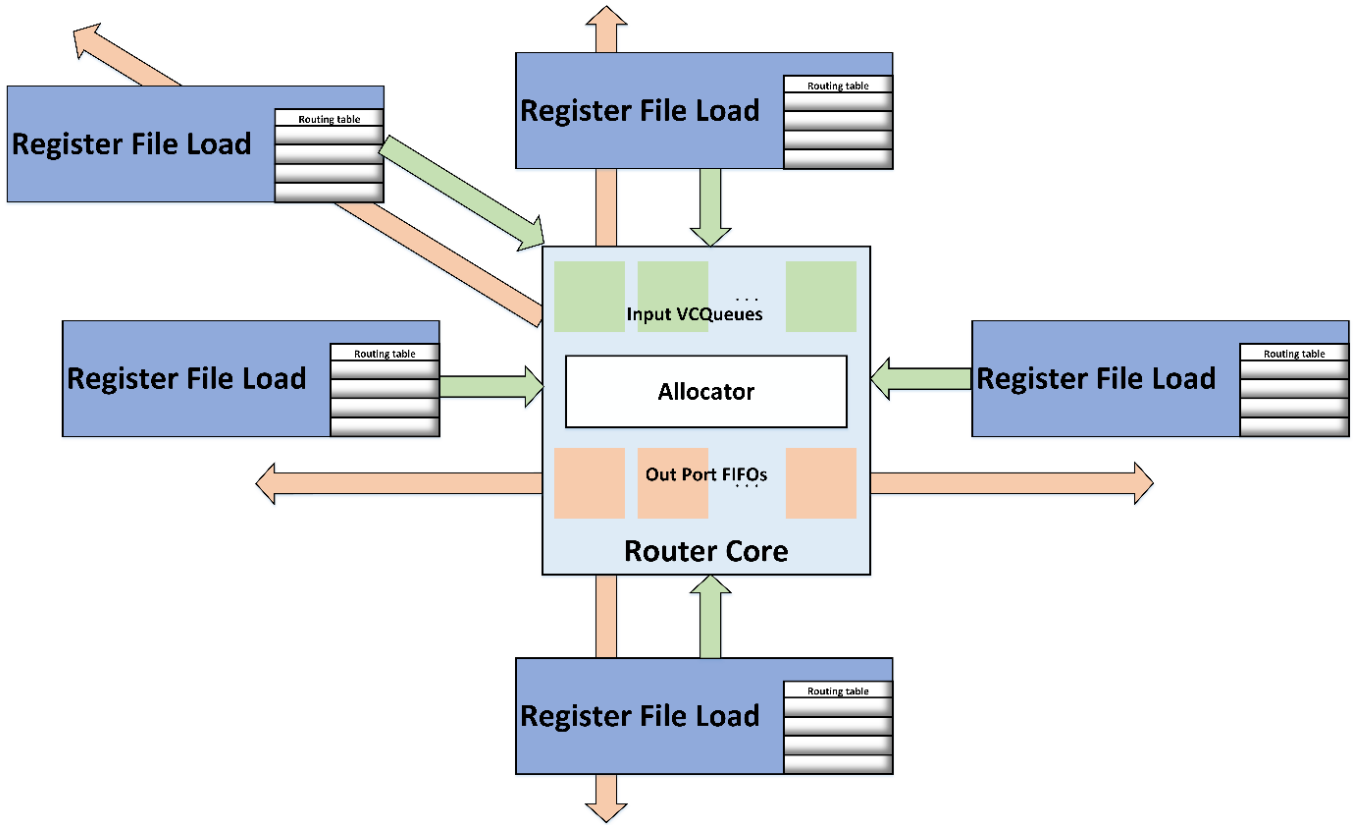


Figure 4.2: CONNECT NoC router core internal structure

The chosen network configuration is a 4x4 mesh network with two virtual channels, credit-based flow control and buffer depth varying from 4 to 64. The CONNECT 4x4 mesh with two virtual channels is recommended by [25] giving optimal performance with highest throughput and lowest latency for high injection rates.

The following subsection describes the changes made to CONNECT RTL in order to introduce the reconfigurability during the runtime. Additionally, it shows how the reconfiguration improves network flexibility to be adaptive according to the user benchmark requirements on the expense of adding a significant area overhead.

4.2. Reconfiguration changes to CONNECT

The reconfiguration of the network during the runtime requires dynamic adaptation to the routing as each router needs to know if the neighbor router is still active.

Therefore, some changes are applied to the generated fixed CONNECT RTL in order to introduce the runtime reconfigurability to each router. The RTL updates are as follows:

- A global input configuration is added for indicating the current network status of each router. Each bit corresponds to a single node whether it is active and normally functioning with the other nodes or inactive and being bypassed by the other nodes.

- The routing inside all the routers is adapted considering the possibility that the neighbor is inactive and some packets may need to change route. This change adds significant area overhead due to preserving the original routing tables. All the routing combinations are considered covering the four routers possible reconfiguration. The popular XY routing algorithm is modified in order to find a substitute routing path when the neighbor router is getting reconfigured. For example, when the packet needs to be routed in the horizontal path while it is not available, the router release the packet in the available vertical path in order to let the packet to go in a surrounding path around the obstacle.

- Connectivity switches are added around each router to bypass the inactive neighbor routers in case any of them being reconfigured.

The previous changes implies a limitation to the reconfiguration as it is not allowed to reconfigure two consecutive routers at the same time, The reason behind this is that each router has the information of the availability of the first level neighbors only. If the second level neighbor is also reconfigured at the same time, some packets might be mistakenly lost. However, the wire delays between routers need to be considered.

In general, the RTL changes added an area overhead to the CONNECT RTL implementation. The area overhead is for the added network map module, connectivity switches, and the routing adapted LUTs. The new structure for the reconfigurable CONNECT NoC is shown in Figure 4.3. All these RTL changes are described and discussed in [29].

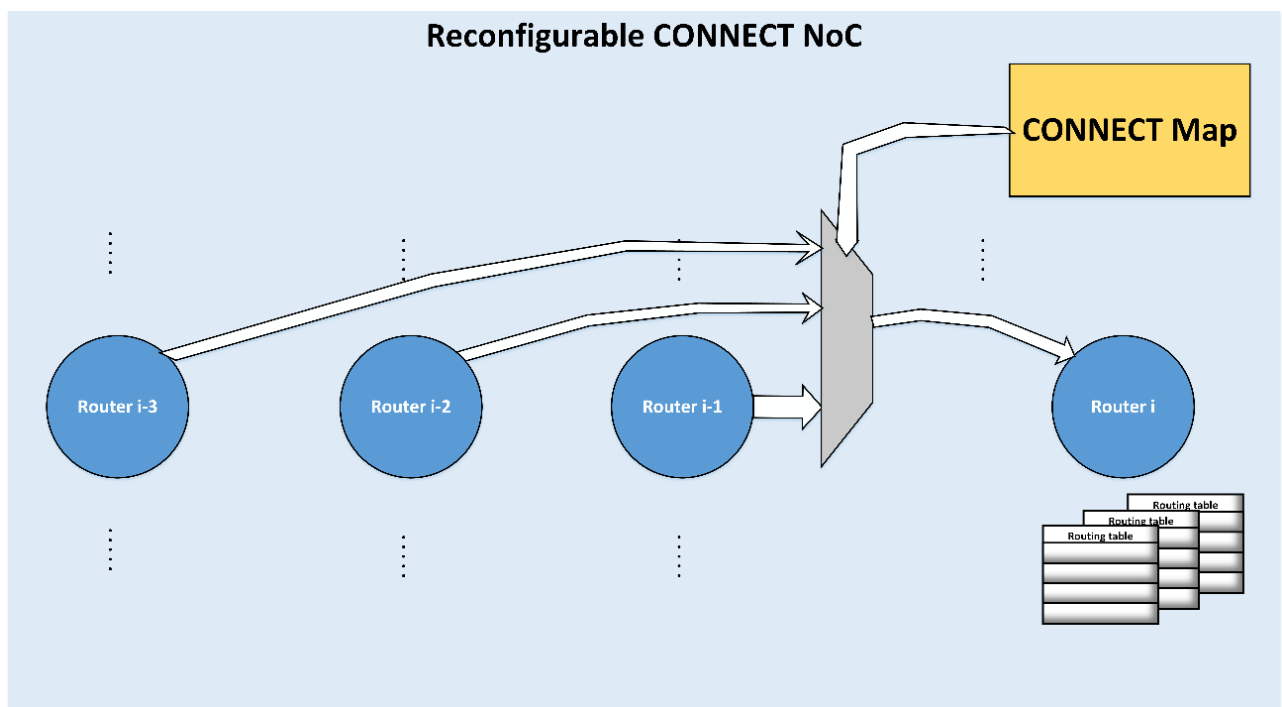


Figure 4.3: Reconfigurable CONNECT NoC structure

Those changes permitted the network to be fully customizable according to the desired requirements during the runtime. In Figure 4.4(a), the 4x4 network is fully functioning and can be reconfigured during the runtime to any network structure such as the 3x3 network in Figure 4.4(c) by disabling the 7 edge routers, or the 2x3 network as depicted in Figure 4.4(d) by disabling the 10 edge routers, or even any irregular form as shown in Figure 4.4(b).

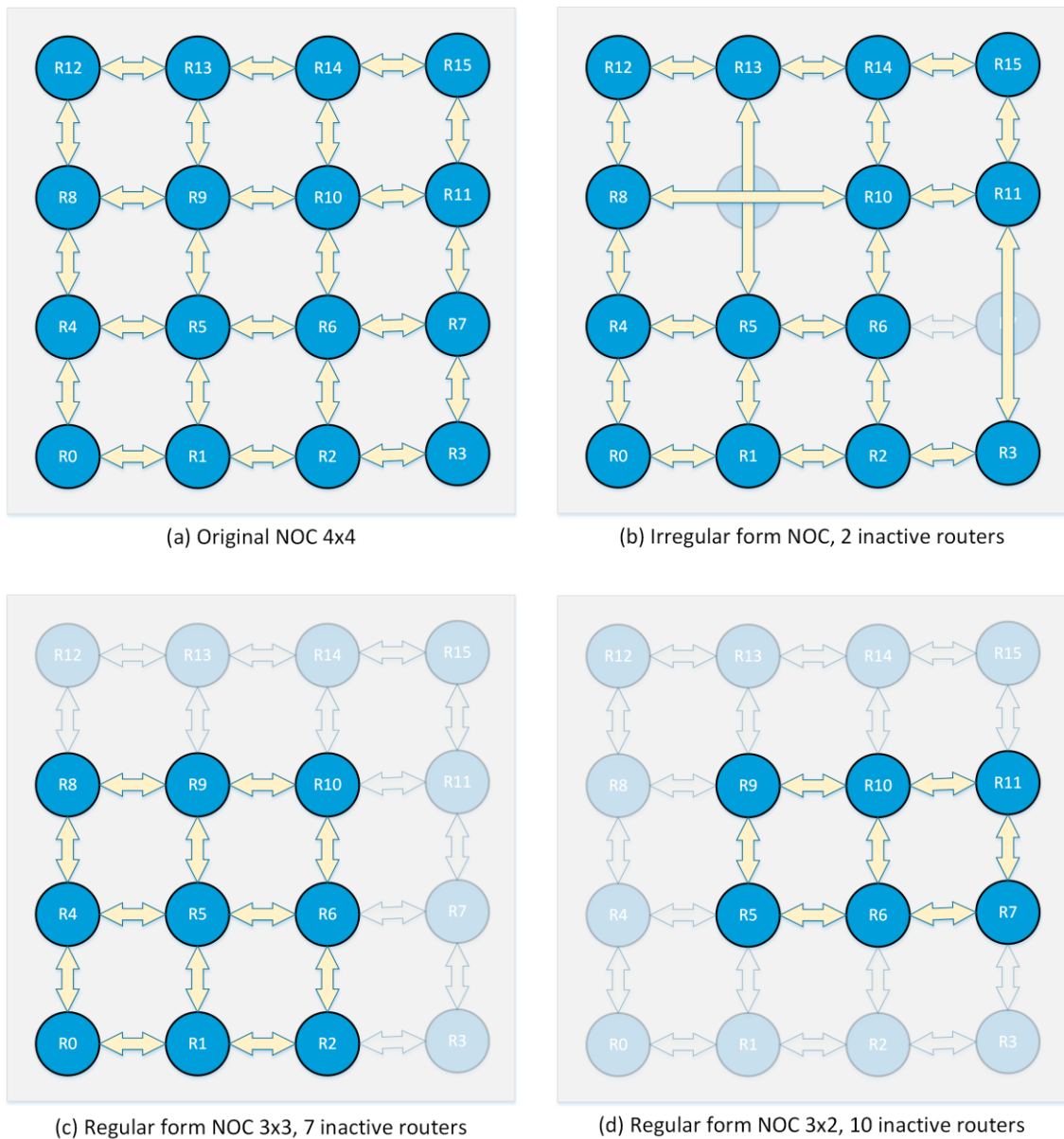


Figure 4.4: Reconfigurable examples of CONNECT NoC during runtime [29]

4.3. Test Environment structure

The modified CONNECT RTL is placed inside an environment driven from [25] with the structure shown in Figure 4.5. Each network node is connected against a packet generation element and a credit handling element. The main function of the packet generator is to provide each node with input flits targeting a totally randomized destinations and virtual channels. The packet generator takes into consideration the current active nodes, the desired traffic density and the available virtual channels. The credit element monitors the input and output packets to and from every virtual channel. This credit is considered as an indication to the available space in each channel buffer.

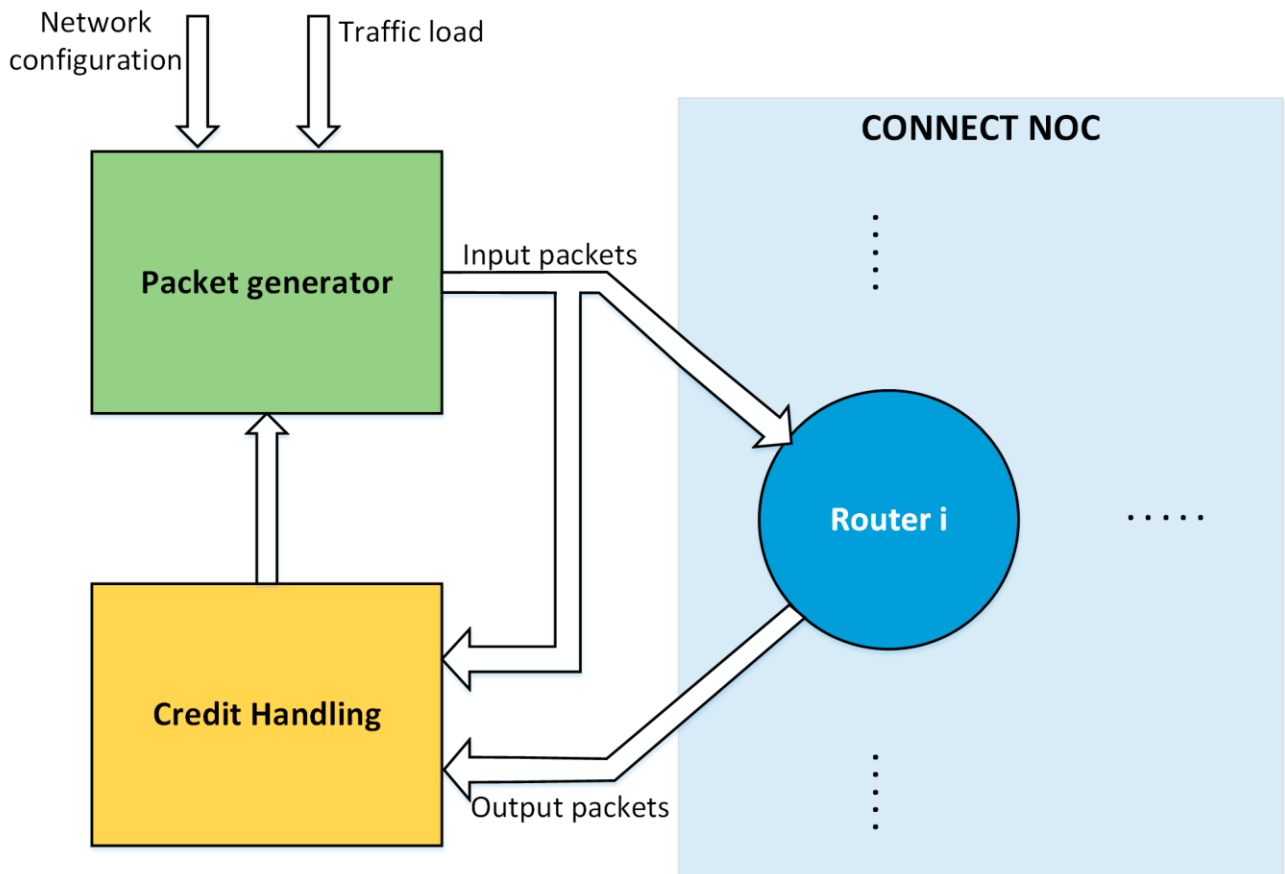


Figure 4.5: Environment structure of CONNECT NoC during runtime [29]

4.4. Reconfiguration tool

In general, the aim of all those modifications is to develop a tool providing the recommended network structures based on the planned benchmarks to be used during the runtime. The referred tool is developed as shown in Figure 4.6.

The tool requires all the input user benchmarks with their corresponding traffic densities and performance requirements. The output of the tool is the recommended network structure for each benchmark to be switched during reconfiguration. Table 4.1 shows an example for how the tool outputs could recommend different network structures leading to area saving when switching from a benchmark to another.

Table 4.1: Reconfiguration tool output for different user benchmarks

Benchmark	Target Throughput	Expected Traffic	Network Configuration	Virtual Channel	Buffer Depth
Benchmark 1	0.68	80%	mesh 4x4	2	64
Benchmark 2	0.53	55%	mesh 4x3	2	32
Benchmark 3	0.4	40%	mesh 3x3	2	16
Benchmark 4	0.1	20%	mesh 2x2	2	4

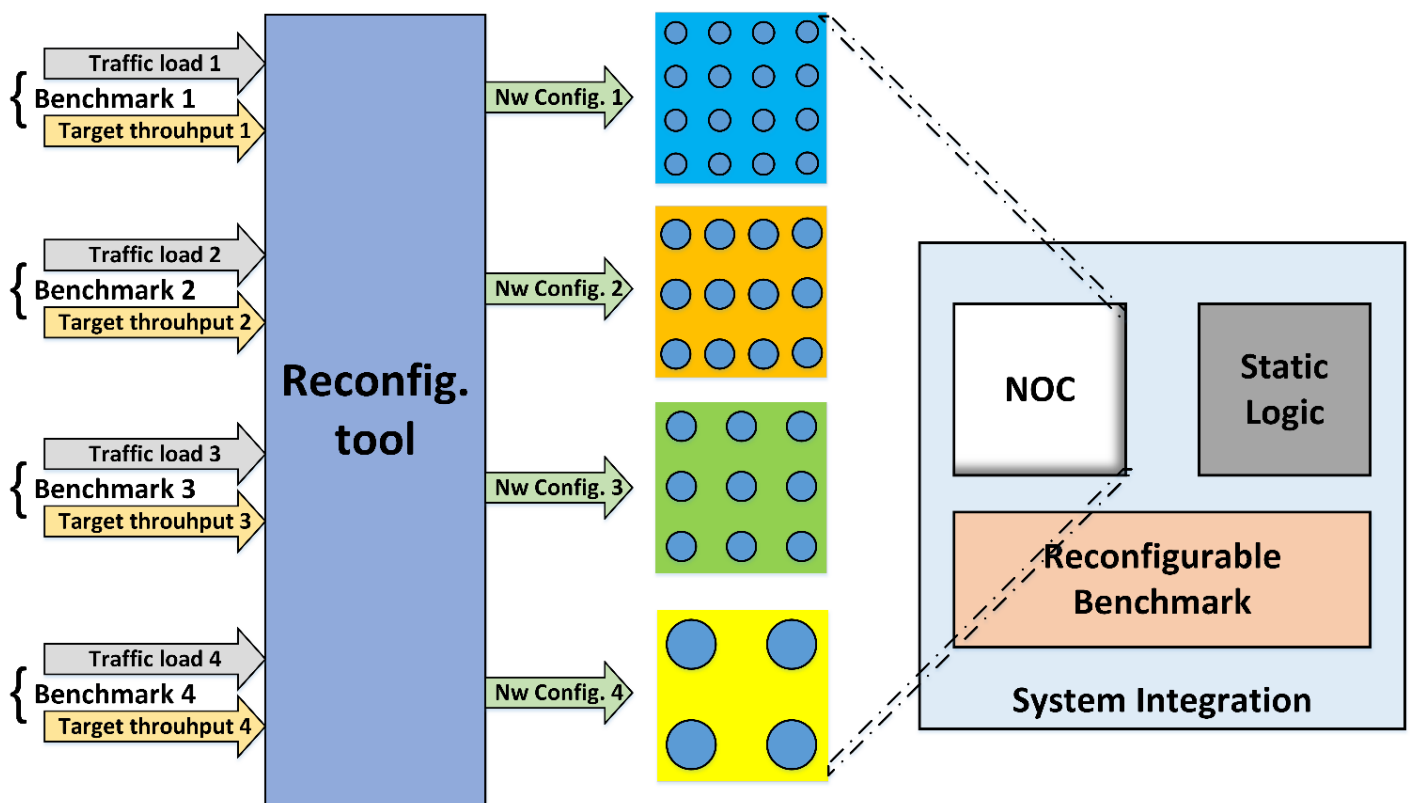


Figure 4.6: Reconfiguration tool structure [29]

The configuration tool is developed using Python as a scripting language. The criteria is based on searching for the best fitting network within the evaluation results. The required throughput and the expected traffic load are the two inputs for the reconfiguration tool from which the tool use to list all the fitting networks in an output file.

The reconfiguration tool interface with the user is available in the batch mode and the GUI mode as well.

In Batch mode, the python script “*search.py*” is called through the shell passing the expected traffic and required throughput. Figure 4.7 shows an example on the tool interface in batch mode with 80% expected traffic and 0.59 pkts/cycle/node required throughput.

```
\Python> python .\search.py 80 0.59
\Python>
```

Figure 4.7: Reconfiguration tool interface in Batch mode

In GUI mode, the python script “*gui.py*” is called through the shell letting a user interface initiates. Figure 4.8 shows the GUI interface of the tool. The GUI interface lets the user parse the expected traffic and the required throughput.

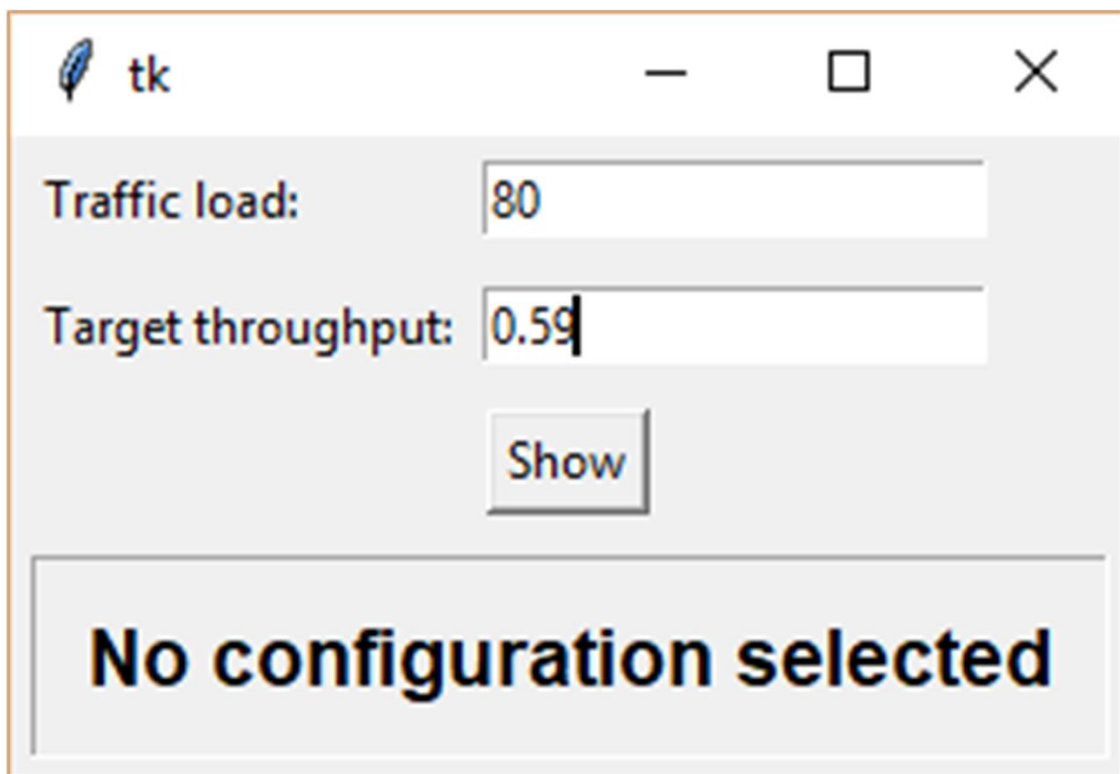


Figure 4.8: Reconfiguration tool interface in GUI mode

The output of the tool in both modes is a list of all network configurations that are fitting the requirements. The output configuration list of the last example is shown in Figure 4.9.

```
NW= 4, VC= 2, BD= 4, PDR= -, which router= none, Config= ffff
NW= 4, VC= 2, BD= 8, PDR= -, which router= none, Config= ffff
NW= 4, VC= 2, BD= 8, PDR= 0, which router= corner, Config= fffe
NW= 4, VC= 2, BD= 8, PDR= 12, which router= corner, Config= efff
NW= 4, VC= 2, BD= 8, PDR= 15, which router= corner, Config= 7fff
NW= 4, VC= 2, BD= 8, PDR= 1, which router= edge, Config= fffd
NW= 4, VC= 2, BD= 16, PDR= -, which router= none, Config= ffff
NW= 4, VC= 2, BD= 16, PDR= 0, which router= corner, Config= fffe
NW= 4, VC= 2, BD= 16, PDR= 3, which router= corner, Config= fff7
NW= 4, VC= 2, BD= 16, PDR= 12, which router= corner, Config= efff
NW= 4, VC= 2, BD= 16, PDR= 15, which router= corner, Config= 7fff
NW= 4, VC= 2, BD= 16, PDR= 1, which router= edge, Config= fffd
```

Figure 4.9: Reconfiguration tool output in Batch and GUI modes

For the detailed use of the Reconfiguration tool, please refer to Appendix A: Reconfiguration tool User Manual.

Chapter 5 : Impact of Dynamic reconfiguration on Network on Chip performance

5.1. Environment Setup

The previously referred CONNECT environment in the last Chapter is generalized having the capability to test different network configurations with various traffic densities ranging from 5% to 100%. The environment handles different virtual channels ranging from 2 to 8, different buffer depths ranging from 4 to 64. Moreover, the DPR evaluation is applied to all the possible networks nodes starting from all the 16 nodes functioning (4x4 mesh) up to disabling 14 nodes and keeping only two nodes functioning (2x1 mesh).

The network evaluation is based on the throughput as a performance metric and it plays an important role in deciding the optimal network configuration.

The throughput is calculated after flooding the network with input packets according to the desired traffic density. Then, monitoring the number of output packets per cycle per node.

When many routers are disabled during network reconfiguration, the network size become smaller in size. This leads into area saving on the expense of performance degradation as the neighbor routers are required to handle higher traffic load. The network reconfiguration is varying from a full network with highest performance to an inadequate network with the lowest performance. And the benchmark desired performance plays an important role in deciding the best configuration that meets this requirement. The selection criteria is prioritizing the less significant area size which corresponds to the smallest network possible.

The following subsections discuss the impact of different configurations on the performance. And, provides an in-depth view on how each configuration works individually on boosting the network overall performance.

5.2. Network topology DPR Evaluation

From the early evaluation results, the regular forms of the network give better results than the irregular ones for the same number of routers. For the mesh 3x3 network for example, it consists of 9 routers and gives better results than any irregular network form with the same area size (9 routers). Accordingly, the results shown here are mainly focusing on the regular network forms such as: mesh 4x4, mesh 4x3, mesh 3x3, mesh 3x2, mesh 2x2, and mesh 2x1.

Figure 5.1 shows the performance of the specified networks with credit-based flow control, two virtual channels, buffer depth of four and under a traffic load ranging from 5% to 100%. It is clear that lowering the network size using reconfiguration leads to less significant throughput rates. This is because the remaining active nodes after

reconfiguration are required to compensate the absence of the reconfigured inactive nodes.

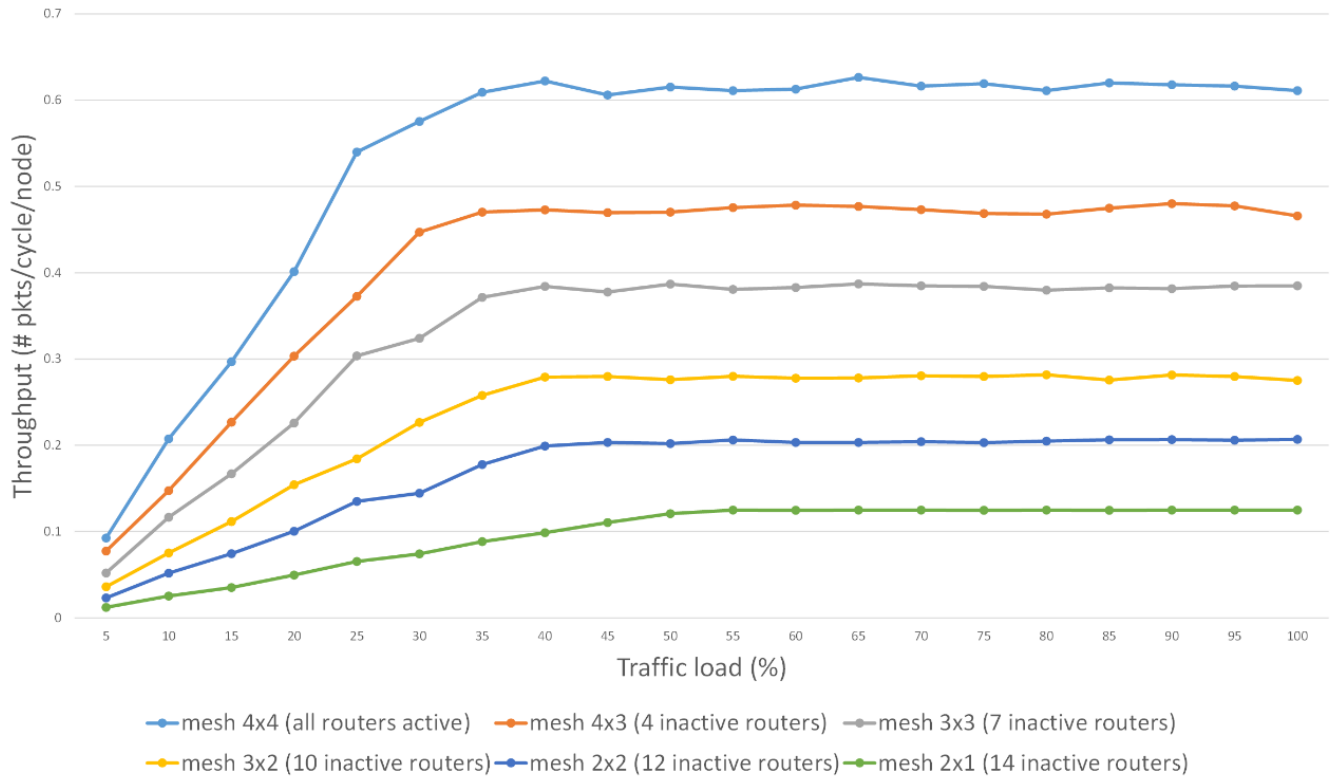


Figure 5.1: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=2 – BD=4

In general, shrinking the network size using DPR into a smaller topology degrades the network performance. However, DPR saves significant area for other logic to be used. This is beneficial with the applications that does not require high performance at the moment and switches into a smaller network topology.

The network topology DPR evaluation result of a network with 4 virtual channels, Credit-based flow control, and different buffer depths are shown in Figures 5.2, 5.3, 5.4, 5.5, and 5.6. Figures 5.2, 5.3, 5.4, 5.5, and 5.6 correspond to buffer depths of 4, 8, 16, 32, and 64 respectively.

In addition, the network topology DPR evaluation result of a network with 8 virtual channels, Peek-based flow control, and different buffer depths are shown in Figures 5.7, 5.8, 5.9, 5.10, and 5.11. Figures 5.7, 5.8, 5.9, 5.10, and 5.11 correspond to buffer depths of 4, 8, 16, 32, and 64 respectively.

It is obvious in all the network configurations that shrinking the network size always results in throughput degradation.

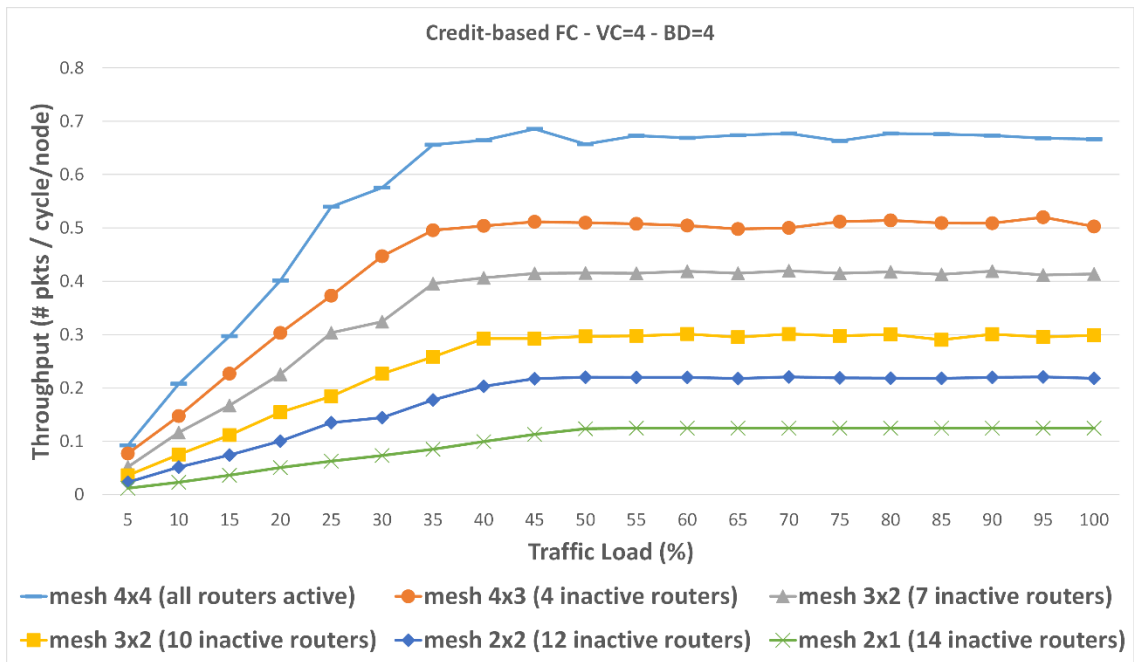


Figure 5.2: CONNECT Performance all PDR configurations – Credit-based flow control – VC=4 – BD=4

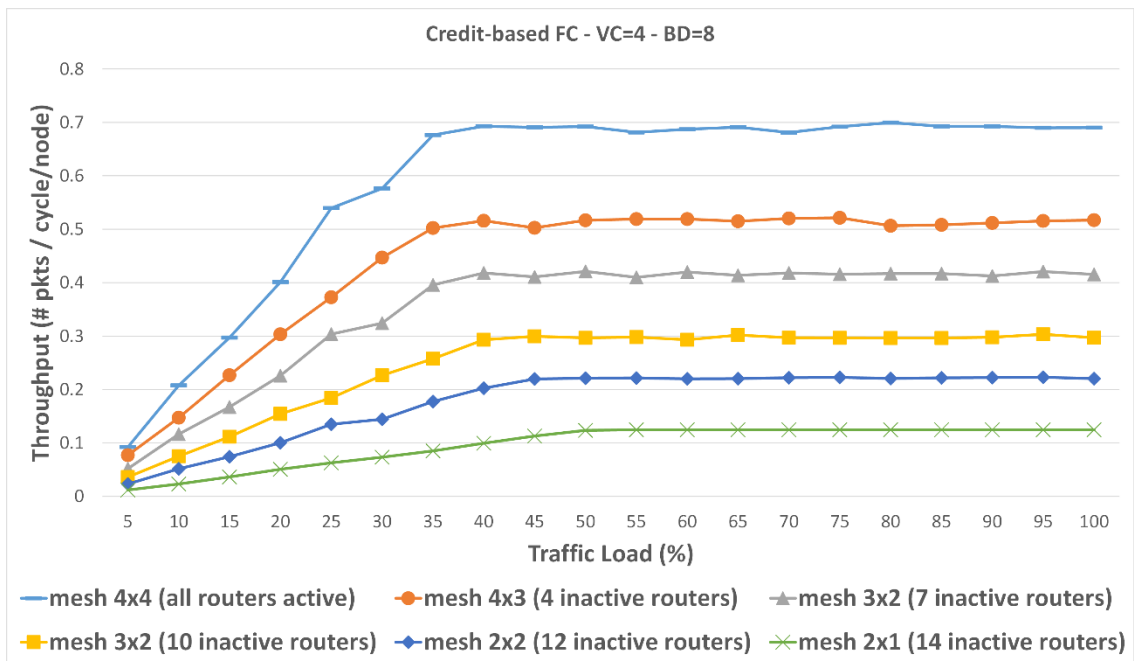


Figure 5.3: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=8

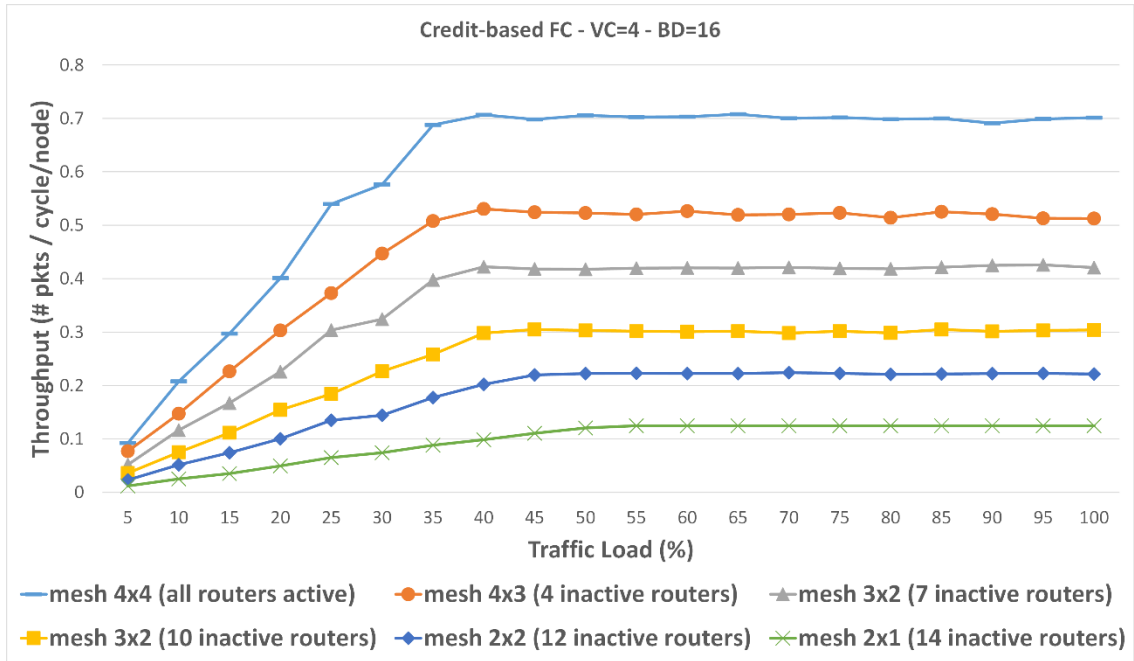


Figure 5.4: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=16

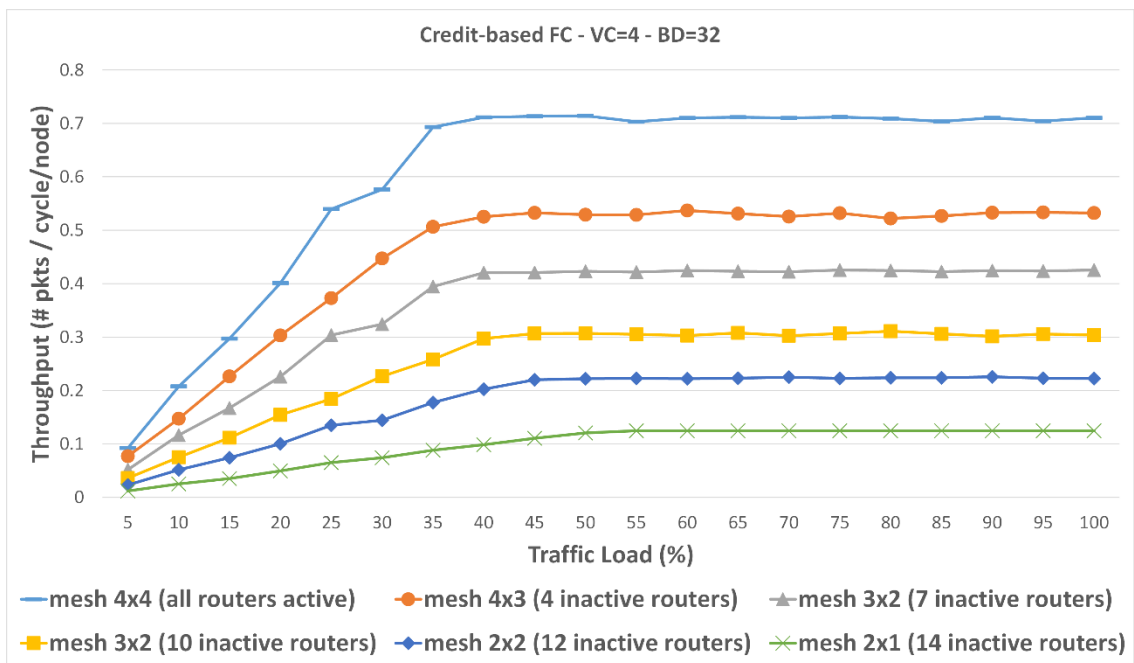


Figure 5.5: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=32

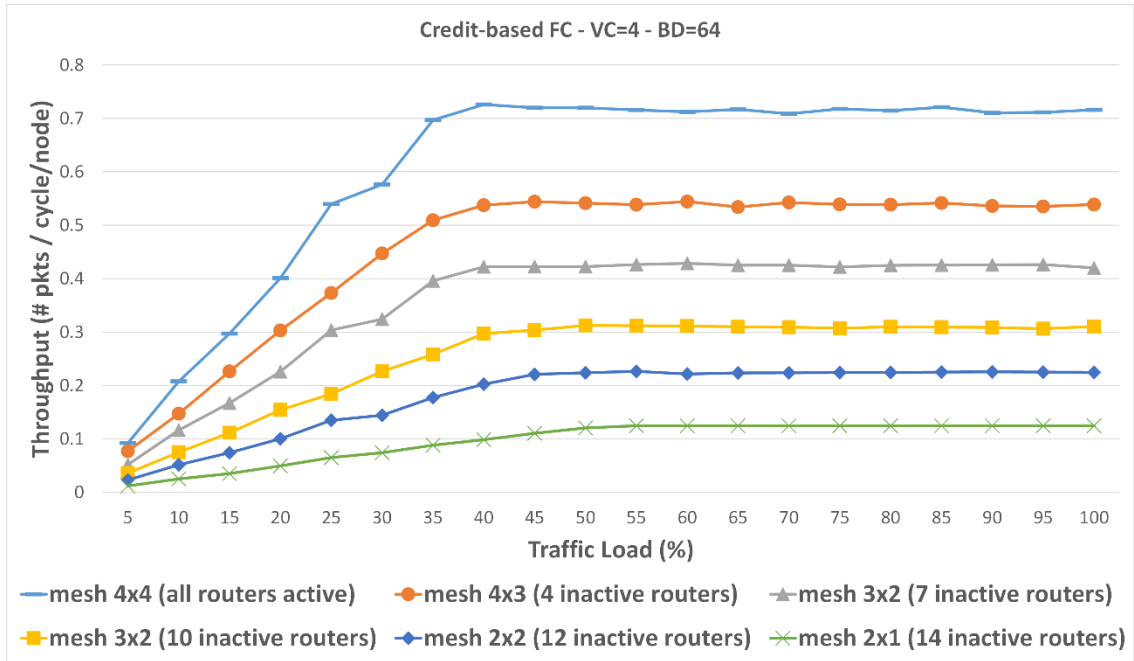


Figure 5.6: CONNECT throughput of all PDR configurations – Credit-based flow control – VC=4 – BD=64

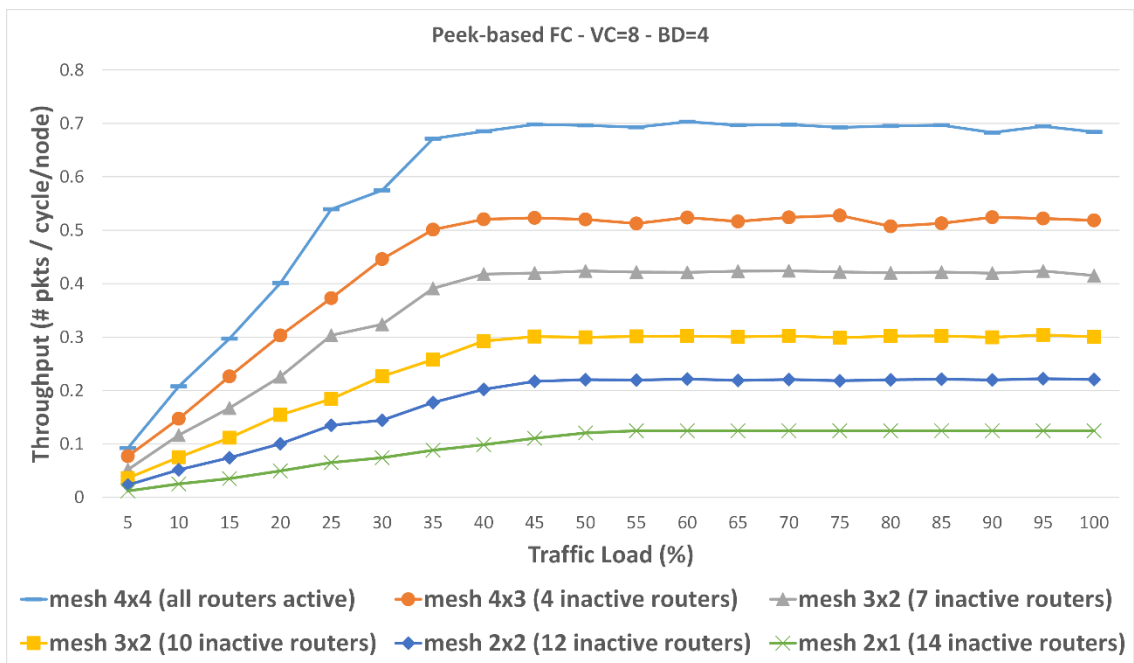


Figure 5.7: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=4

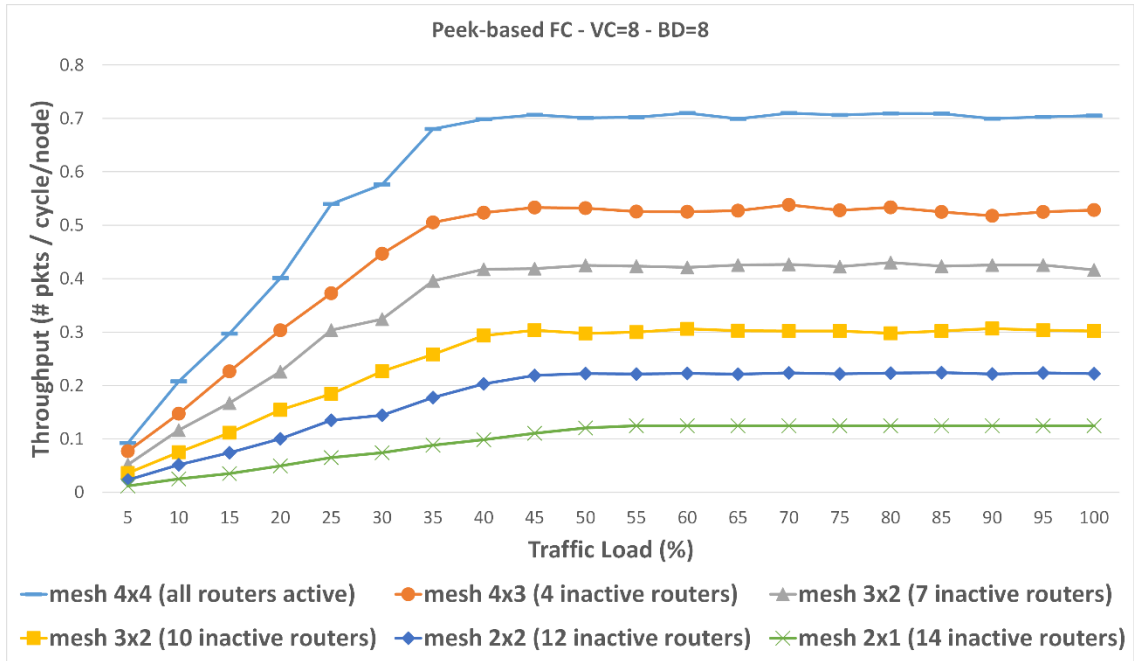


Figure 5.8: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=8

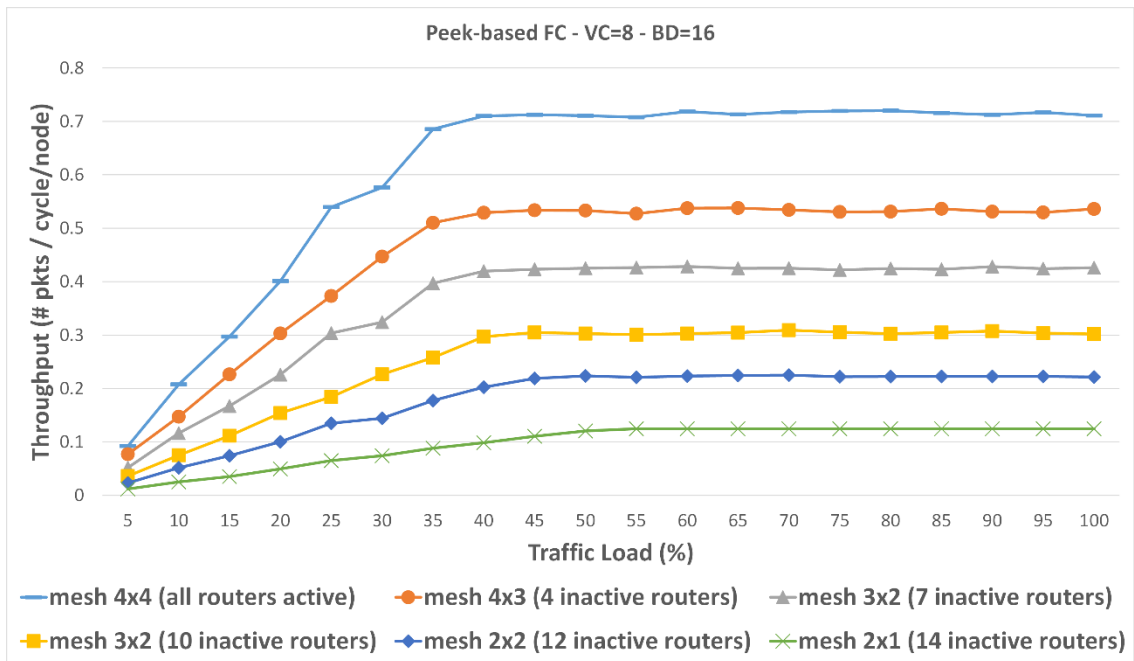


Figure 5.9: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=16

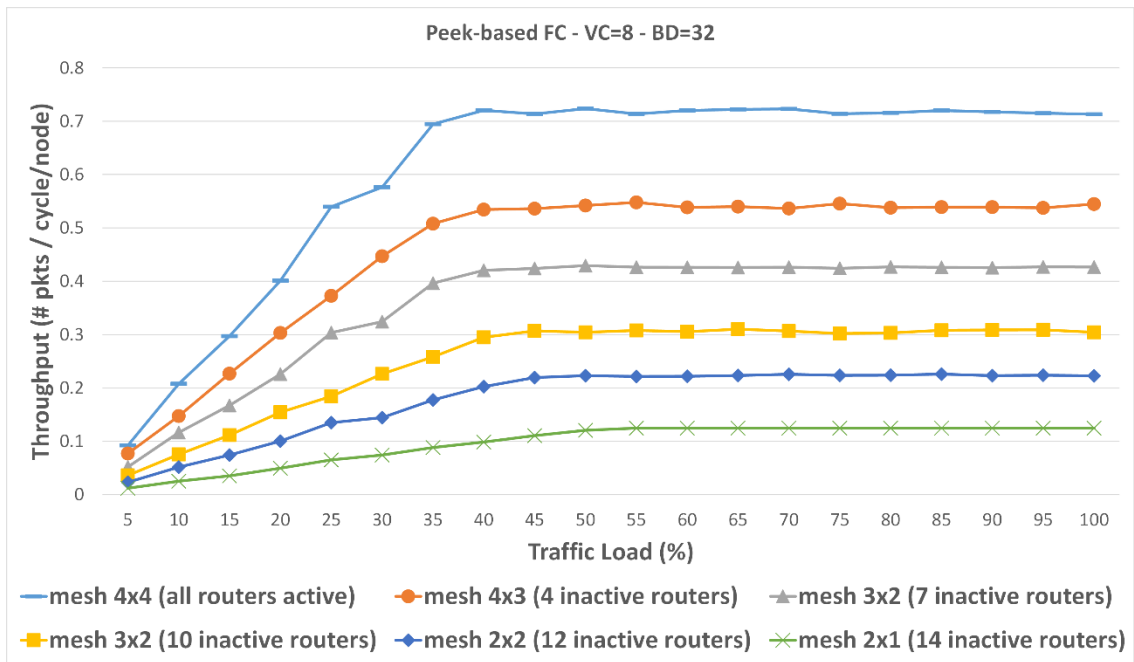


Figure 5.10: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=32

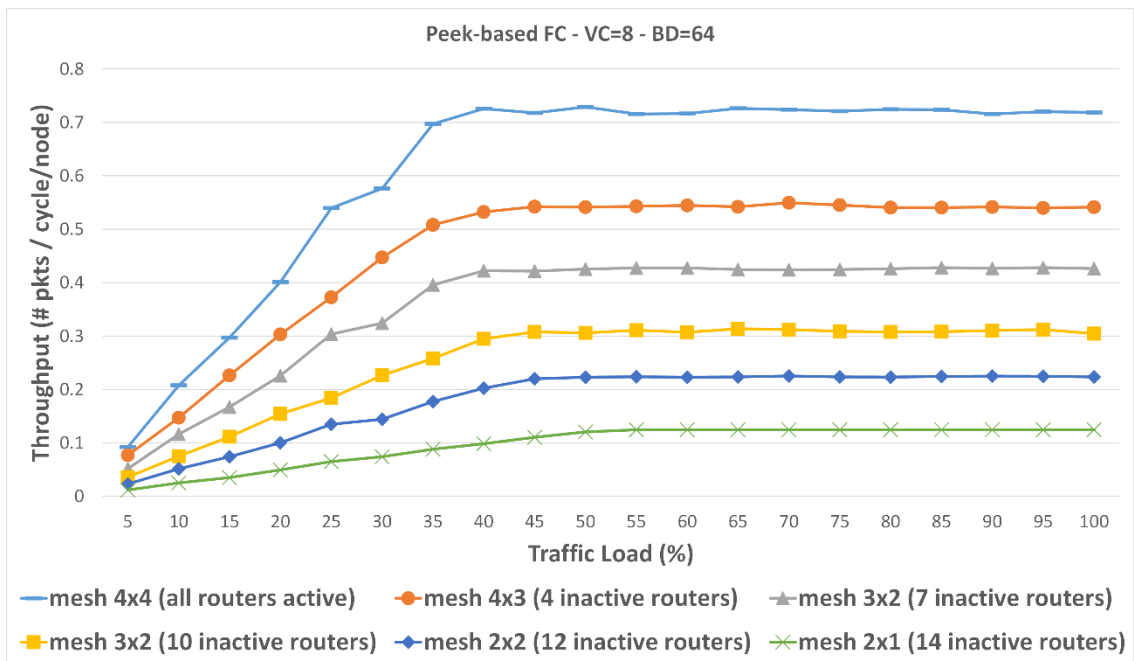


Figure 5.11: CONNECT throughput of all PDR configurations – Peek-based flow control – VC=8 – BD=64

5.3. Buffer Depth DPR Evaluation

Applying DPR into the buffer depth as a network parameter impacts the network performance differently. The performance shown here is for the buffer depth of each router inside the 4x4 specified network with credit-based and peek-based flow controls, two virtual channels, buffer depth of four and under a traffic load ranging from 5% to 100%.

In general, increasing the buffer depth impacts the performance of all the network configurations positively. This is because each node become capable of receiving and handling more packets at the expense of significant area overhead.

Figure 5.12 represents the performance of the buffer depth as a variant on a mesh 4x3 network after reconfiguring 4 nodes from the original mesh 4x4 network. The network performance is the same with the larger buffer depth at the low traffic densities and is enhanced with the larger buffer depth at the high traffic densities. The performance of the buffer depth of 64 for example is more significant than the buffer depth of 32.

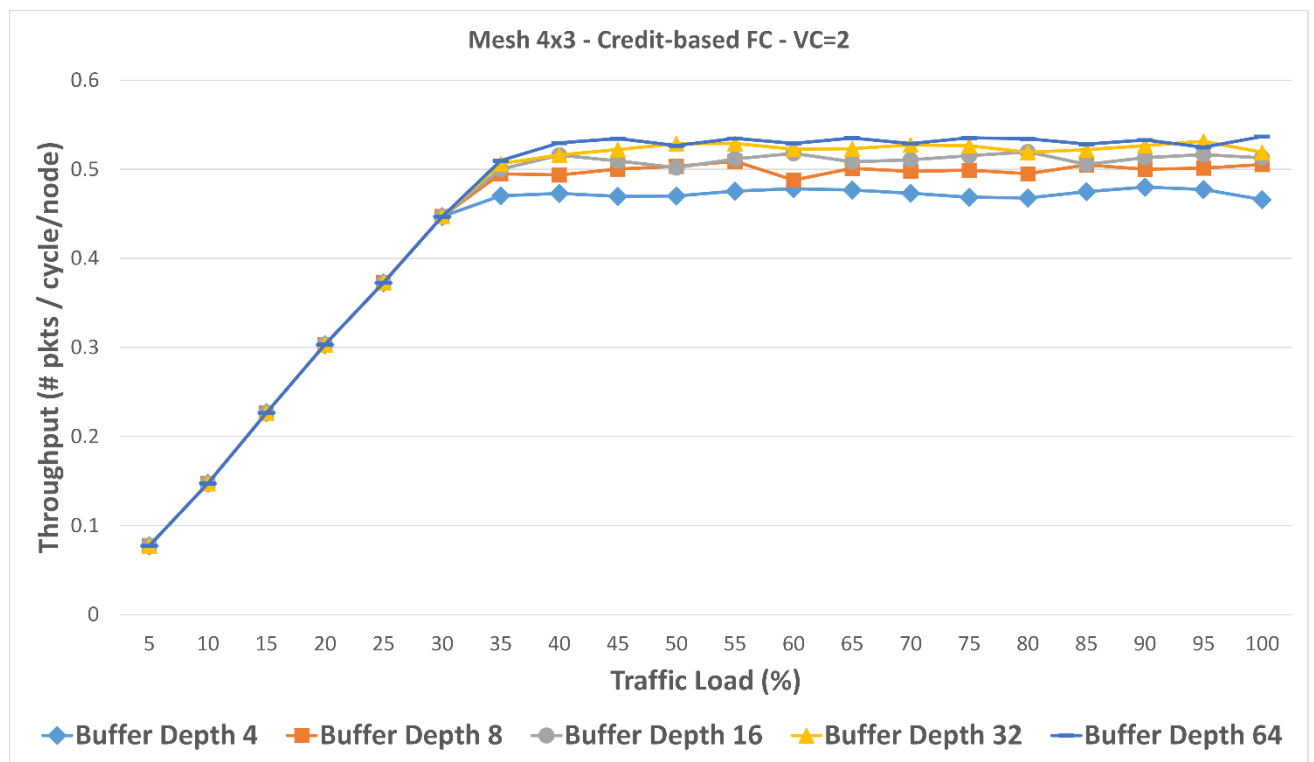


Figure 5.12: CONNECT throughput of all BD configurations – Mesh 4x3 – Credit-based flow control – VC=2

In Figure 5.13, switching to a larger buffer depth such as 16, 32, or 64 has a small impact on the performance of the relatively small networks (such as mesh 2x2 – 12 routers are reconfigured). This is due to the network low latency as the packets don't consume much time till reaching the destination.

Additionally, the larger buffer depth with the mesh 2x2 networks results in a bit different performance impact. The throughput impact is almost the same at the low traffic densities till the traffic density of 40%. However, the complete mesh 4x4 network has the same performance till a traffic load of nearly 30% as shown in Figure 5.12.

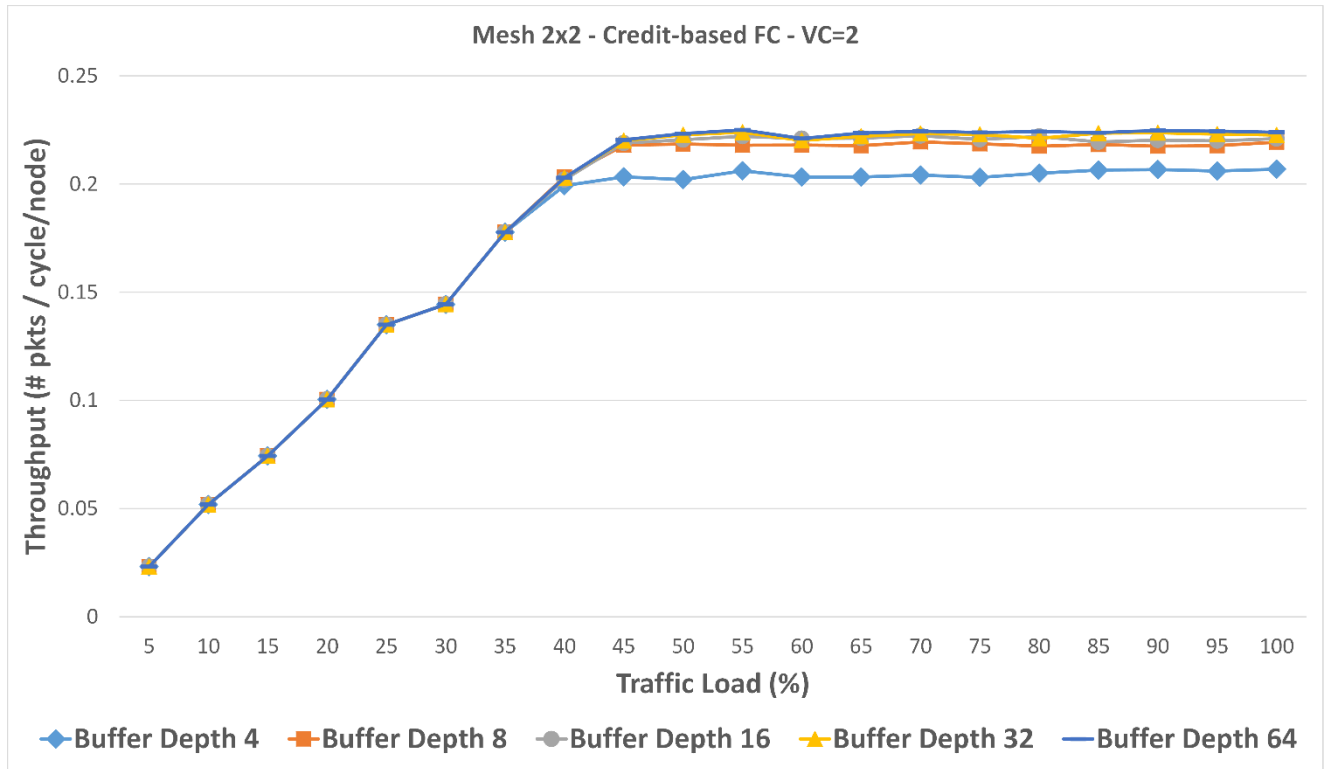


Figure 5.13: CONNECT throughput of all BD configurations – Mesh 2x2 – Credit-based flow control – VC=2

Figure 5.14 shows the throughput of mesh 2x1 network using various buffer depths ranging from four to 64, while keeping the other configurations constant. The buffer depth impact becomes non-noticeable with shrinking the network size. And, this is because most of the buffer depth is not used efficiently especially with the mesh 2x2 and mesh 2x1 network sizes. Using a buffer depth of 64 in this case is a waste of because it has the same performance of using the buffer depth of four.

The larger buffer depth with the smaller network (14 routers are reconfigured) gives a very different performance impact from the relatively large networks. All the throughput values are the same at all traffic densities.

More buffer depth DPR evaluation results are shown are shown in Figures 5.15, 5.16, 5.17, 5.18, 5.19, and 5.20. The results are for a Peek-based flow control, two virtual channels, and a network topology configurations of (Mesh 4x4, 4x3, 3x3, 3x2, 2x2, and 2x1 respectively).

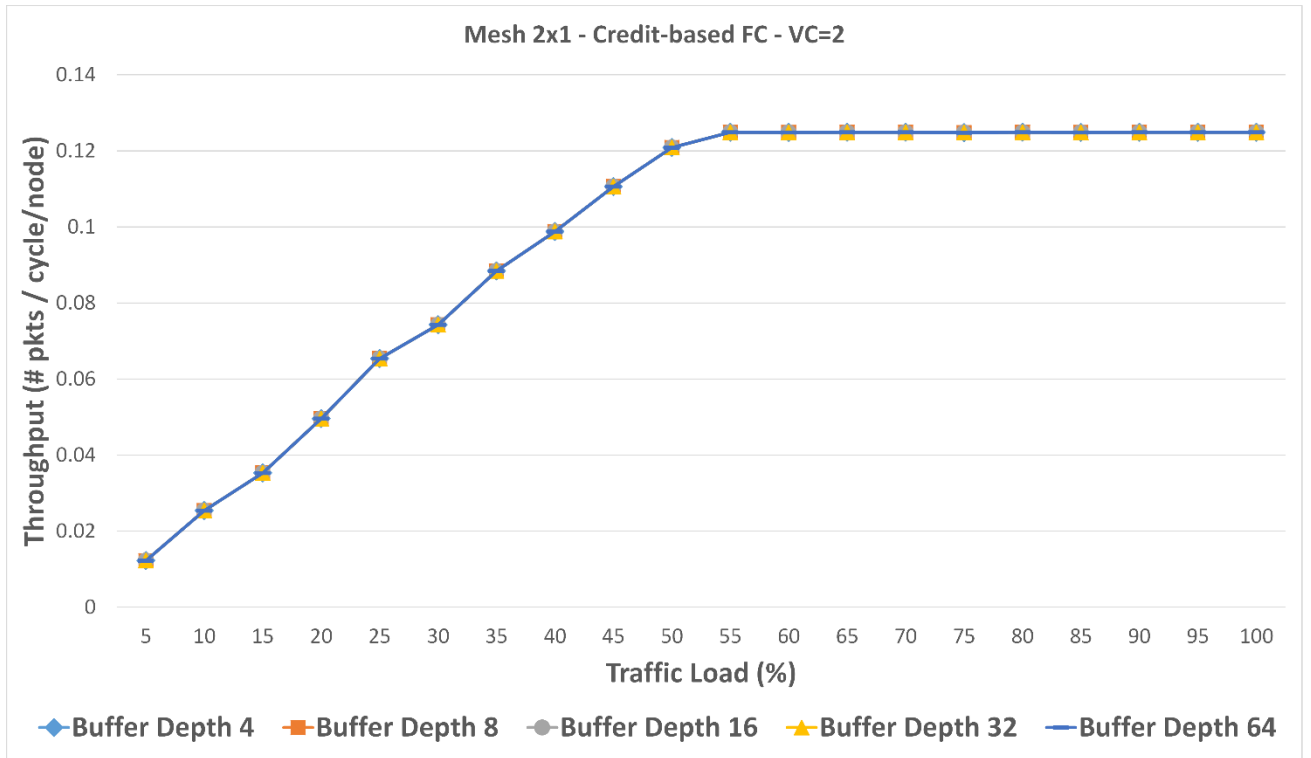


Figure 5.14: CONNECT throughput of all BD configurations – Mesh 2x1 – Credit-based flow control – VC=2

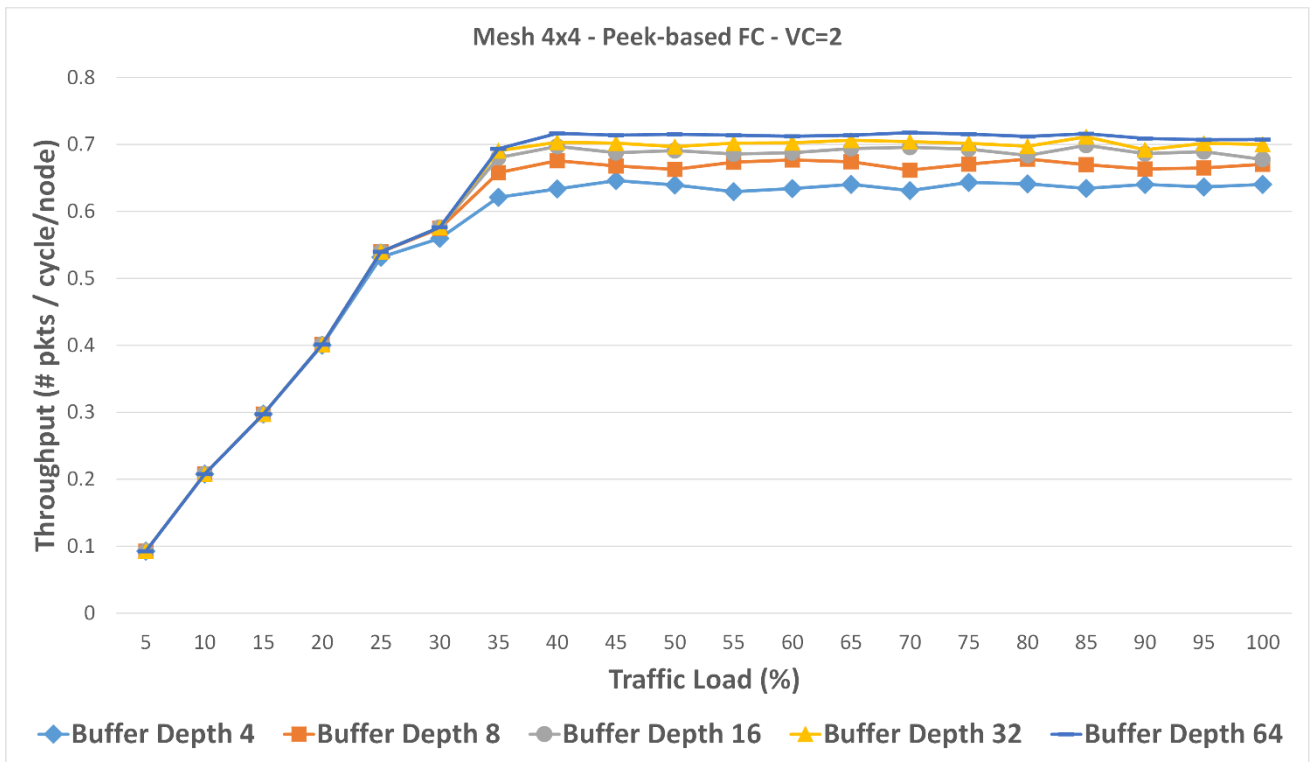


Figure 5.15: CONNECT throughput of all BD configurations – Mesh 4x4 – Peek-based flow control – VC=2

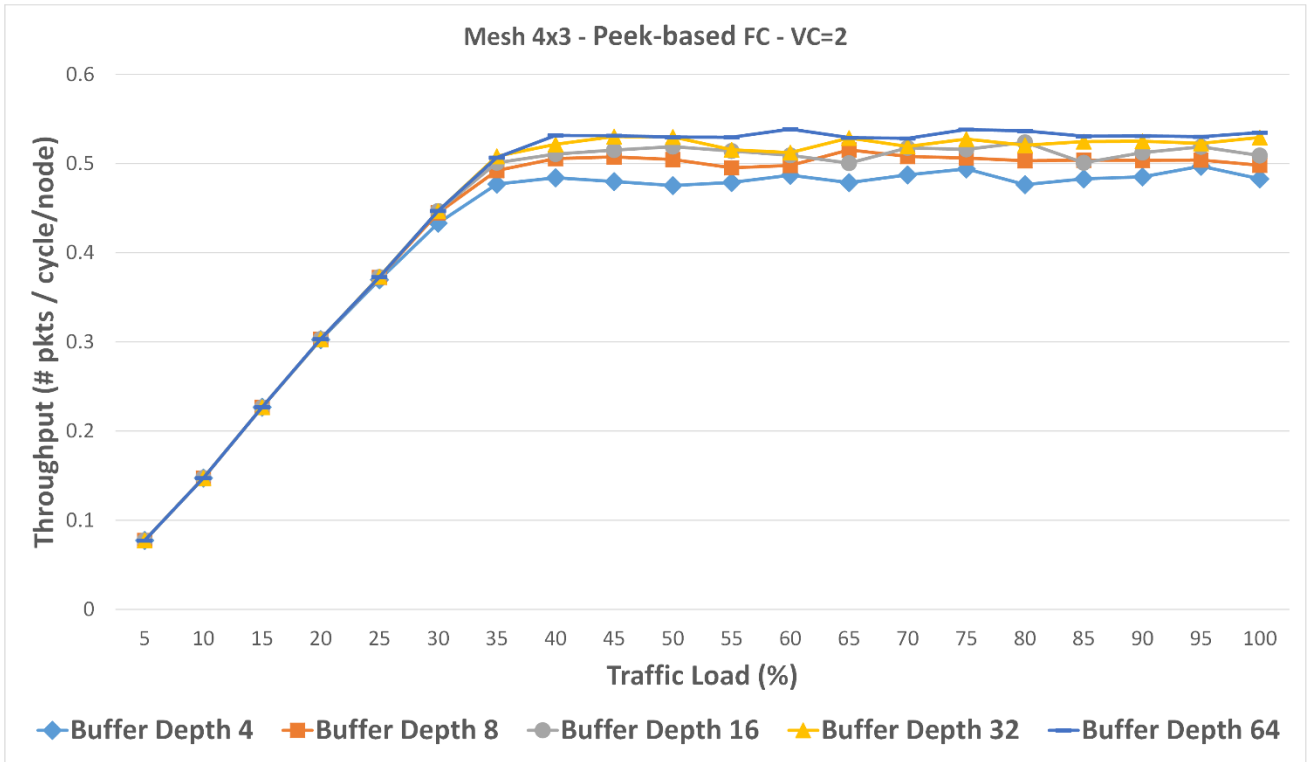


Figure 5.16: CONNECT throughput of all BD configurations – Mesh 4x3 – Peek-based flow control – VC=2

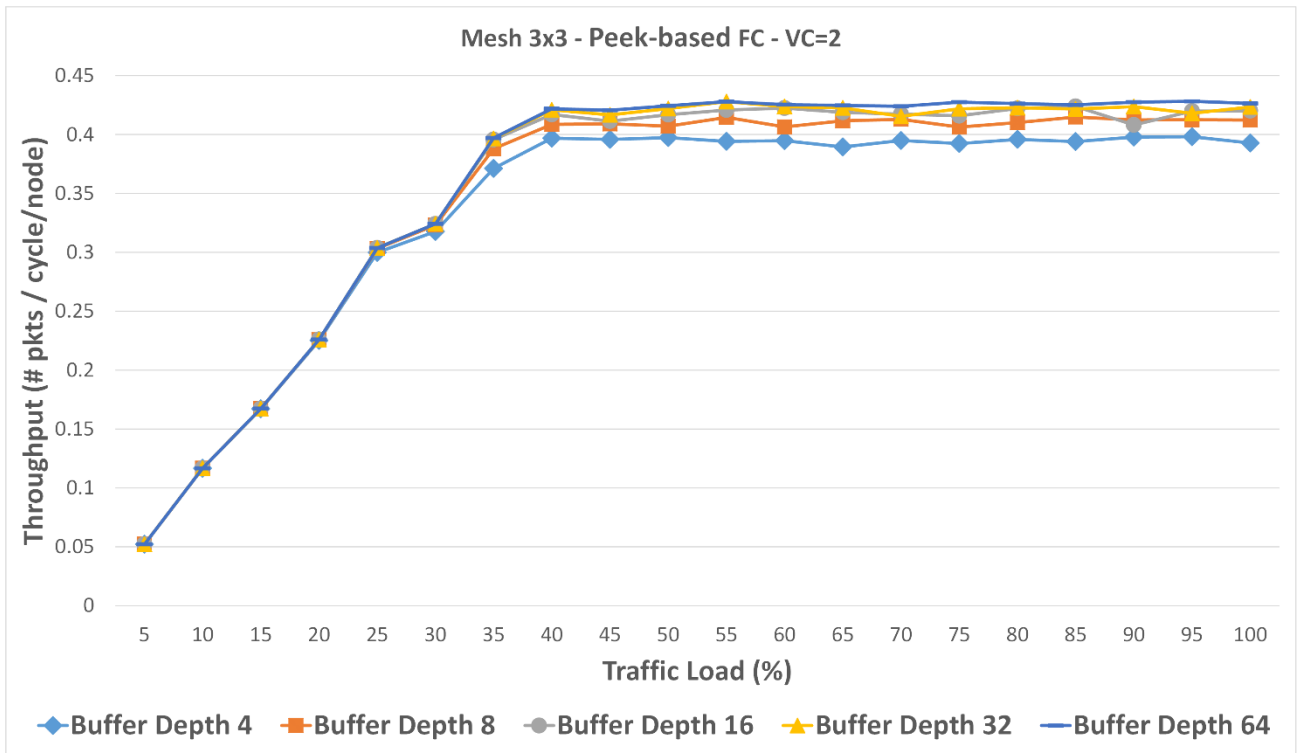


Figure 5.17: CONNECT throughput of all BD configurations – Mesh 3x3 – Peek-based flow control – VC=2

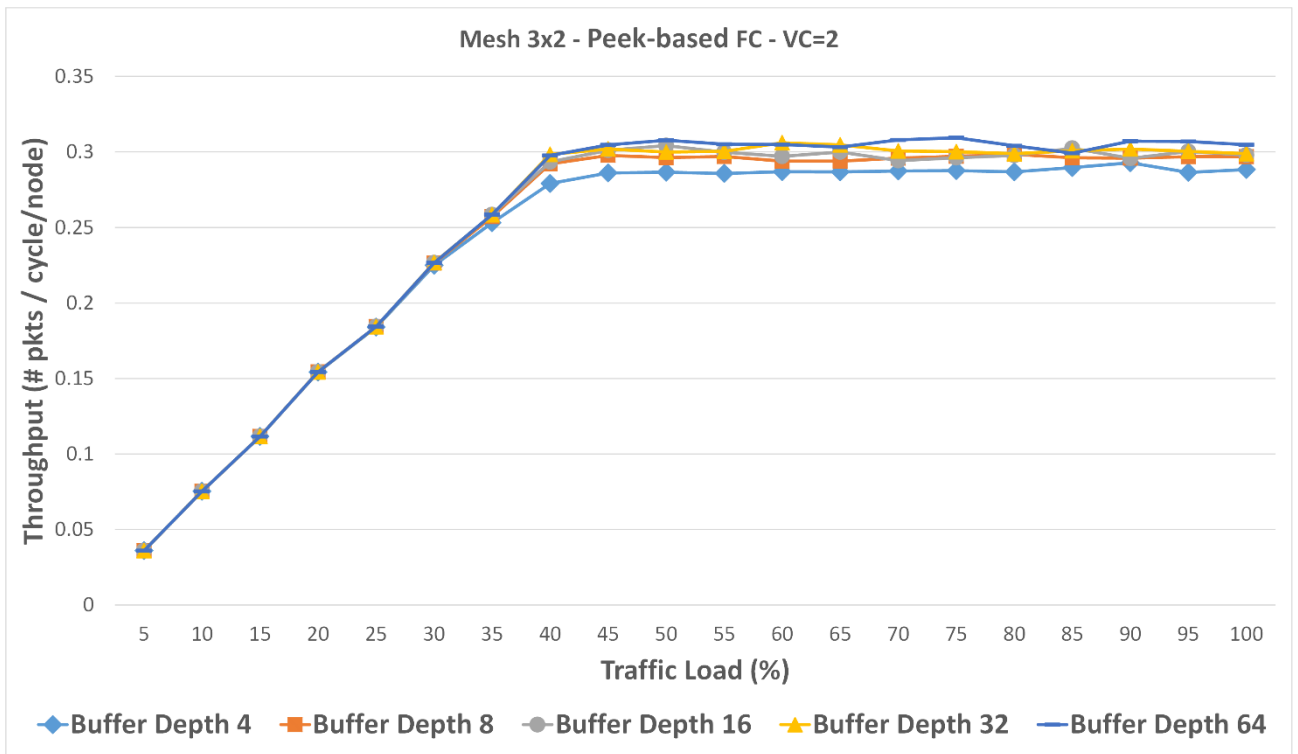


Figure 5.18: CONNECT throughput of all BD configurations – Mesh 3x2 – Peek-based flow control – VC=2

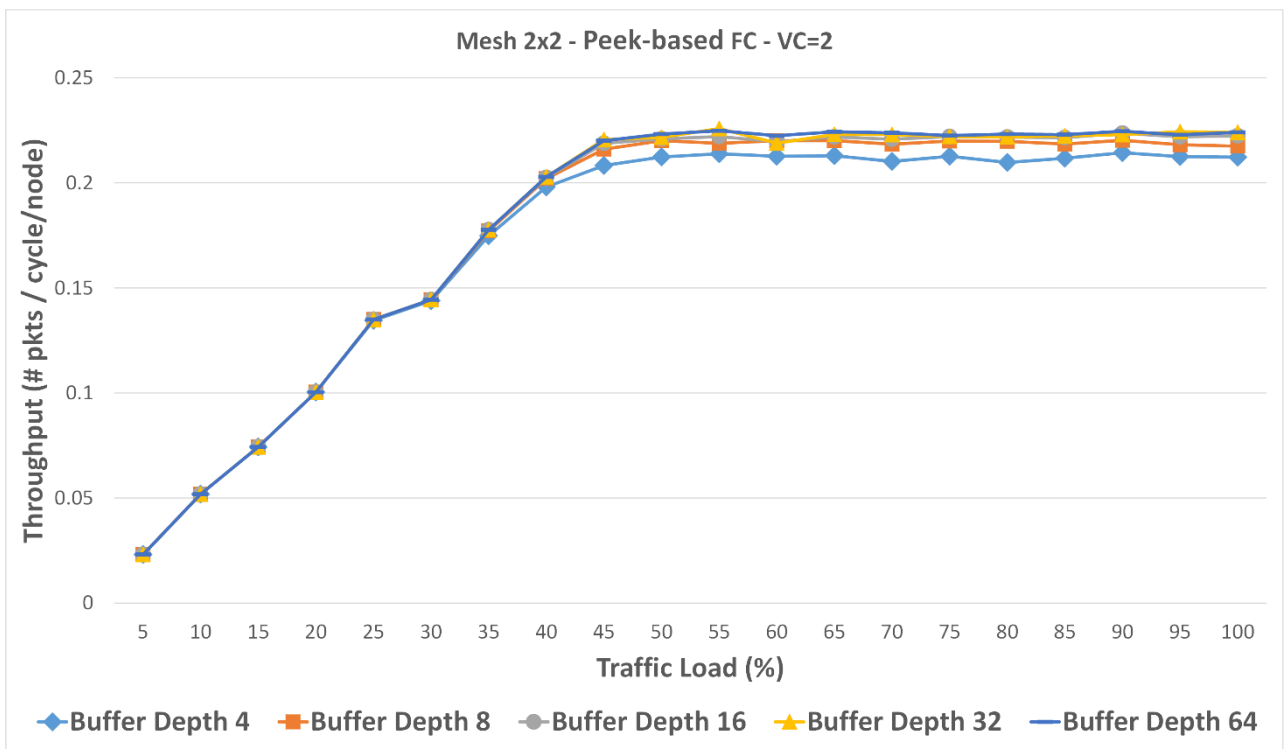


Figure 5.19: CONNECT throughput of all BD configurations – Mesh 2x2 – Peek-based flow control – VC=2

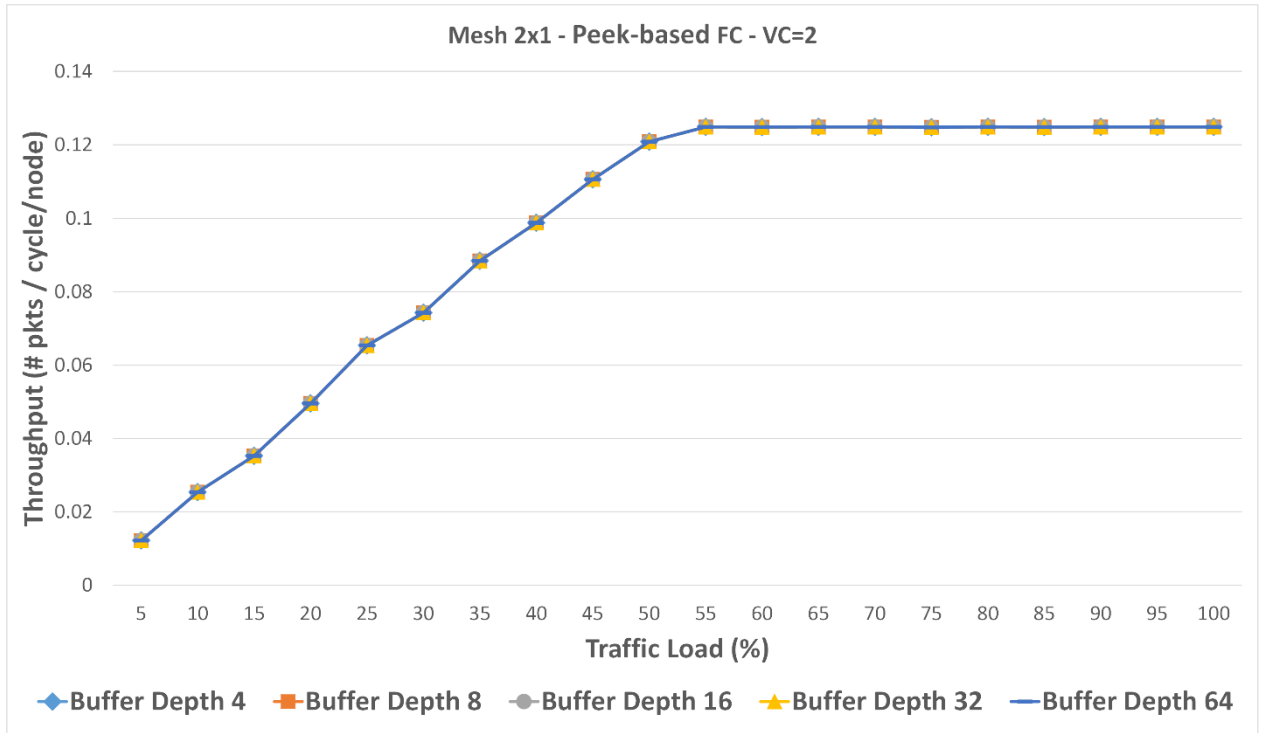


Figure 5.20: CONNECT throughput of all BD configurations – Mesh 2x1 – Peek-based flow control – VC=2

5.4. Virtual Channel DPR Evaluation

Applying DPR into the virtual channel as a network parameter impacts the network performance in a different way than the buffer depth. The performance shown here is for the virtual channels inside the mesh 4x4 network with credit-based and peek-based flow controls, buffer depth of 4, 8, and 64 and under a traffic load ranging from 5% to 100%.

In general, increasing the virtual channels improves the performance of all the network configurations positively. This is because it creates new routing paths in parallel with the original network paths. Accordingly, the network capability to handle and receive packets has increased at the expense of significant area overhead.

Figure 5.21 represents the throughput of a mesh 4x4 network using various virtual channels varying from two to eight, while keeping the other configurations constant. The network performance is the same with the four and eight virtual channels at the low traffic densities till nearly 30% and is enhanced at the high traffic densities above 30%.

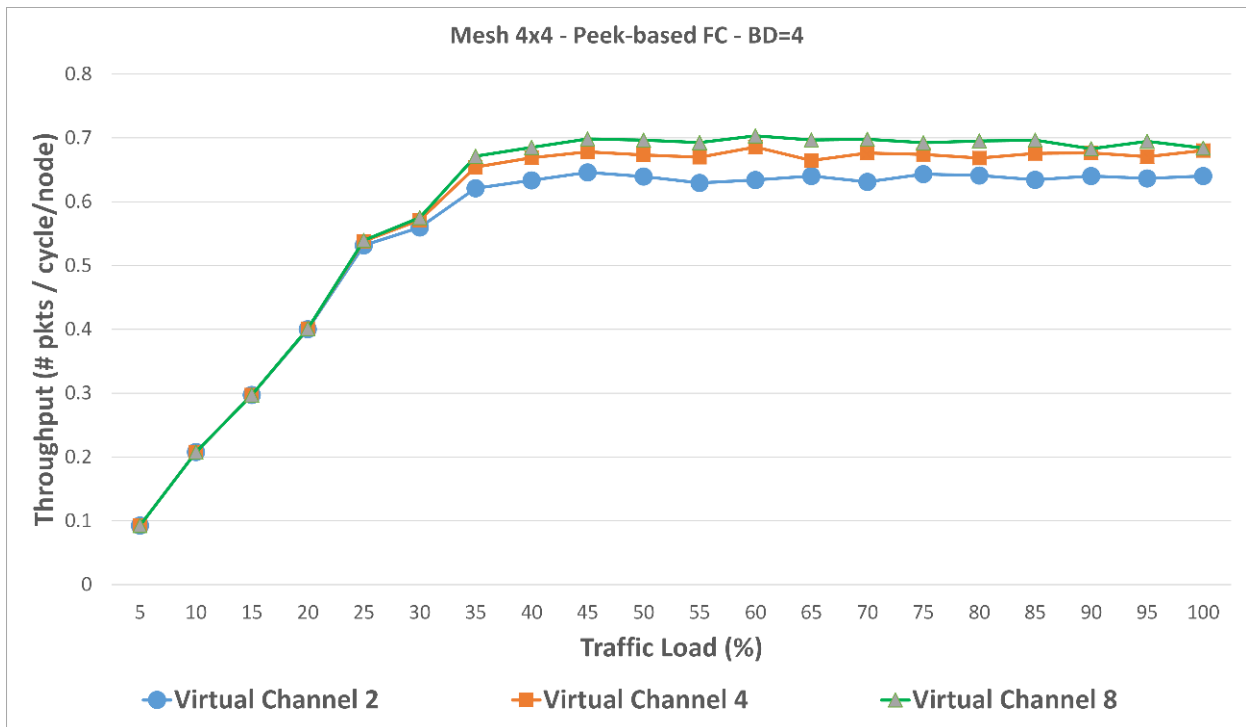


Figure 5.21: CONNECT throughput of all VC configurations – Mesh 4x4 – Peek-based flow control – BD=4

Figure 5.22 shows the virtual channel DPR performance with a mesh 2x1 network using the same other configurations. The virtual channel impact becomes non-noticeable with the shrinking of the network size. And, this is because the low latency of the small networks which lowers the probability of congestion even with high injection rates.

The buffer depth of 64 even with the mesh 4x4 network gives the same performance impact for all virtual channels as in Figure 5.23. With the large buffer depths, increasing the virtual channels become useless.

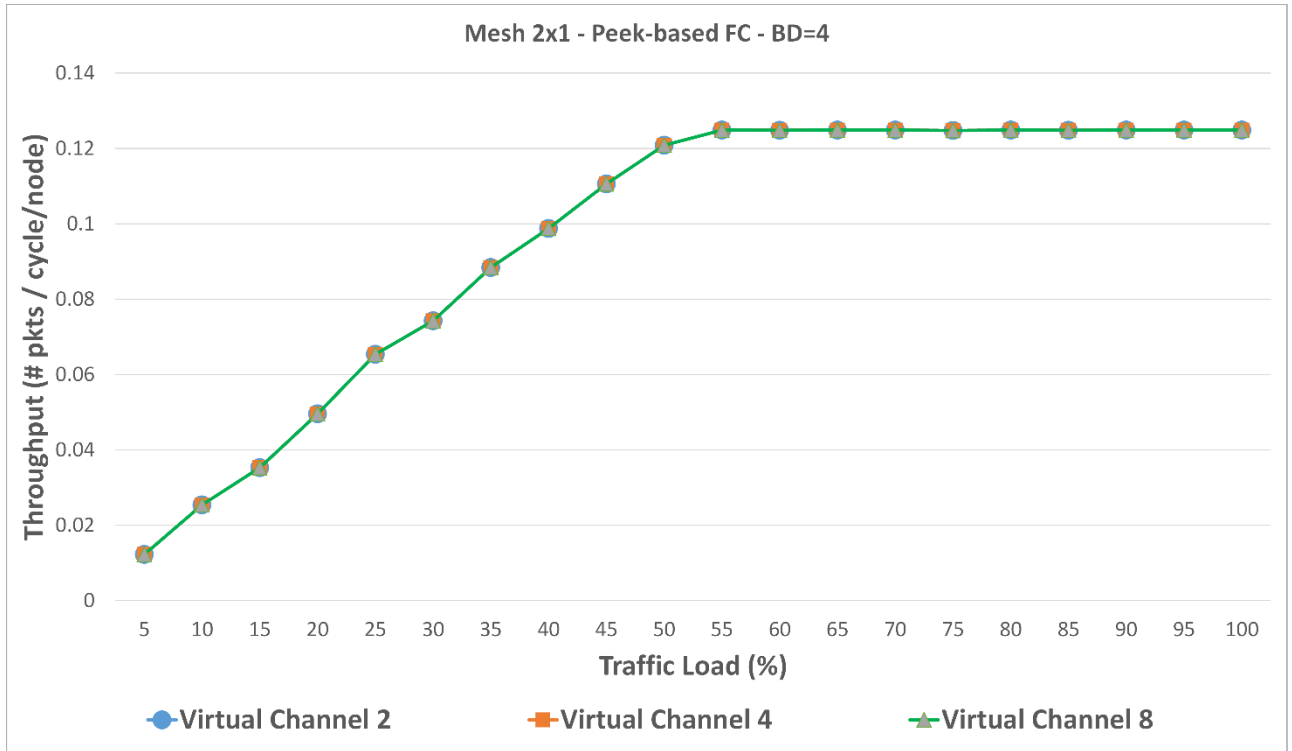


Figure 5.22: CONNECT throughput of all VC configurations – Mesh 2x1 – Peek-based flow control – BD=4

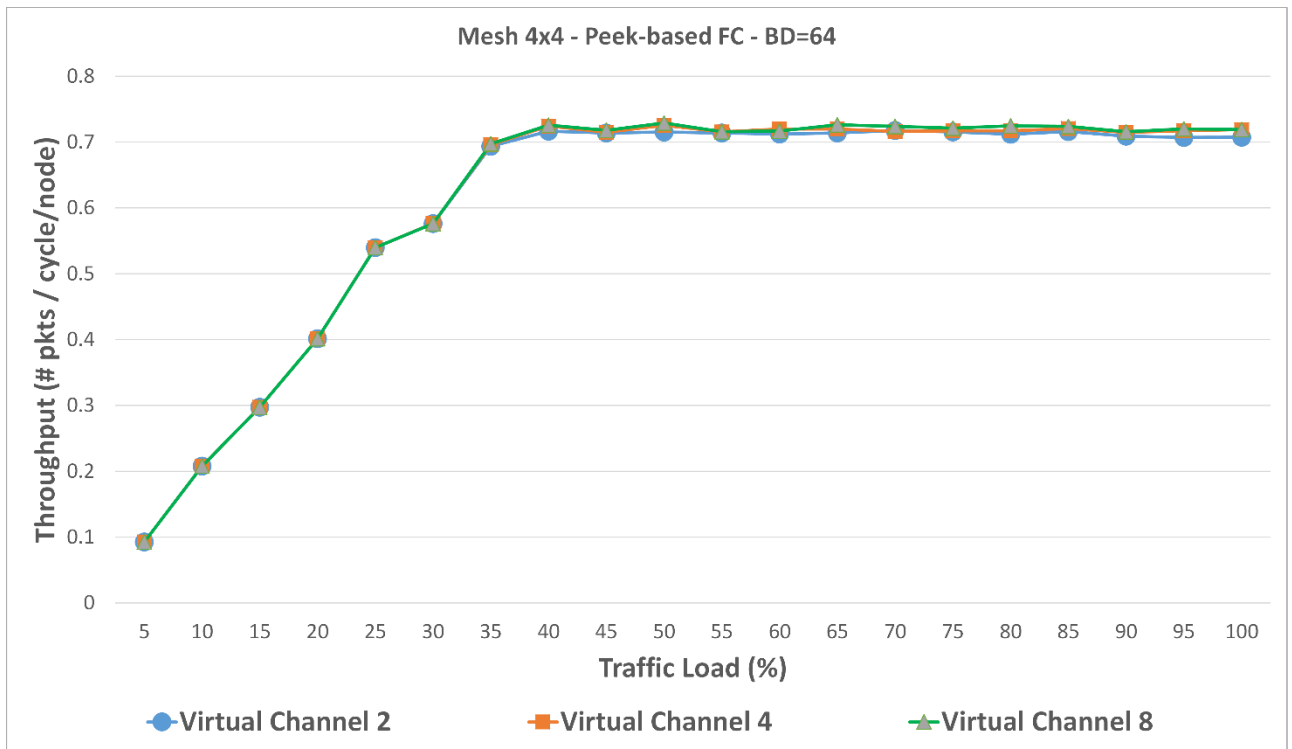


Figure 5.23: CONNECT throughput of all VC configurations – Mesh 4x4 – Peek-based flow control – BD=64

More results of virtual channel DPR are shown in Figures 5.24, 5.25, 5.26, 5.27, 5.28, and 5.29. The results are for a Credit-based flow control, buffer depth of 8, and a network topology configurations of (Mesh 4x4, 4x3, 3x3, 3x2, 2x2, and 2x1 respectively).

It is noticeable in all the results that the positive impact of virtual channel DPR is valuable only with the relatively large networks with small buffer depths. Investing in virtual channel DPR in the small networks or the large buffer depths results in a waste of area.

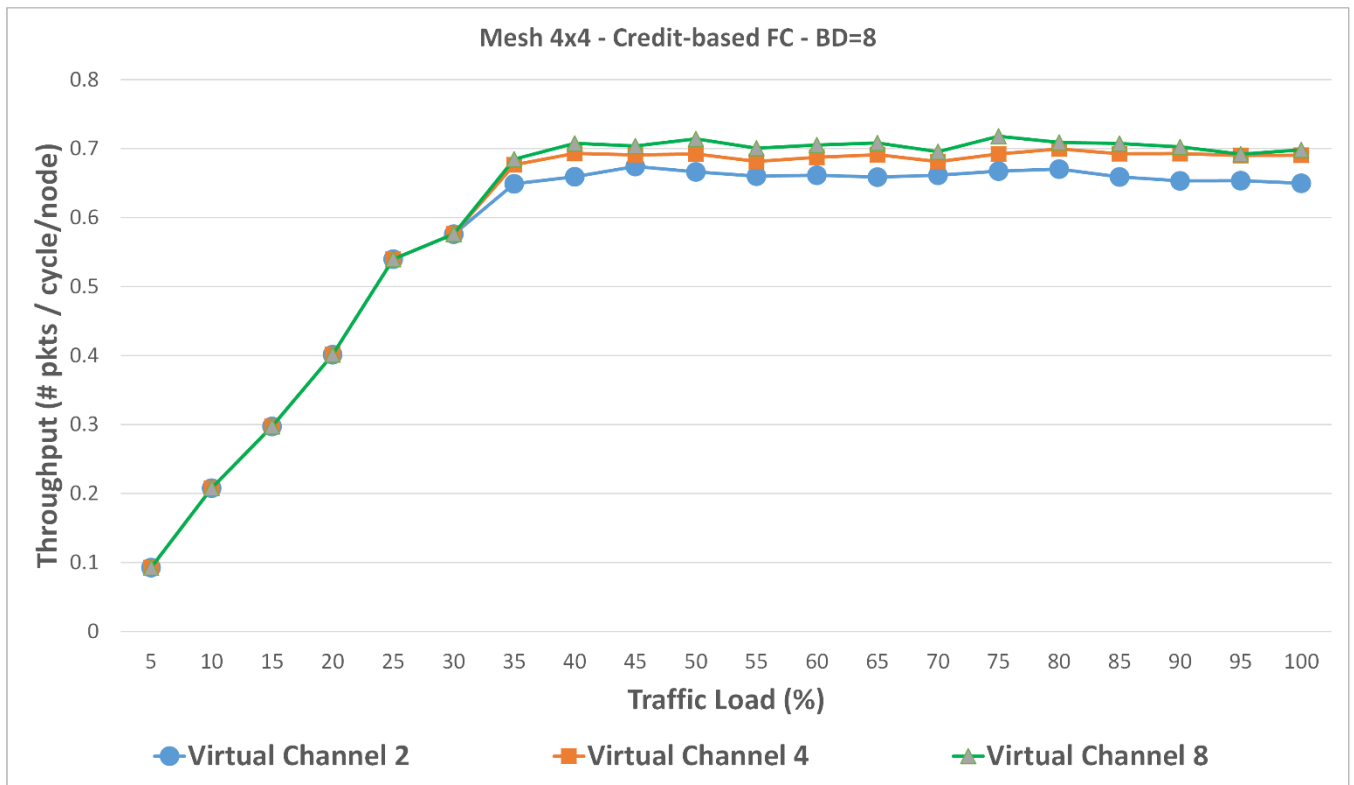


Figure 5.24: CONNECT throughput of all VC configurations – Mesh 4x4 – Credit-based flow control – BD=8

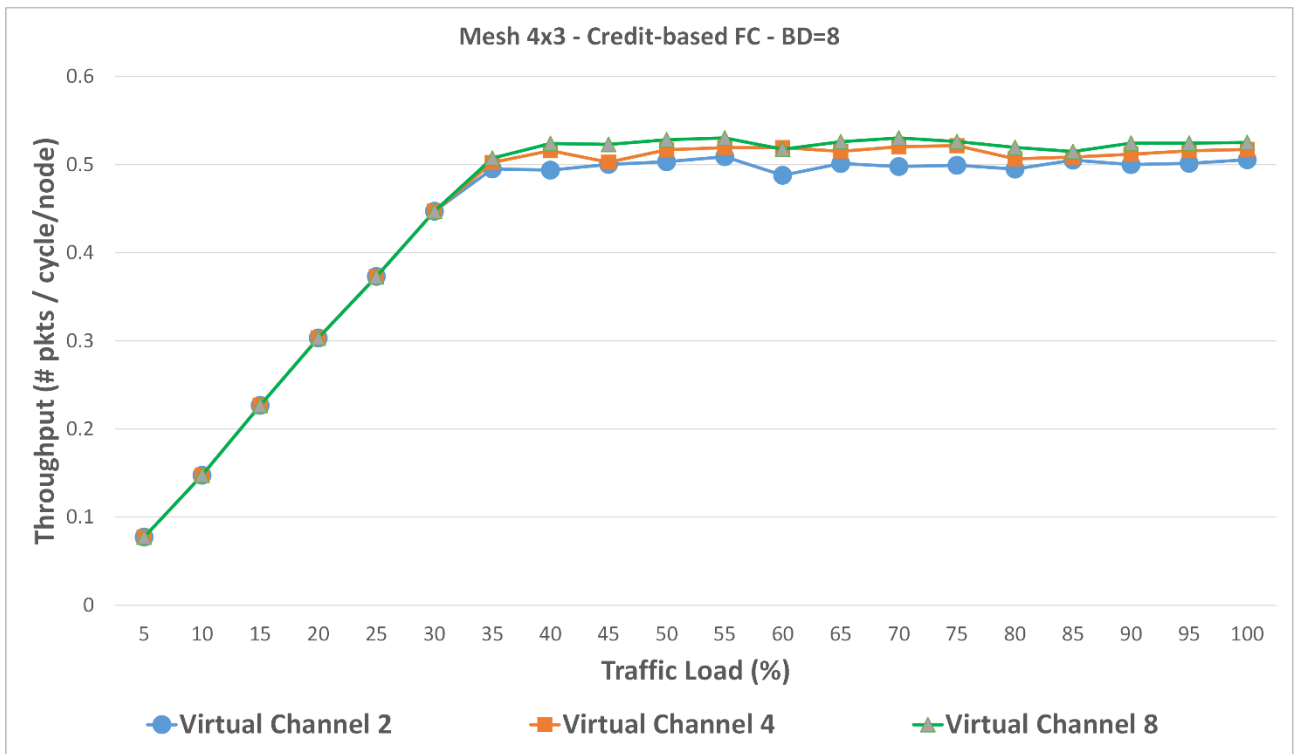


Figure 5.25: CONNECT throughput of all VC configurations – Mesh 4x3 – Credit-based flow control – BD=8

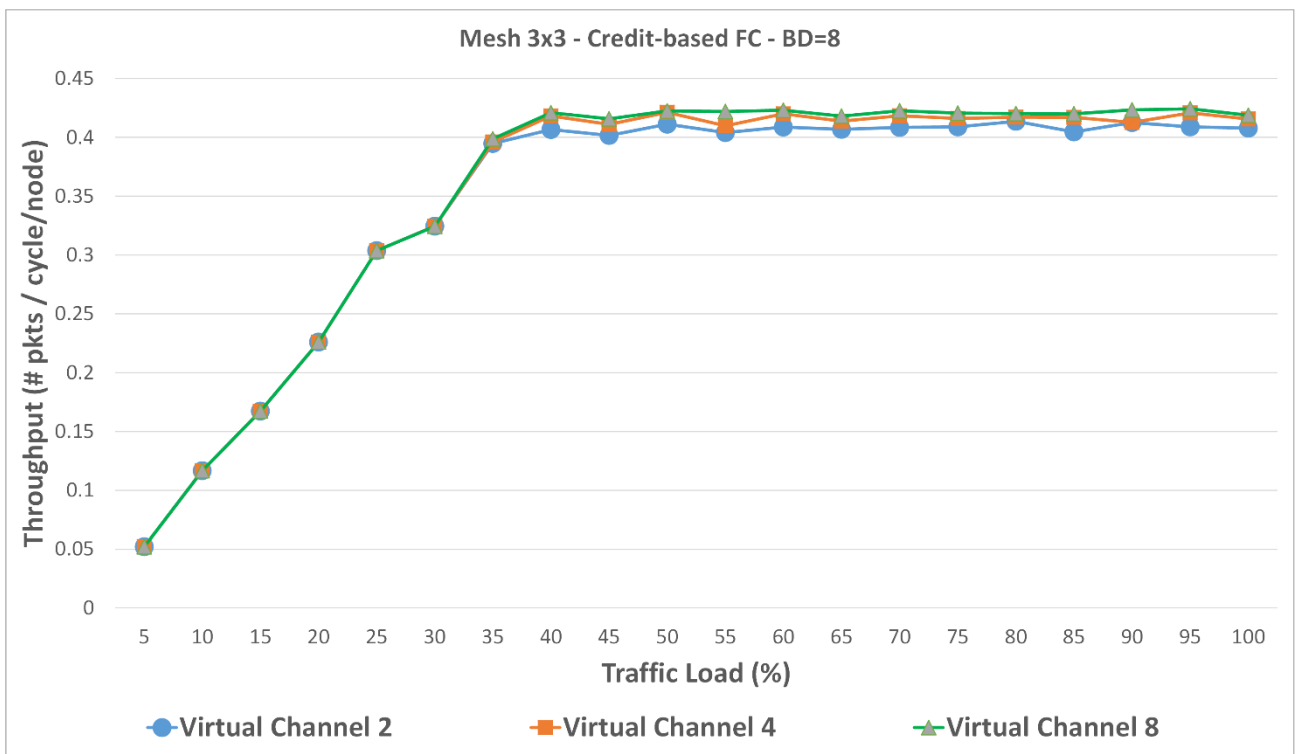


Figure 5.26: CONNECT throughput of all VC configurations – Mesh 3x3 – Credit-based flow control – BD=8

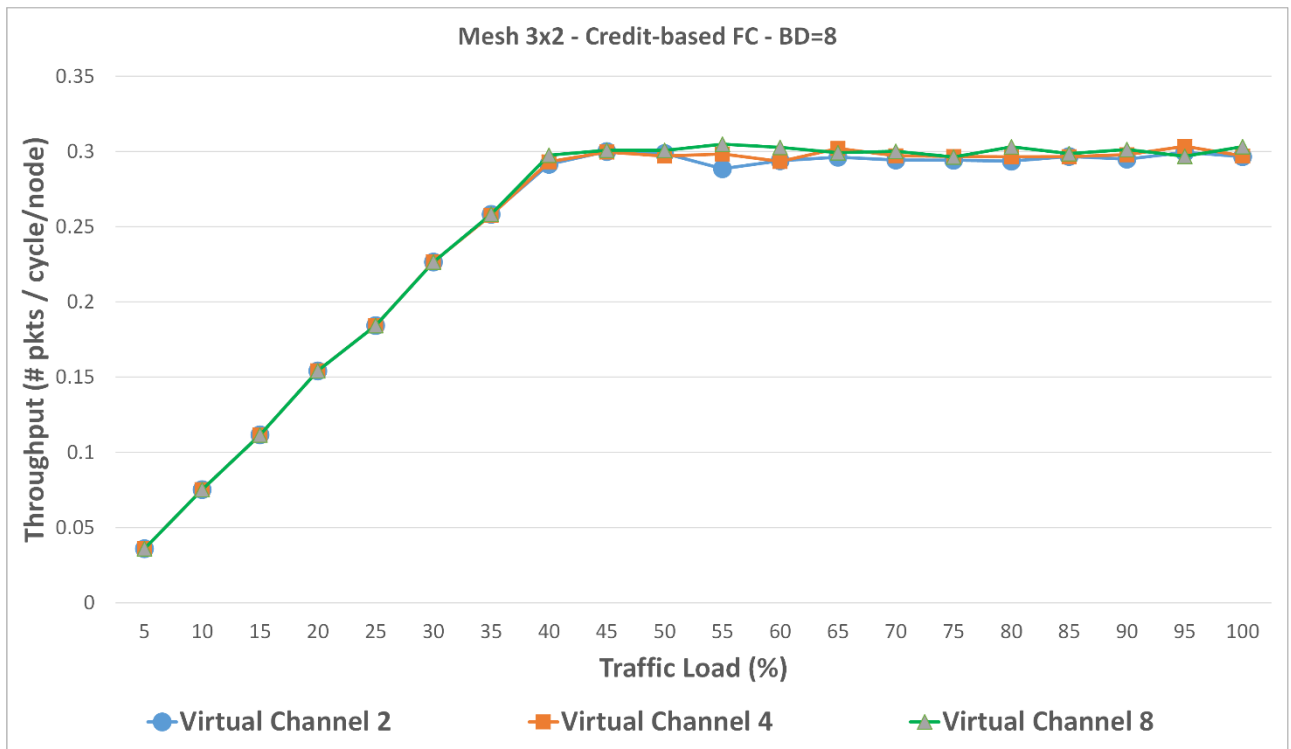


Figure 5.27: CONNECT throughput of all VC configurations – Mesh 3x2 – Credit-based flow control – BD=8

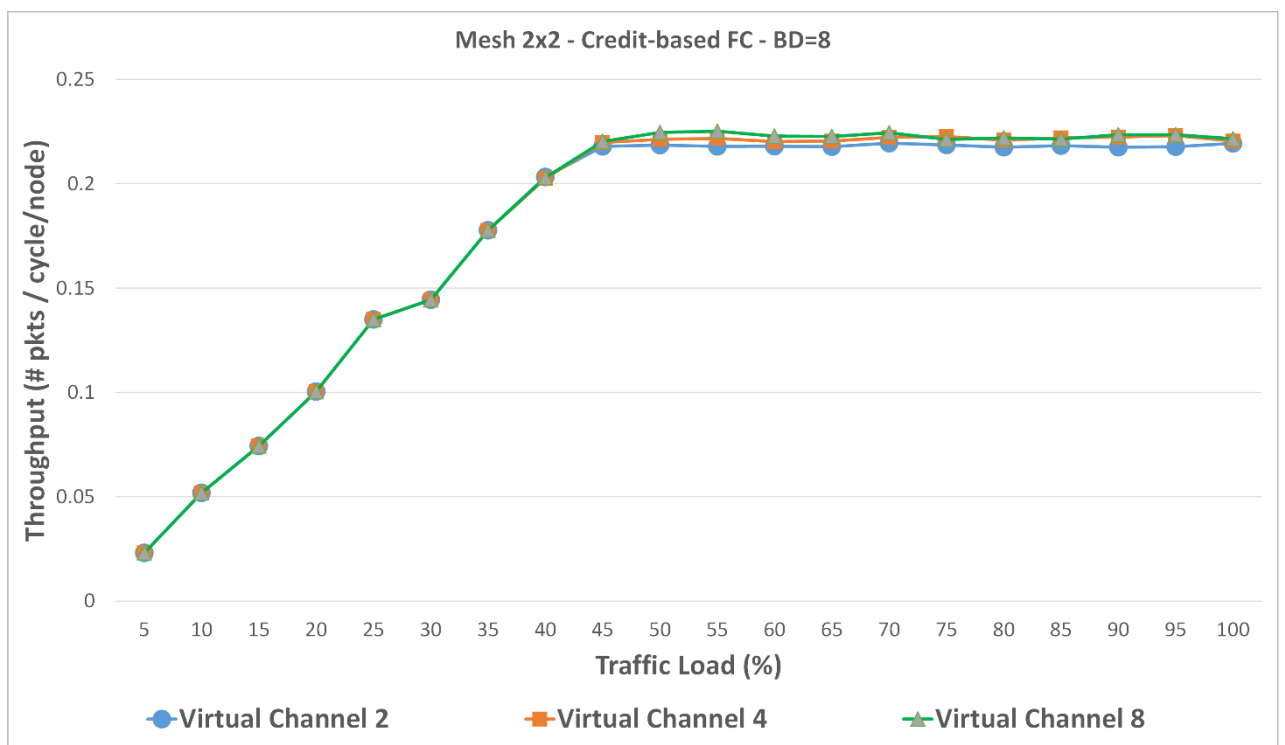


Figure 5.28: CONNECT throughput of all VC configurations – Mesh 2x2 – Credit-based flow control – BD=8

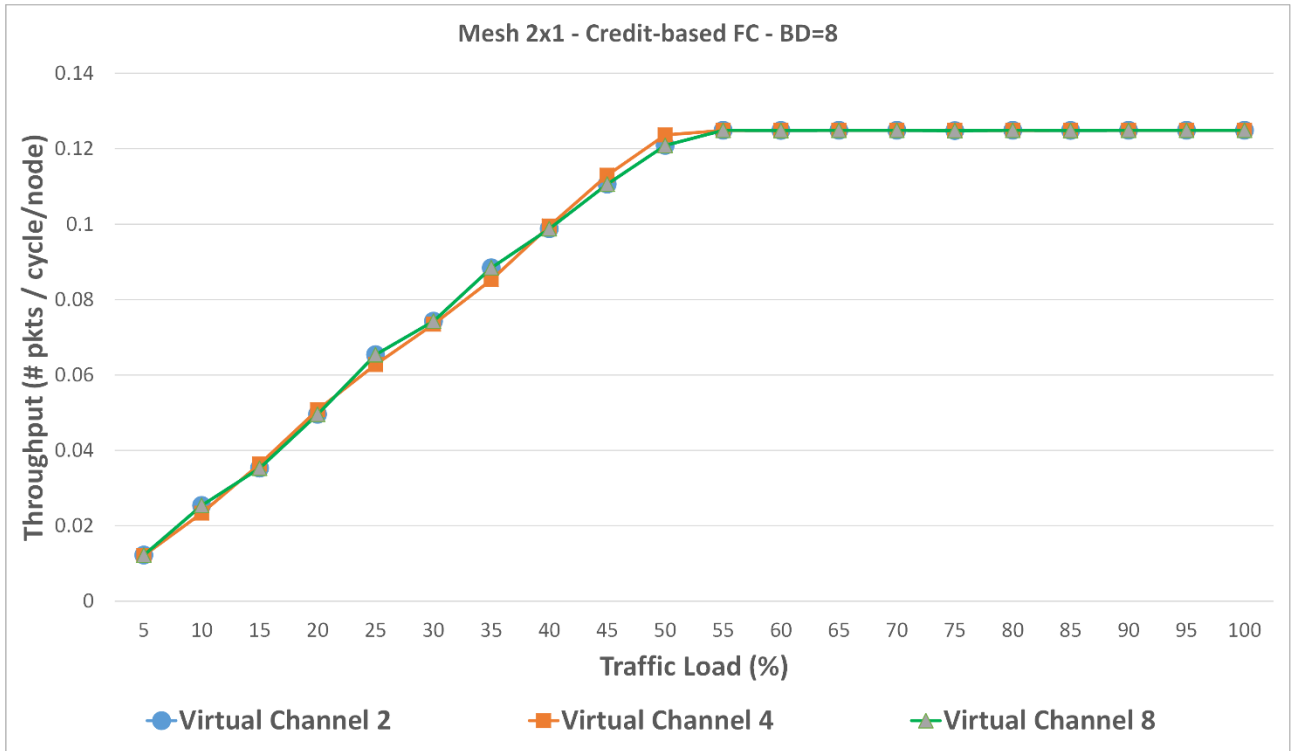


Figure 5.29: CONNECT throughput of all VC configurations – Mesh 2x1 – Credit-based flow control – BD=8

5.5. Buffer Depth vs Virtual Channel

Applying DPR into the buffer depth or the virtual channel impacts the network performance in nearly the same way. The performance shown here is for the buffer depths vs virtual channels inside the mesh 4x4 network with credit-based and peek-based flow controls, under a traffic load ranging from 5% to 100%.

Figures 5.30 and 5.31 show how using virtual channel/ buffer depth gives nearly the same effect with the same network configurations. This is proven using the 4-virtual channels and a buffer depth of eight compared with the eight virtual channels and a buffer depth of four.

The same result is highlighted in Figures 5.32 and 5.33 even with using a moderate-sized network (mesh 3x3). In addition, the flow control type does not change this conclusion as credit-based flow control in Figure 5.32 and peek-based flow control in Figure 5.33.

It is obvious that the rule of the small networks effect still applies. This is shown in Figure 5.34 and 5.35. In those Figures, a small-sized (mesh 2x2) network is used with credit-based and peek-based flow controls similar to the previous networks.

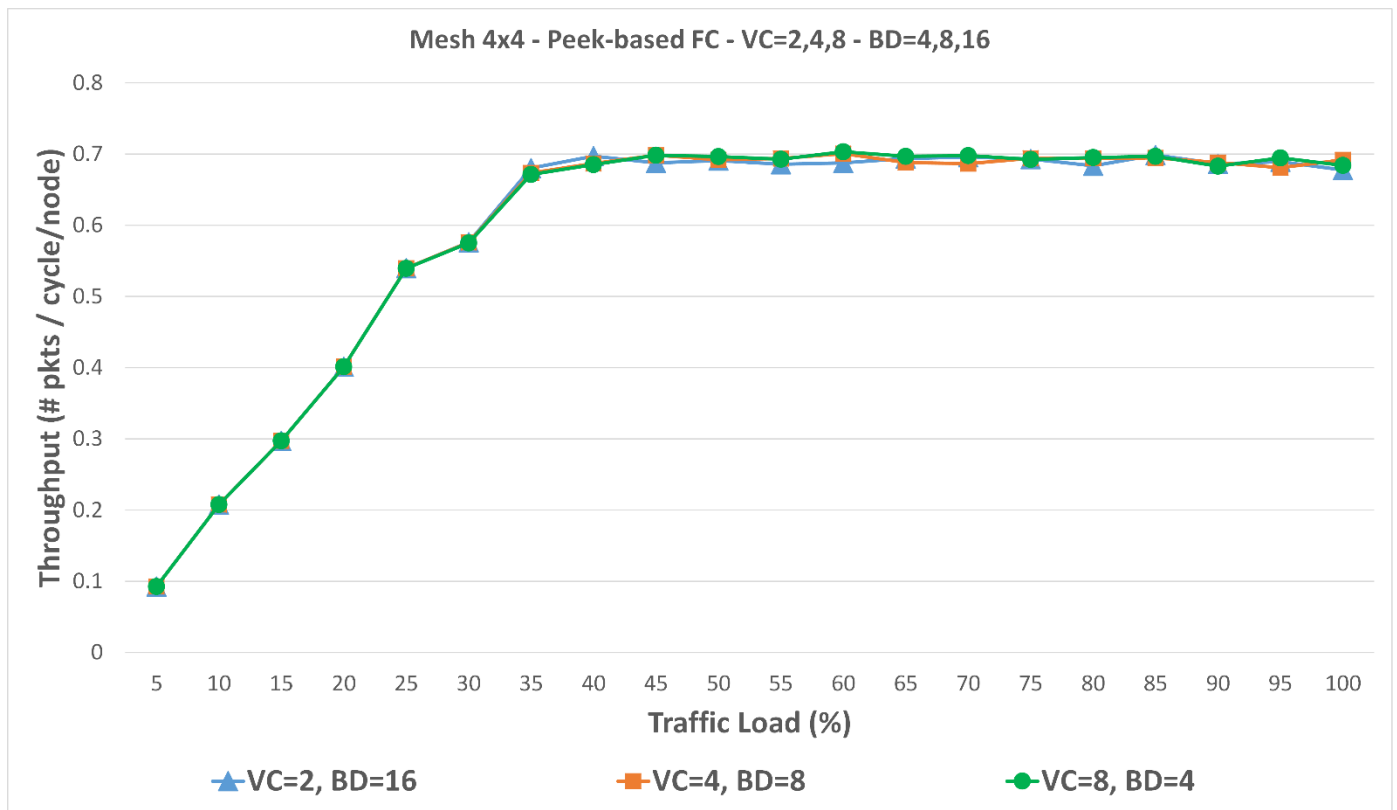


Figure 5.30: CONNECT throughput of BD vs VC configurations – Mesh 4x4 – Peek-based flow control – VC=2, 4, and 8 – BD=4, 8, and 16

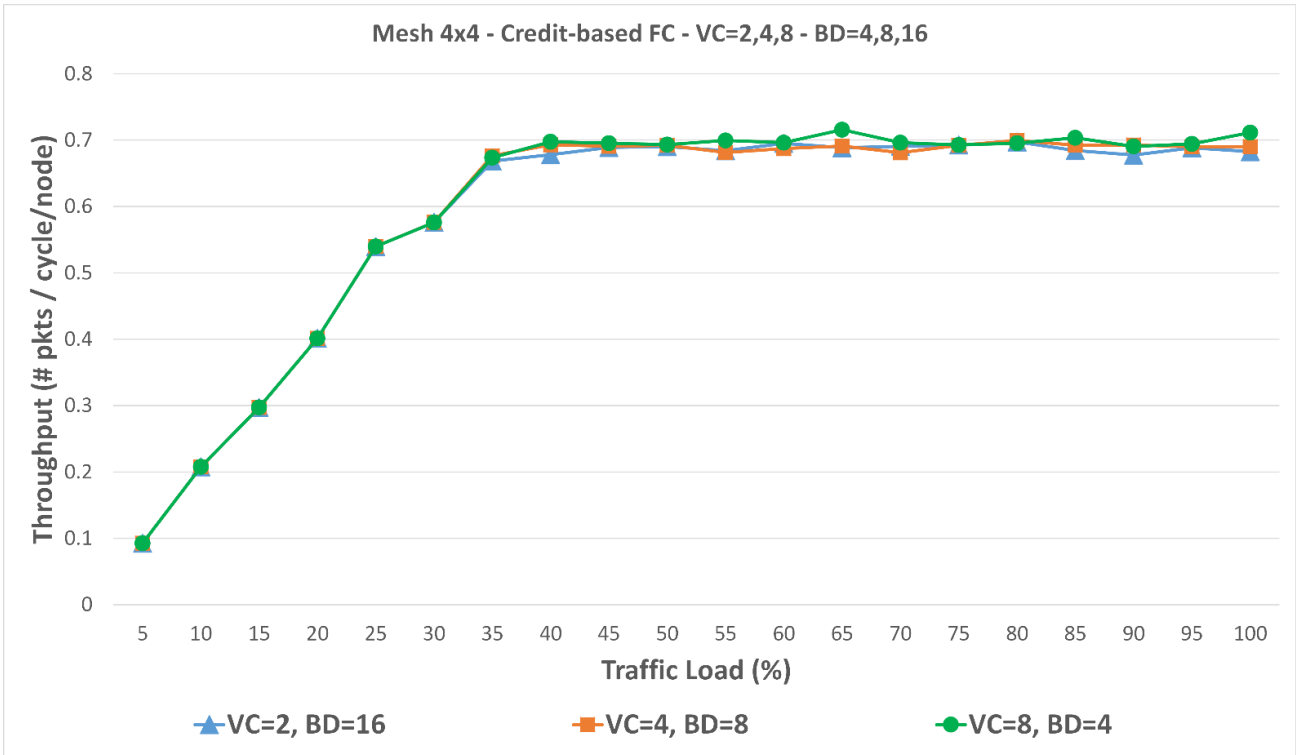


Figure 5.31: CONNECT throughput of BD vs VC configurations – Mesh 4x4 – Credit-based flow control – VC=2, 4, and 8 – BD=4, 8, and 16

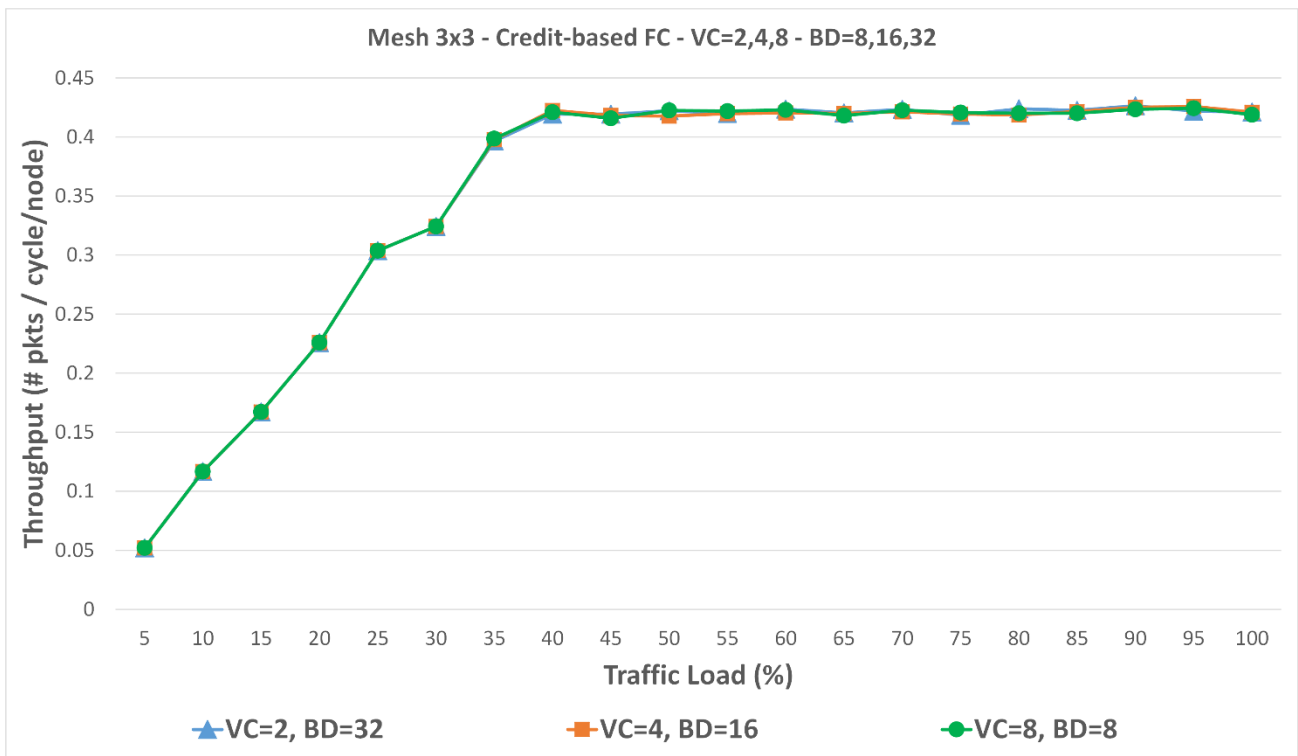


Figure 5.32: CONNECT throughput of BD vs VC configurations – Mesh 3x3 – Credit-based flow control – VC=2, 4, and 8 – BD=8, 16, and 32

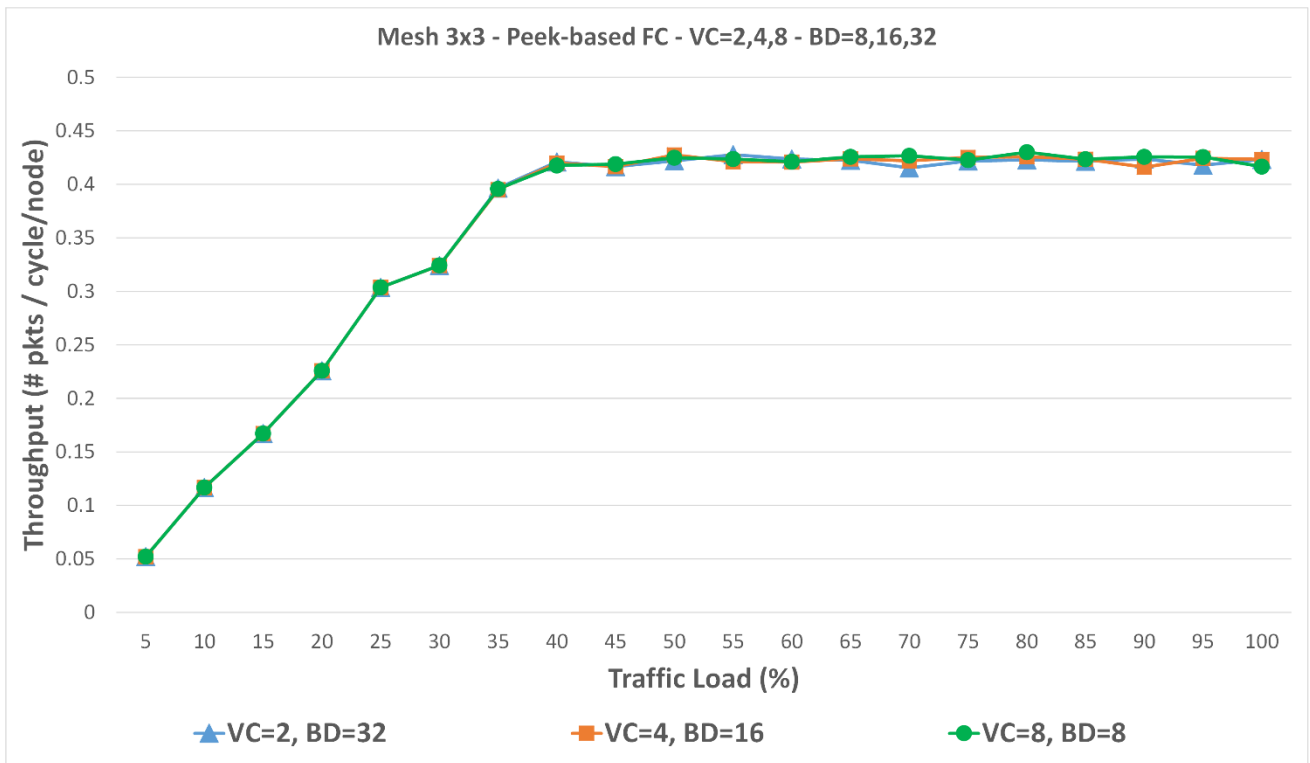


Figure 5.33: CONNECT throughput of BD vs VC configurations – Mesh 3x3 – Peek-based flow control – VC=2, 4, and 8 – BD=8, 16, and 32

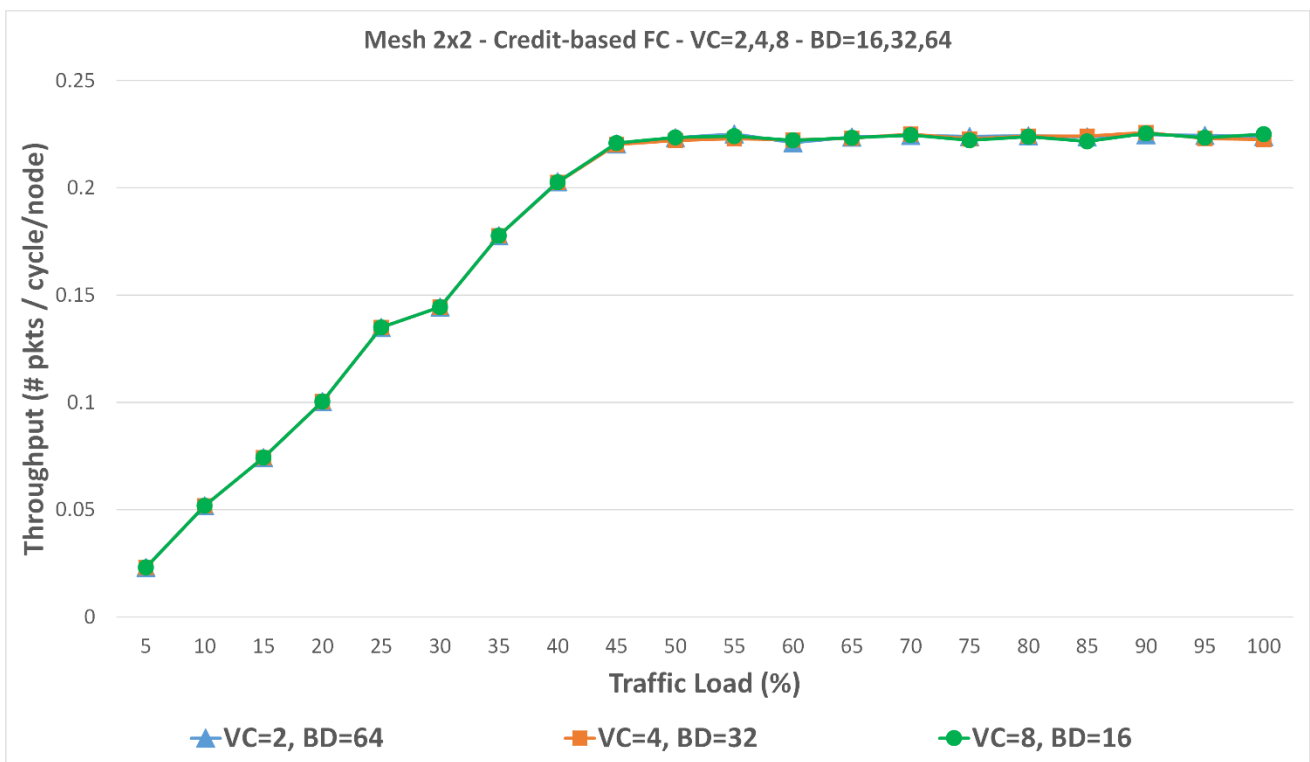


Figure 5.34: CONNECT throughput of BD vs VC configurations – Mesh 2x2 – Credit-based flow control – VC=2, 4, and 8 – BD=16, 32, and 64

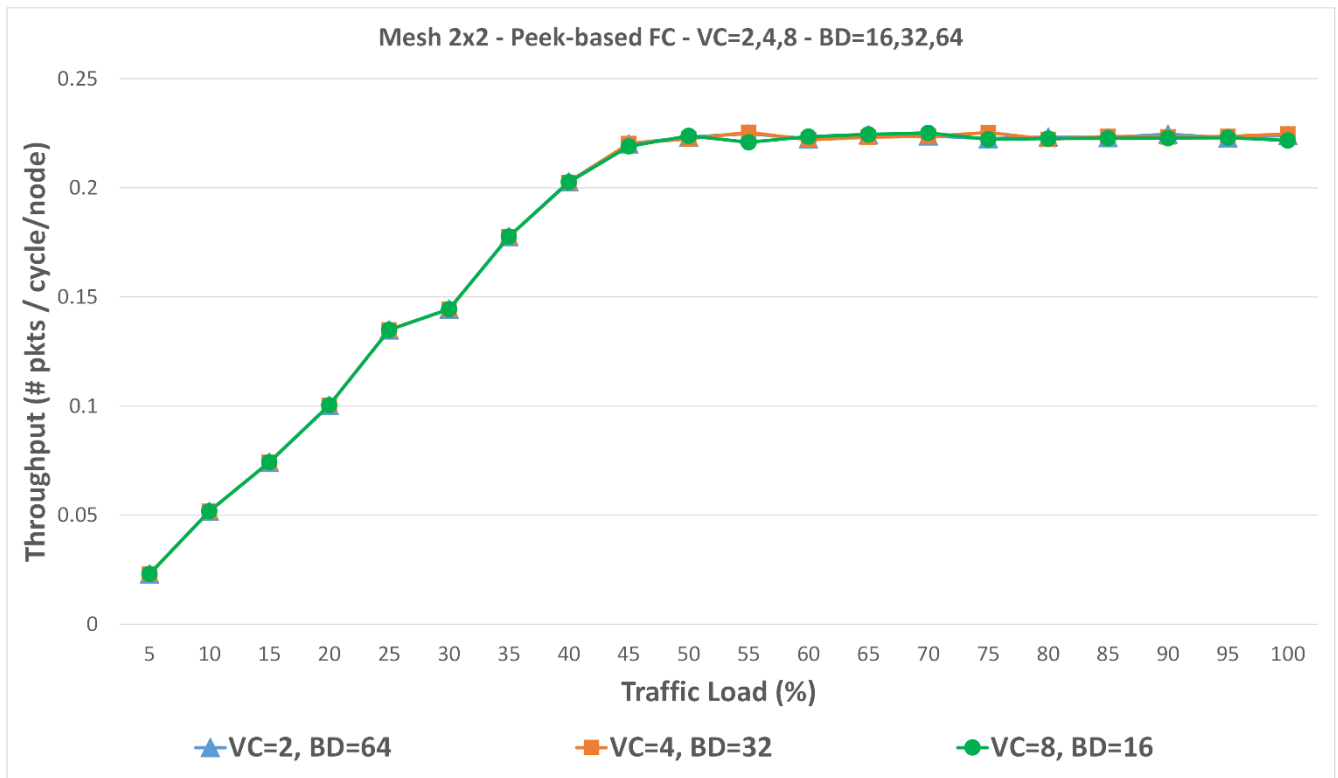


Figure 5.35: CONNECT throughput of BD vs VC configurations – Mesh 2x2 – Peek-based flow control – VC=2, 4, and 8 – BD=16, 32, and 64

5.6. Flow Control DPR Evaluation

The network flow control mechanism defines the feedback technique while communicating with neighbor routers. The credit-based flow control provides a detailed feedback when there is a free space in each virtual channel. However, the peek-based flow control just provides a busy signal indicating the availability of each virtual channel. Accordingly, The peek-based flow control is much simpler and allows maximizing the use of network resources. Nevertheless, the credit-based flow control provides more intelligence to the network in case of choosing different routing paths.

The following Figures show that the impact of Flow control mechanism is very slight. It is only noticeable with the small buffer depths that the peek-based flow control is preferred over the credit-based flow control. Note that the rule of the small networks still applies.

Figure 5.36 corresponds to a 4x4 mesh network with 2 virtual channels and a buffer depth of four. Figure 5.37 shows a 2x1 mesh network with 2 virtual channels and a buffer depth of four. Figure 5.38 corresponds to a 3x3 mesh network with four virtual channels and a buffer depth of 16. Figure 5.39 corresponds to a 2x2 mesh network with four virtual channels and a buffer depth of 32. Figure 5.40 corresponds to a 4x3 mesh network with eight virtual channels and a buffer depth of eight. Figure 5.41 corresponds to a 2x1 mesh network with eight virtual channels and a buffer depth of 64.

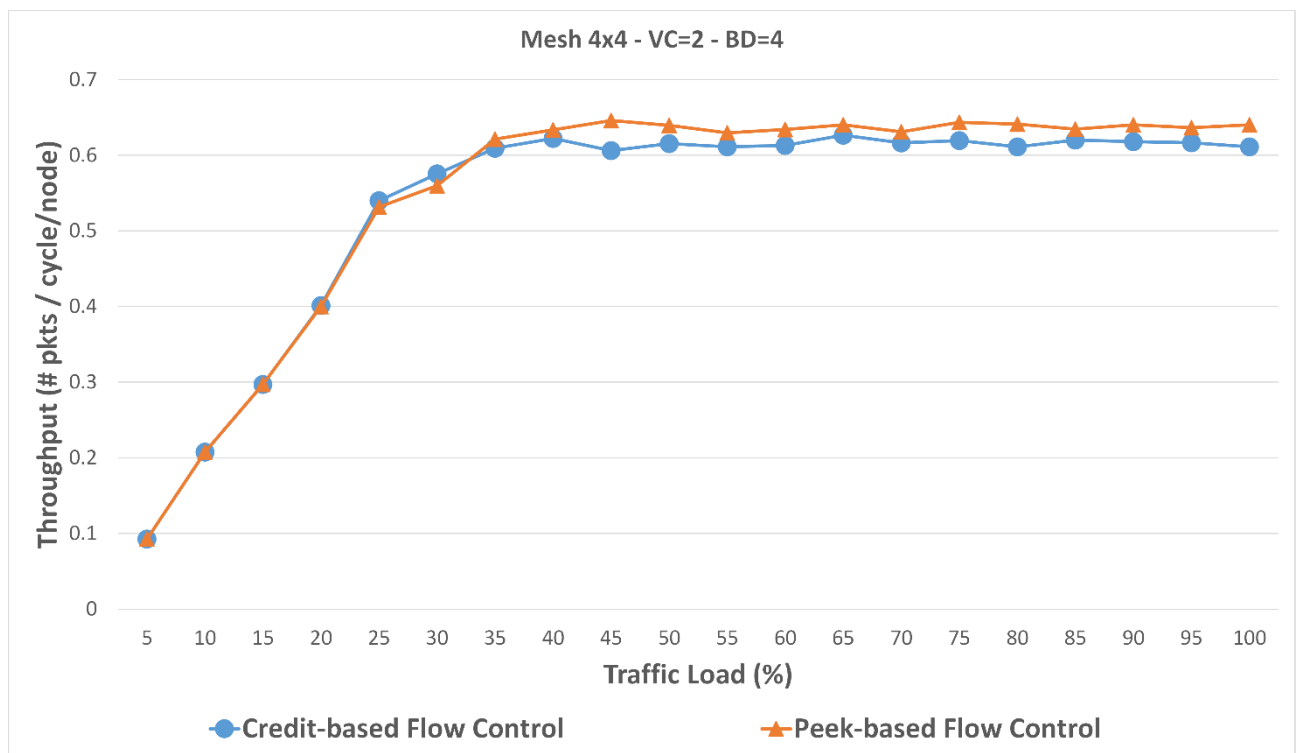


Figure 5.36: CONNECT throughput of Flow Control configurations – Mesh 4x4 – VC=2 – BD=4

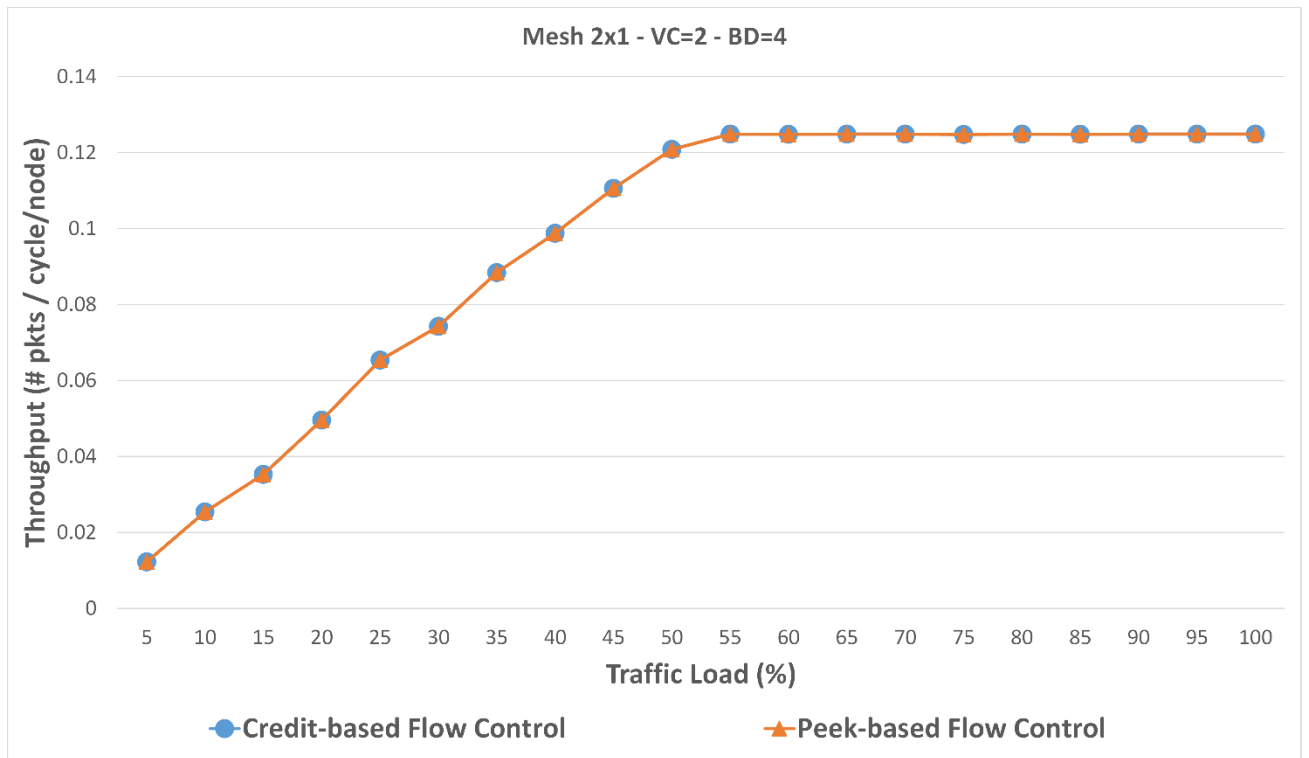


Figure 5.37: CONNECT throughput of Flow Control configurations – Mesh 2x1 – VC=2 – BD=4

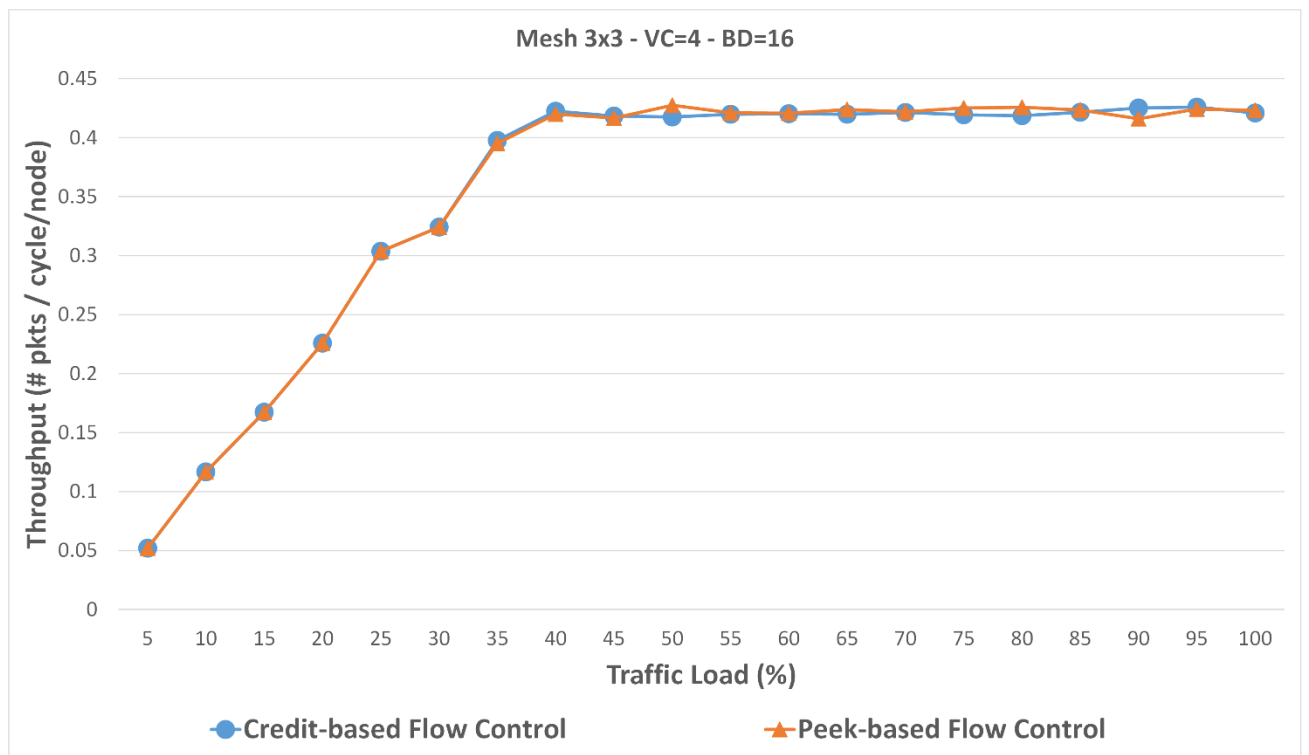


Figure 5.38: CONNECT throughput of Flow Control configurations – Mesh 3x3 – VC=4 – BD=16

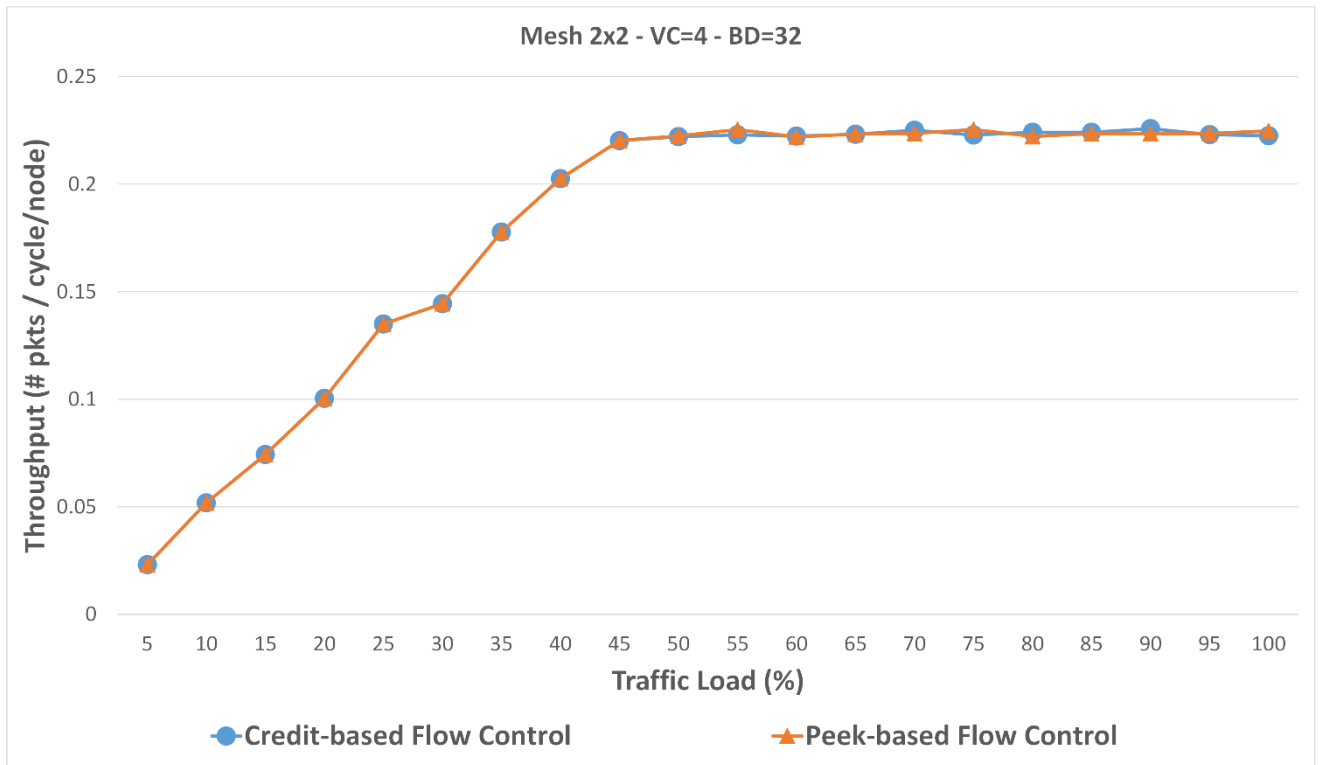


Figure 5.39: CONNECT throughput of Flow Control configurations – Mesh 2x2 – VC=4 – BD=32

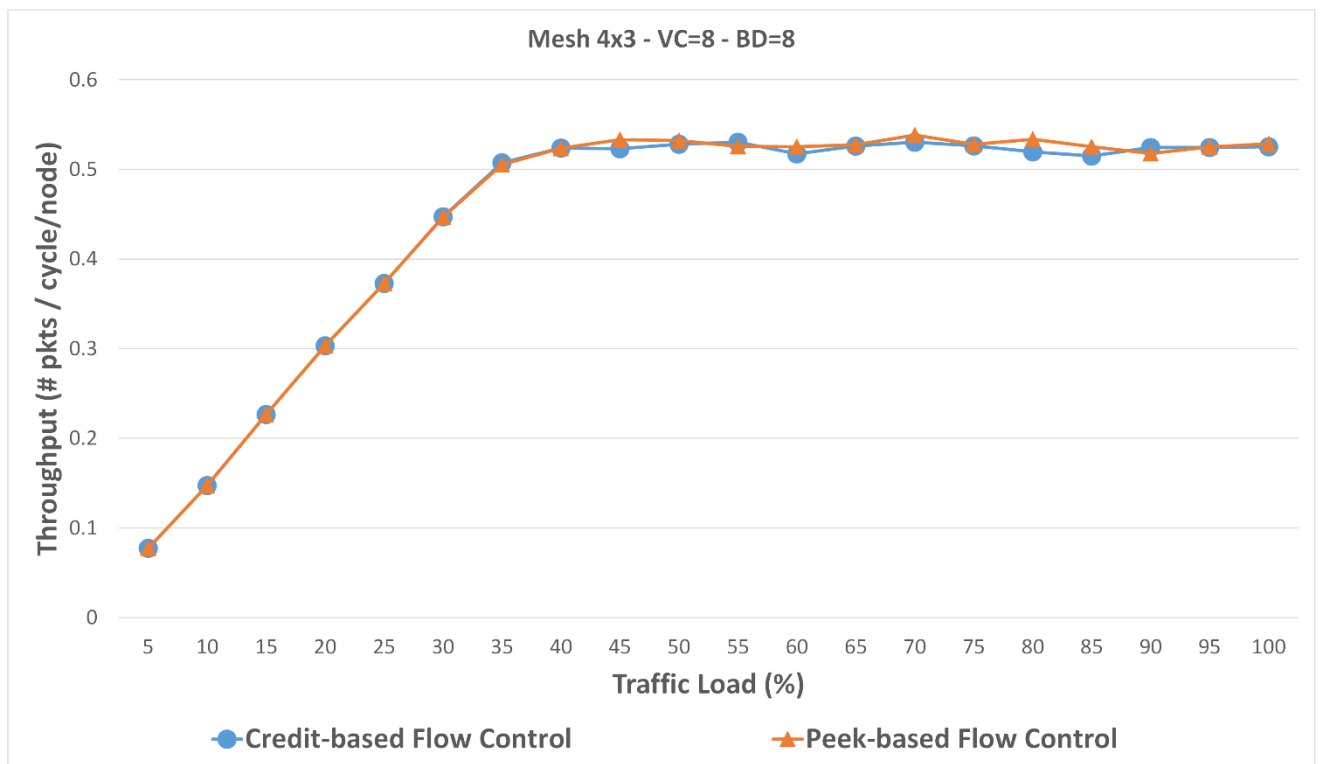


Figure 5.40: CONNECT throughput of Flow Control configurations – Mesh 4x3 – VC=8 – BD=8

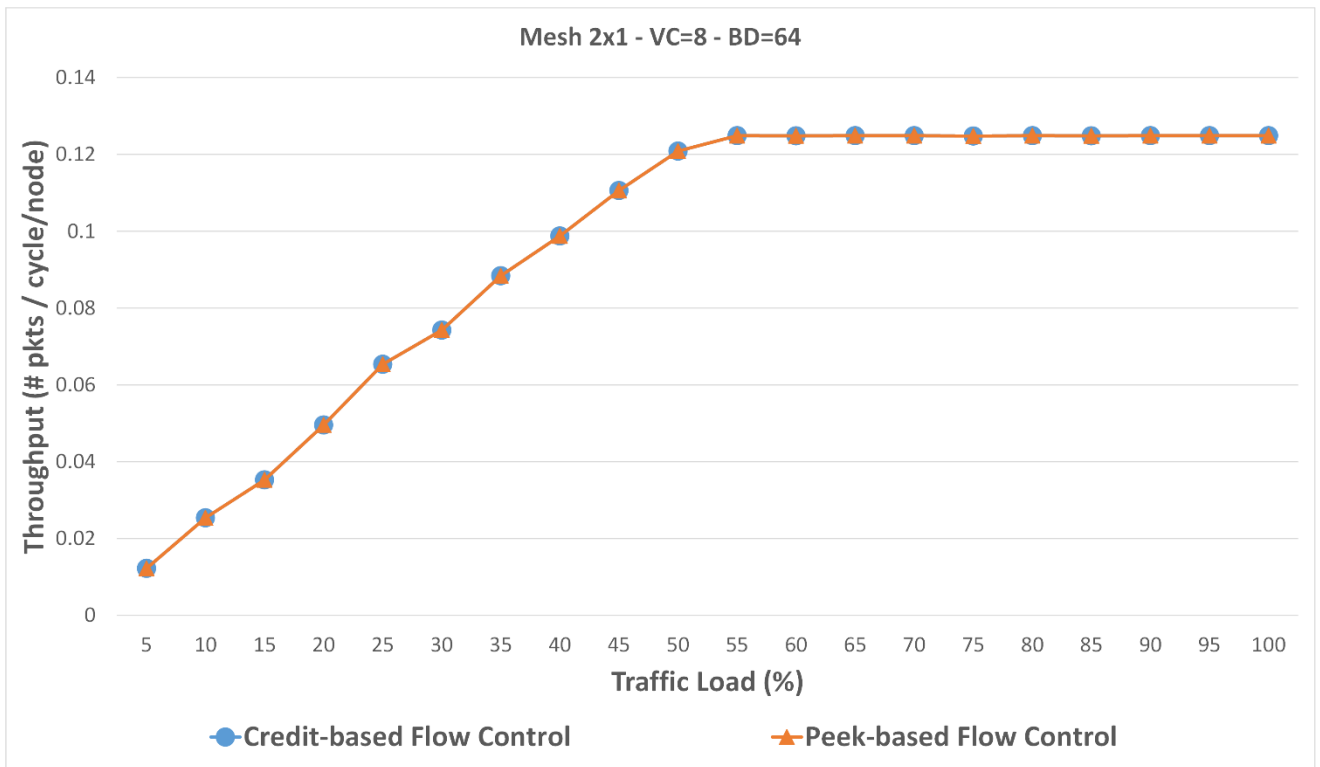


Figure 5.41: CONNECT throughput Flow Control configurations – Mesh 2x1 – VC=8 – BD=64

5.7. Area Evaluation

As a different performance metric, the experimental results of the area for the different networks is considered the most important factor in deciding the most suitable network with every benchmark. This metric is mainly used by the reconfiguration tool in order to prevent the throughput from misleading the network selection criteria.

In this subsection, an example highlights how the area is the valuable gain by using the reconfigurable NoCs over the static NoCs.

The original CONNECT and the reconfigurable CONNECT are implemented on Virtex-5 xc5v1x110tff1136-1 FPGA. At the implementation level, adding the runtime reconfigurability to the 4x4 mesh CONNECT increases the area. This overhead is due to the switches and routing adaptation.

For the area resources in Virtex 5 Xilinx FPGAs, Table 5.1 shows the estimation for the area for each Virtex 5 resource with respect to the equivalent number of gates and the absolute area in mm². These area results are used by the reconfiguration tool as a metric for the best fitting network. This criteria is used as the main aim of the reconfigurable NoC is to allow area saving when switching between the different networks.

Table 5.1: Estimated Virtex 5 Xilinx FPGA resource area [26, 27, 33]

Resource	Equivalent Number of gates	Silicon Area in mm ²
Register	7	0.000341
LUT	24	0.001171
IO	100	0.004882
BRAM	-	0.025436

5.7.1. Case Study

This is a case study highlighting the effect of the reconfigurable NoC. The five benchmarks listed in Table 5.2 are designed using the Static NoC approach and the Reconfigurable NoC approach.

The different reconfigurable benchmarks force designing the static NoC with the highest performance requirements and consuming the biggest area all the time. However, the reconfigurable NoC offers switching to the best fit configuration satisfying the benchmark requirement without reserving unused area.

Table 5.2: Case study benchmarks with Static NoC and Reconfigurable NoC approaches

Benchmark	Static NoC approach	Reconfigurable NoC approach
Benchmark 1	Mesh 4x4	Mesh 4x4
Benchmark 2	Mesh 4x4	Mesh 4x3
Benchmark 3	Mesh 4x4	Mesh 3x3
Benchmark 4	Mesh 4x4	Mesh 3x2
Benchmark 5	Mesh 4x4	Mesh 2x2

Figure 5.42 represents a usage model example of two NoCs switching between the set of listed benchmarks. These benchmarks are ranging from a 4x4 mesh network down to a 2x2 mesh network. The first NoC is a static NoC, which shows a constant area resource usage even with moving from an application to another. However, the second NoC which is a reconfigurable NoC shows a variable area with each new reconfiguration. The switching takes place and hence the current area utilization are determined according to the current application used. In general, the NoC reconfigurability power appears when the usage model does not require the high throughput all the time.

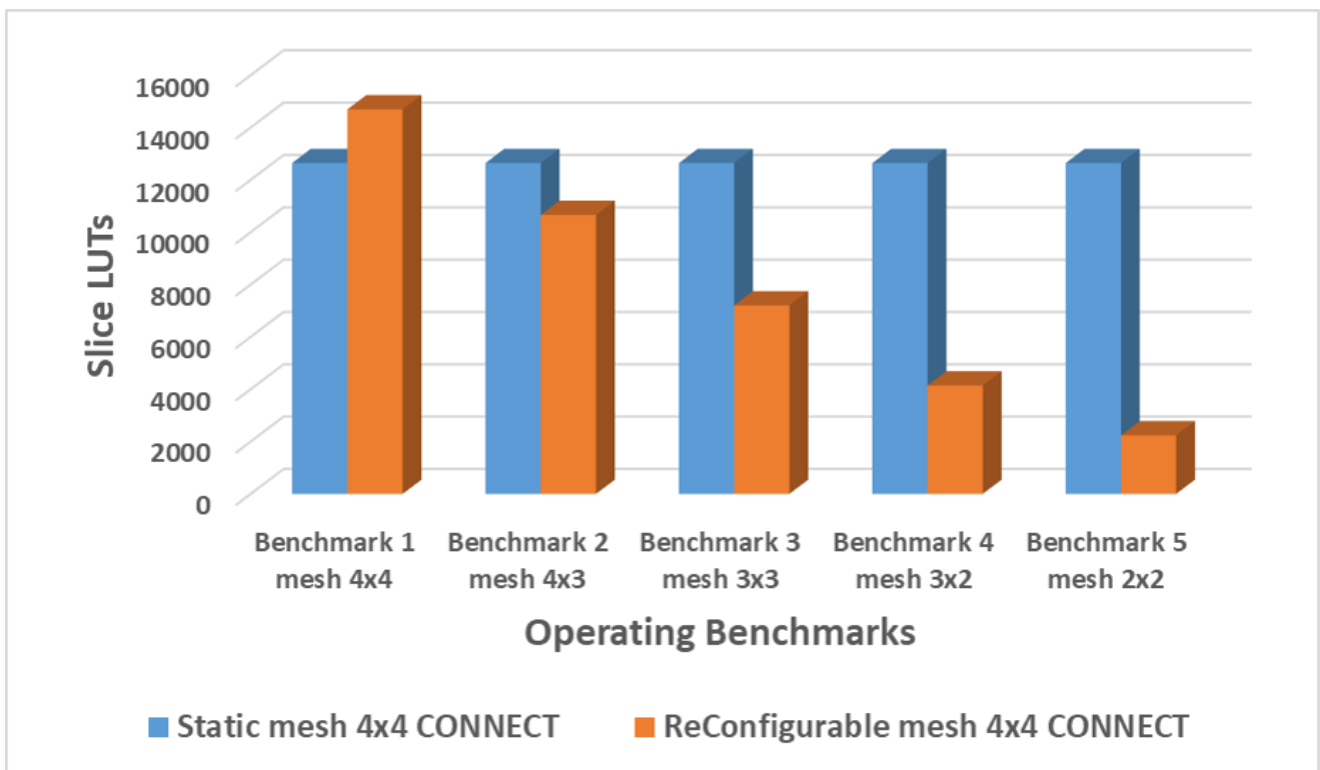


Figure 5.42: Virtex 5 xc5vlx110tff1136-1 Area results of reconfigurable mesh 4x4 CONNECT vs Static mesh 4x4 CONNECT

The area resources of all those networks are listed in Table 5.3. The area reduction is noticed with decreasing the network size when switching between the different benchmarks.

Table 5.3: Area resources for different networks corresponding to different benchmarks

Network	VC	BD	Slice Regs	Slice LUTs	LUT-FF pairs	IOs	BUFG
Mesh 4x4 (Static)	2	4	3758	12644	2287	914	2
Mesh 4x4 (Reconfig.)	2	4	3758	14696	2276	930	2
Mesh 4x3 (Reconfig.)	2	4	2728	10660	1668	818	1
Mesh 3x3 (Reconfig.)	2	4	1976	7202	1231	733	1
Mesh 3x2 (Reconfig.)	2	4	1207	4145	766	648	1
Mesh 2x2 (Reconfig.)	2	4	726	2238	459	590	1

Assuming that the five benchmarks are operating with equal times, the overall area saving in this case study is as follows:

- Saving in Slice Registers: **44.67%**
- Saving in Slice LUTs: **38.4%**
- Saving in LUT-FF pairs: **44%**
- Saving in IOs: **18.6%**
- Saving in BUFG/BUFGCTRL: **40%**

5.8. Design recommendations

From the previous evaluations, some recommendations should be considered when planning to use NoCs in the user design and whether static or reconfigurable NoCs are going to be used. These design recommendations are listed below:

- Reconfigurable NoCs are preferred when there are multiple benchmarks are going to run with different requirements (traffic load, throughput). The reconfigurable NoCs will give the design the required adaptability to the runtime requirements with area and power gains.
- Static NoCs are preferred when only a single benchmark is running. Additionally, it is suitable for multiple benchmarks with the same performance requirements. The adaptability here has no meaning as the runtime requirements does not need a lot of variations.
- The main gain behind applying PDR to the NoC topology is the area and power saving. Removing a set of routers and changing the network topology during the runtime degrades the performance while saving the area of the reconfigured nodes. This area saved is going to be used by the rest of the design.
- Applying DPR to the network buffer depth is beneficial with the high traffic loads and the relatively large networks. It has its minimal effect with the small networks or the low traffic loads.
- Applying DPR to the network virtual channel has its most effect with high traffic loads and relatively large networks plus a small buffer depth. Large buffer depth networks could prevent making the most of the available virtual channels.
- Choosing buffer depth DPR or virtual channel DPR depends mainly on the usage model of the design specified benchmarks. The virtual channel is preferred when planning to use the parallel loading and packet injection. However, the buffer depth is preferred when the internal router storage is more important than routing resources.
- Flow control mechanism DPR could help when fine tuning the network parameters while DPR selection. The peek flow control is much simpler in the implementation which means less area and power. However, the credit flow control gives the router a more detailed information about the traffic going through the neighbors and gives a more smart insight with the possible routing paths.

Conclusion

In this thesis, four reconfiguration methods for DPR in Xilinx FPGAs are reviewed and results are discussed.

It is obvious that JTAG and Serial Mode reconfiguration methods are much slower than the ICAP and SelectMap methods. However, serial reconfiguration methods have less significant area overhead compared to the parallel methods. Therefore, the performance with JTAG and Serial methods is better than the parallel methods with small design areas. With these less significant designs, the area overhead is very noticeable. Despite that, the performance with JTAG and Serial modes are not recommended with the more significant design areas. The area overhead is not significant compared to the large design areas. On the other hand, JTAG allows sending internal signals to the outside for debugging purposes.

In addition, the methods that use a parallel port support a high speed reconfiguration compared to the others, especially with large designs.

This work presents a study on the reconfiguration impact on the NoC performance. Additionally, it focuses on how shrinking the network size during the runtime results in area saving. This saving is at the expense of degrading the performance. In general, the low performance is suitable with certain benchmarks under certain traffic loads. Moreover, other network configuration parameters are studied and their impact on the network performance is analyzed.

Since the large network requires hosting the packets for a longer time than the smaller network, the buffer depth contributes in enhancing the performance with the larger network sizes. Moreover, the Virtual channel acts as a booster for the performance especially with the large networks and small buffer depths. In addition, the flow control mechanism impact is noticeable with the small buffer depths. Finally, the area metric plays a very important role in the best network selection. The area results are considered the main advantage that is gained from applying DPR into NoC.

Future Work

The reconfigurable mesh 4x4 CONNECT holds a detailed analysis on the throughput as a performance metric of the network performance. However, this study can be extended in the future to include the following:

- Evaluate larger networks such as 6x6 and 8x8 mesh networks in order to provide more configuration options to the user and more detailed analysis on the large scale NoCs.
- Evaluate other network topologies such as Ring and Star networks and providing a detailed analysis and a comparison among them.
- Consider the impact of the network latency as a performance metric. The network latency shall reflect on the best network structure selection by the reconfiguration tool.
- Engage this work with the new NoC simulators and study the opposite way how the NoC should impact the reconfiguration time and the DPR performance in general. This should give the user a complete holistic view on the reconfiguration and its usage with the NoC.
- Propose a technique for estimating traffic load for every user benchmark.

References

1. S. M. Trimberger, "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology," in *Proceedings of the IEEE*, vol. 103, no. 3, pp. 318-331, March 2015.
2. Farooq U, Marrakchi Z, and Mehrez H., "FPGA architectures: An overview. In: *Tree-based Heterogeneous FPGA Architectures*," Springer, New York, pp 7-48, 2012.
3. Xilinx Inc., "Virtex-4 FPGA Configuration Guide, UG071", June 2017.
4. http://www.fpl2012.org/Presentations/Keynote_Steve_Teig.pdf. Accessed: 2018-06-03
5. M. Liuzy , Z. Luy , W. Kuehnz, and A Jantsch, "Reducing FPGA Reconfiguration Time Overhead using Virtual Configurations," in *ReCoSoC*, 2010.
6. P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgeford, "Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration on XILINX FPGAS," in *Proceedings of the 16th International Conference on Field Programmable Logic and Applications, FPL06, Madrid, Spain, August 2006*.
7. Xilinx Inc., "Partial Reconfiguration User Guide, UG702", 2013.
8. <http://users.ece.cmu.edu/~mpapamic/connect/>. Accessed: 2018-06-03
9. M. Katevenis, "Buffer requirements of credit-based flow control when a minimum draining rate is guaranteed," *The Fourth IEEE Workshop on High-Performance Communication Systems, Greece*, pp. 168-178, 1997.
10. A. Ahmadinia, C. Bobda, J. Ding, M. Majer, J. Teich, S. P. Fekete, and J. C. van der Veen, "A Practical Approach for Circuit Routing on Dynamic Reconfigurable Devices," in *Proceedings of RSP*, pp. 84-90, 2005.
11. C. Bobda and A. Ahmadinia, "Dynamic interconnection of reconfigurable modules on reconfigurable devices," *Design & Test of Computers*, vol. 22, no. 5, pp. 443-451, 2005.
12. C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. Fekete, and J. van der Veen, "DyNoC: A dynamic infrastructure for communication in dynamically reconfigurable devices," in *Proc. Int. Conf. Field Program. Logic Appl.*, pp. 153-158, Aug. 2005.
13. T. Pionteck, R. Koch, and C. Albrecht, "Applying partial reconfiguration to networks-on-chips," in *Proc. Int. Conf. Field Programmable Logic and Applications, FPL*, pages 1-6. IEEE, 2006.
14. T. Pionteck, C. Albrecht, and R. Koch, "A Dynamically Reconfigurable PacketSwitched Network-on-Chip," in *Proceeding of the conference on Design, Automation and Test in Europe, DATE'06*, vol.1, pp.8-9, March 2006.
15. M. Modarressi, H. Sarbazi-Azad, and A. Tavakkol, "An efficient dynamically reconfigurable on-chip network architecture," *Proc. of the 47th Design Automation Conference, DAC 2010*, pp. 310-313, 2010.

16. E. J. McDonald, "Runtime FPGA Partial Reconfiguration," in Proc. of 2008 IEEE Aerospace Conference, pp. 1-7, Mar. 2008.
17. Jean-Philippe Delahaye, Pierre Leray, Christophe Moy, and Jacques Palicot, "Managing Dynamic Partial Reconfiguration on Heterogeneous SDR Platforms," SDR Forum Technical Conference'05, Anaheim, USA, November 2005.
18. H. Tan, R. F. DeMara, A. J. Thakkar, A. Ejnoui, and J. Sattler, "Complexity and Performance Evaluation of Two Partial Reconfiguration Interfaces on FPGAs: A Case Study," in Proceedings of ERSA'06, Las Vegas, Nevada, USA, pp. 253-256, June 2006.
19. Liu, M., Kuehn, W., Lu, Z., and Jantsch, A., "Run-time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration," in Proceedings of FPL, Prague, Czech Republic, 2009.
20. K. Vipin and S. Fahmy, "A high speed open source controller for FPGA partial reconfiguration," Proc. Int. Conf. Field Programmable Technol., FPT, pp.61 -66, 2012.
21. K. Papadimitriou, A. Dollas, and S. Hauck, "Performance of partial reconfiguration in FPGA systems: A survey and cost model," ACM Transactions on Reconfigurable Technology and Systems, TRETs, vol. 4, no. 4, pp. 36:1-36:24, Dec. 2011.
22. C. Claus , F. Muller , J. Zeppenfeld and W. Stechele, "A new framework to accelerate Virtex-II Pro dynamic partial self-reconfiguration," Proc. IEEE Int. Parallel Distrib. Process. Symp., pp.1 -7, 2007.
23. Xilinx Inc., "Virtex-5 FPGA Configuration User Guide, UG191", 2012.
24. Xilinx Inc., "Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode", 2009.
25. Helal, K. A., S. Attia, T. Ismail, and H. Mostafa, "Comparative Review of NoCs in the Context of ASICs and FPGAs", ISCAS, pp. 1866- 1869, 2015.
26. <https://www.xilinx.com/training/downloads/what-is-the-difference-between-an-fpga-and-an-asic.pptx>. Accessed: 2018-06-03
27. F. Arnaud., "A Functional 0.69 Embedded 6T-SRAM bit cell for 65nm CMOS platform," the Digest of Technical Papers of the Symposium on VLSI Technology, pp. 65-66, 2003.
28. A. Hassan, R. Ahmed, H. Mostafa, H. A. H. Fahmy and A. Hussien, "Performance evaluation of dynamic partial reconfiguration techniques for software defined radio implementation on FPGA," 2015 IEEE International Conference on Electronics, Circuits, and Systems, ICECS, Cairo, pp. 183-186, 2015.
29. R. Ahmed, H. Mostafa and A. H. Khalil, "Impact of dynamic partial reconfiguration on CONNECT Network-on-Chip for FPGAs," 2018 13th International Conference on Design & Technology of Integrated Systems In Nanoscale Era DTIS, Taormina, pp. 1-5, 2018.
30. Papamichael, M. K., and J. C. Hoe, "CONNECT: Re-examining Conventional Wisdom for Designing NoCs in the Context of FPGAs," in Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA'12, New York, NY, USA, pp. 37-46, ACM, 2012.

31. Salaheldin, A., K. Abdallah, N. Gamal, and H. Mostafa, "Review of NoC-Based FPGAs Architectures," IEEE International Conference on Energy Aware Computing Systems and Applications, pp. 1-4, 2015.
32. S. Abba and J. A. Lee, "Examining the Performance Impact of NoC Parameters for Scalable and Adaptive FPGA-Based Network-on-Chips," in 2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation, pp. 364–372, Sept 2013.
33. Gamal, N., H. A. H. Fahmy, Y. Ismail, T. Ismail, M. Mohie-Eldin, and H. Mostafa, "Design Guidelines for Soft Implementations to Embedded NoCs of FPGAs," IEEE International Design and Test Symposium, pp. 1-6, 2016.

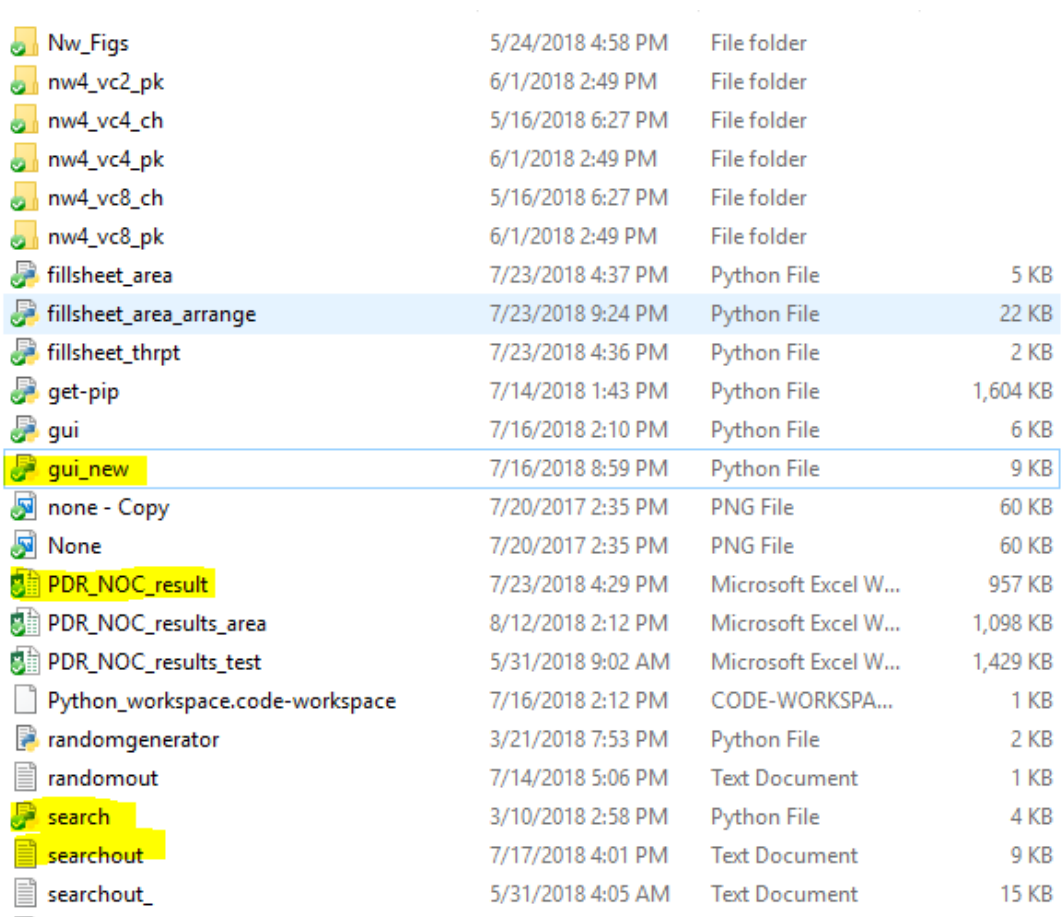
Appendix A: Reconfiguration tool user manual

This part contains the configuration tool user manual. All the needed steps for the user to install the tool and use it properly.

A.1 Environment Setup

Before using the tool, the environment need to be ready by installing the required compilers and placing the needed files and datasheets at their right place.

First, the tool is developed in Python. Therefore, the user need to install python in order to be able to run the tool successfully.



Nw_Figs	5/24/2018 4:58 PM	File folder	
nw4_vc2_pk	6/1/2018 2:49 PM	File folder	
nw4_vc4_ch	5/16/2018 6:27 PM	File folder	
nw4_vc4_pk	6/1/2018 2:49 PM	File folder	
nw4_vc8_ch	5/16/2018 6:27 PM	File folder	
nw4_vc8_pk	6/1/2018 2:49 PM	File folder	
fillsheet_area	7/23/2018 4:37 PM	Python File	5 KB
fillsheet_area_arrange	7/23/2018 9:24 PM	Python File	22 KB
fillsheet_thrpt	7/23/2018 4:36 PM	Python File	2 KB
get-pip	7/14/2018 1:43 PM	Python File	1,604 KB
gui	7/16/2018 2:10 PM	Python File	6 KB
gui_new	7/16/2018 8:59 PM	Python File	9 KB
none - Copy	7/20/2017 2:35 PM	PNG File	60 KB
None	7/20/2017 2:35 PM	PNG File	60 KB
PDR_NOC_result	7/23/2018 4:29 PM	Microsoft Excel W...	957 KB
PDR_NOC_results_area	8/12/2018 2:12 PM	Microsoft Excel W...	1,098 KB
PDR_NOC_results_test	5/31/2018 9:02 AM	Microsoft Excel W...	1,429 KB
Python_workspace.code-workspace	7/16/2018 2:12 PM	CODE-WORKSPA...	1 KB
randomgenerator	3/21/2018 7:53 PM	Python File	2 KB
randomout	7/14/2018 5:06 PM	Text Document	1 KB
search	3/10/2018 2:58 PM	Python File	4 KB
searchout	7/17/2018 4:01 PM	Text Document	9 KB
searchout_	5/31/2018 4:05 AM	Text Document	15 KB

Figure A.1: The Reconfiguration tool required files and sheets

Second, the Python Scripts “*search.py*” and “*gui_new.py*” need to be placed in the run directory as shown in Figure A.1. Additionally, the Sheet “*PDR_NOC_result.xlsx*” is required beside the python scripts in the same directory as it contains all the required network information. The tool output appears in the “searchout” text file.

A.2 Batch mode run

In the Batch mode, the user only needs to run the “*search.py*” script in the python terminal passing the expected network traffic and the desired throughput as in Figure A.2 (the terminal part).

```
>> python .\search.py ExpectedTraffic RequiredThroughput
```

A.2 GUI mode run

In the GUI mode, the user only needs to run the “*gui_new.py*” script in the python terminal. The GUI interface appears letting the user enter the expected network traffic and the desired throughput as in Figure A.2 (the GUI part).

```
>> python .\gui_new.py
```

A.3 Tool Output

The tool output in the GUI mode is the best fitting Network structure and configuration in addition to a list with all the fitting networks as in Figure A.2 (the output part). The Output list is the only output in the Batch mode run.

Traffic load: 80
Target throughput: 0.59
Show

No configuration selected

```

Python> python .\search.py 80 0.59
Python>

```

NW=	4,	VC=	2,	BD=	4,	PDR=	-,	which router=	none,	Config=	ffff
NW=	4,	VC=	2,	BD=	8,	PDR=	-,	which router=	none,	Config=	ffff
NW=	4,	VC=	2,	BD=	8,	PDR=	0,	which router=	corner,	Config=	fffe
NW=	4,	VC=	2,	BD=	8,	PDR=	12,	which router=	corner,	Config=	efff
NW=	4,	VC=	2,	BD=	8,	PDR=	15,	which router=	corner,	Config=	7fff
NW=	4,	VC=	2,	BD=	8,	PDR=	1,	which router=	edge,	Config=	fffd
NW=	4,	VC=	2,	BD=	16,	PDR=	-,	which router=	none,	Config=	ffff
NW=	4,	VC=	2,	BD=	16,	PDR=	0,	which router=	corner,	Config=	fffe
NW=	4,	VC=	2,	BD=	16,	PDR=	3,	which router=	corner,	Config=	fff7
NW=	4,	VC=	2,	BD=	16,	PDR=	12,	which router=	corner,	Config=	efff
NW=	4,	VC=	2,	BD=	16,	PDR=	15,	which router=	corner,	Config=	7fff
NW=	4,	VC=	2,	BD=	16,	PDR=	1,	which router=	edge,	Config=	fffd

Figure A.2: The Reconfiguration tool interface and output

Appendix B: Reconfiguration tool source code

This part contains the python configuration tool source code. The python source code requires the existence of the Connect NoC results data base for searching among it for the most suitable Noc structure fitting the desired network requirements.

For more information on how to use the Configuration tool, please refer to Appendix A which is the user manual for the reconfiguration tool.

The Source code:

```
from tkinter import *
from PIL import ImageTk, Image
from tkinter import messagebox
from openpyxl import load_workbook
import string
import sys
import random

def calculate(pktrate, thrpt):
    column = getPacketRateCol(pktrate)
    streamWriter=open('searchout.txt','w')
    minarea = -1
    which = 'NotFound'
    NW = -1
    VC = -1
    BD = -1
    temp = -1
    for r in range(4,ws1.max_row+1):
        cellval = ws1[column+str(r)].internal_value
        if thrpt < cellval:
            temp = ws1['AB'+str(r)].internal_value
            if minarea == -1 or minarea > temp:
                minarea = temp
                which = ws1['E'+str(r)].internal_value
                NW = ws1['A'+str(r)].internal_value
                VC = ws1['B'+str(r)].internal_value
```

```

BD = ws1['C'+str(r)].internal_value

streamWriter.write("NW= %2d, VC= %2d, BD= %3d, PDR= %4s, which router= %6s,
Config= %4s
\n"%(ws1['A'+str(r)].internal_value,ws1['B'+str(r)].internal_value,ws1['C'+str(r)].internal_value,str
(ws1['D'+str(r)].internal_value),str(ws1['E'+str(r)].internal_value),str(ws1['F'+str(r)].internal_value
)))
#print("NW=%2d      VC=%2d      BD=%3d      PDR=%4s      which=%6s
Config=%4s"%(ws1['A'+str(r)].internal_value,ws1['B'+str(r)].internal_value,ws1['C'+str(r)].internal_v
alue,str(ws1['D'+str(r)].internal_value),str(ws1['E'+str(r)].internal_value),str(ws1['F'+str(r)].internal_v
alue)))
#print(str(r) + ' NW=' + str(ws1['A'+str(r)].internal_value) + ' VC=' +
str(ws1['B'+str(r)].internal_value)+ ' BD=' + str(ws1['C'+str(r)].internal_value)+ ' PDR=' +
str(ws1['D'+str(r)].internal_value) + ' which=' + str(ws1['E'+str(r)].internal_value) + ' Config=' +
str(ws1['F'+str(r)].internal_value))
for r in range(4,ws2.max_row+1):
    cellval = ws2[column+str(r)].internal_value

    if thrpt < cellval:
        temp = ws2['AB'+str(r)].internal_value
        if minarea == -1 or minarea > temp:
            minarea = temp
            which = ws2['E'+str(r)].internal_value
            NW = ws2['A'+str(r)].internal_value
            VC = ws2['B'+str(r)].internal_value
            BD = ws2['C'+str(r)].internal_value

            streamWriter.write("NW= %2d, VC= %2d, BD= %3d, PDR= %4s, which router= %6s,
Config= %4s
\n"%(ws2['A'+str(r)].internal_value,ws2['B'+str(r)].internal_value,ws2['C'+str(r)].internal_value,str
(ws2['D'+str(r)].internal_value),str(ws2['E'+str(r)].internal_value),str(ws2['F'+str(r)].internal_value
)))
#print("NW=%2d      VC=%2d      BD=%3d      PDR=%4s      which=%6s
Config=%4s"%(ws1['A'+str(r)].internal_value,ws1['B'+str(r)].internal_value,ws1['C'+str(r)].internal_v
alue,str(ws1['D'+str(r)].internal_value),str(ws1['E'+str(r)].internal_value),str(ws1['F'+str(r)].internal_v
alue)))
#print(str(r) + ' NW=' + str(ws1['A'+str(r)].internal_value) + ' VC=' +
str(ws1['B'+str(r)].internal_value)+ ' BD=' + str(ws1['C'+str(r)].internal_value)+ ' PDR=' +
str(ws1['D'+str(r)].internal_value) + ' which=' + str(ws1['E'+str(r)].internal_value) + ' Config=' +
str(ws1['F'+str(r)].internal_value))

```

```

for r in range(4,ws3.max_row+1):
    cellval = ws3[column+str(r)].internal_value

    if thrpt < cellval:
        temp = ws3['AB'+str(r)].internal_value
        if minarea == -1 or minarea > temp:
            minarea = temp
            which = ws3['E'+str(r)].internal_value
            NW = ws3['A'+str(r)].internal_value
            VC = ws3['B'+str(r)].internal_value
            BD = ws3['C'+str(r)].internal_value

            streamWriter.write("NW= %2d, VC= %2d, BD= %3d, PDR= %4s, which router= %6s,
Config= %4s
\n"%(ws3['A'+str(r)].internal_value,ws3['B'+str(r)].internal_value,ws3['C'+str(r)].internal_value,str
(ws3['D'+str(r)].internal_value),str(ws3['E'+str(r)].internal_value),str(ws3['F'+str(r)].internal_value
)))
            #print("NW=%2d      VC=%2d      BD=%3d      PDR=%4s      which=%6s
Config=%4s"%(ws['A'+str(r)].internal_value,ws['B'+str(r)].internal_value,ws['C'+str(r)].internal_v
alue,str(ws['D'+str(r)].internal_value),str(ws['E'+str(r)].internal_value),str(ws['F'+str(r)].internal_v
alue)))
            #print(str(r) + ' NW=' + str(ws['A'+str(r)].internal_value) + ' VC=' +
str(ws['B'+str(r)].internal_value)+ ' BD=' + str(ws['C'+str(r)].internal_value)+ ' PDR=' +
str(ws['D'+str(r)].internal_value) + ' which=' + str(ws['E'+str(r)].internal_value) + ' Config=' +
str(ws['F'+str(r)].internal_value))
            streamWriter.close()
            return minarea, which, NW, VC, BD

```

```

def getPacketRateCol(val, row=2):
    alpha = list(string.ascii_uppercase)
    alpha = alpha[6:]
    for col in alpha:
        curr_cell = ws1[col+str(row)].internal_value
        if val > curr_cell:
            continue
        else:
            return col

```

```

def validate(event=None):
    fail = False
    try:
        trafficLoad = int(entry1.get())
        if trafficLoad < 0 or trafficLoad > 100:
            raise ValueError
    except ValueError:
        if entry1.get() == "":
            messagebox.showerror("Traffic load invalid input", "Traffic load shouldn't be left empty.")
        else:
            messagebox.showerror("Traffic load input error", "You should enter value between: 0 and
100")

    fail = True

    try:
        throughput = float(entry2.get())
        if throughput < 0 or throughput > 1:
            raise ValueError
    except ValueError:
        if entry1.get() == "":
            messagebox.showerror("Target throughput invalid input", "Target throughput shouldn't be
left empty.")
        else:
            messagebox.showerror("Target throughput input error", "You should enter value
between: 0 and 1")
    fail = True

    if fail == True:
        return

minarea, which, NW, VC, BD= calculate(trafficLoad, throughput)
if minarea == -1:
    panel1.config(text="No configuration selected",font='Helvetica 14 bold')
    panel2.configure(image="")
    fail = True
    messagebox.showerror("Configuration not found", "The networks available don't meet the
required configurations")

```



```

# return #TEMP
if not fail:
    panel1.config(text = "Minimum Area="+str(minarea)+', '+'NW="+str(NW)+',
'+ "VC="+str(VC)+', '+'BD="+str(BD), width = "50")
    #####image
identifier#####
    testval = random.randint(1,101)
    if testval > 50:
        number = 2
    else:
        number = 1
    path = "test" + str(number) + ".png"

#####
#####

    image = Image.open(path)
    image = image.resize((820, 820), Image.ANTIALIAS) #The (250, 250) is (height, width)
    img = ImageTk.PhotoImage(image)
    #img = ImageTk.PhotoImage(Image.open(which))
    panel2.configure(image=img)
    panel2.image = img # keep a reference!

if __name__ == '__main__':
    master = Tk()
    master.title('Re-Configuration Tool')
    master.geometry("850x1000")
    master.resizable(0,0)
    frame1 = Frame(master,width=300)
    frame1.pack(side=TOP, padx=5, pady=5)
    label1 = Label(frame1, width=10, text='Traffic load:',anchor=W, padx=50)
    label1.pack(side=LEFT)
    entry1 = Entry(frame1)
    entry1.pack(side=LEFT)
    entry1.bind('<Return>',validate)
    frame2 = Frame(master,width=300)
    frame2.pack(side=TOP, padx=5, pady=5)
    label2 = Label(frame2, width=10, text='Target throughput:',anchor=W, padx=50)

```

```

label2.pack(side=LEFT)
entry2 = Entry(frame2)
entry2.pack(side=LEFT)
entry2.bind('<Return>',validate)
frame3 = Frame(master)
frame3.pack(side=TOP, fill=X, padx=5, pady=5)
button1 = Button(frame3,text="Show",command=validate)
button1.pack()
button1.bind('<Return>',validate)
frame4 = Frame(master,borderwidth=1,relief= GROOVE,padx=10,pady=10)
frame4.pack(side=TOP, fill=BOTH, padx=5, pady=5)
panel1 = Label(frame4, text="No configuration selected",font='Helvetica 14 bold')
panel1.pack()
frame5 = Frame(frame4,borderwidth=1,padx=10,pady=10)
frame5.pack(side=TOP, fill=X, padx=5, pady=5)
panel2 = Label(frame5, text="",font='Helvetica 14 bold')
panel2.pack()
wb = load_workbook(filename = 'PDR_NOC_results.xlsx', data_only=True)
ws1 = wb['Bypass 4x4 vc2']
ws2 = wb['Bypass 4x4 vc4']
ws3 = wb['Bypass 4x4 vc8']
master.mainloop()

#pip install Pillow

```

ملخص الرسالة

مع الزيادة المطردة كل عام في كثافة التصميم داخل النظم على الرقائق، يتم تقديم بنية تصميم شبكة توصيل المعلومات كبنية اتصالات موثوقة لمواجهة تحديات أنظمة التصميم المعقدة. يتم تفضيل إتباع نهج تصميم شبكة توصيل المعلومات على نهج الاتصالات التقليدية لأجل قابلية توسعها، ونمطيتها المحسنة، وأدائها الأفضل.

من ناحية أخرى، فإن التقدم في المصفوفات القابلة للبرمجة و القابلة لإعادة التكوين بشكل ديناميكي يتيح إعادة تشكيل التصاميم المستخدمة أثناء وقت التشغيل. يسمح إعادة التشكيل الديناميكي الجزئي بمزيد من المرونة للمكونات المختلفة للتصميم ويوفر استغلالاً أفضل للمساحة والمزيد من تحسين استخدام الطاقة. كذلك، يتيح استخدام إعادة التشكيل الديناميكي الجزئي تطوير خوارزميات التصاميم التكميلية بناءً على متطلبات وقيود التطبيقات المختلفة.

إدخال مفهوم إعادة التهيئة أو التشكيل الديناميكي الجزئي في واحدة من أكثر بنيات التصميم انتشاراً وإقبالاً مثل شبكات توصيل المعلومات يعتبر فرصة جيدة لاكتساب أقصى استفادة منهما. المرونة العالية وإعادة التشكيل الكامل لشبكات توصيل المعلومات القابلة للتشكيل يمكن أن يفتح الباب أمام شبكات توصيل معلومات قابلة للتكيف بشكل كامل و يلائم عددًا كبيراً من التطبيقات وفقاً لاحتياجات ومتطلبات وقت التشغيل.

إعادة تشكيل المصفوفات القابلة للبرمجة المبنية على أساس خلايا الذاكرة الالكترونية تعد أقوى ميزة على تصاميم الدوائر الالكترونية المصنعة لتطبيقات محددة. يؤكد إعادة التشكيل الجزئي الديناميكي على هذه الميزة بإضافة المزيد من المرونة أثناء مرحلة التشغيل. توفر عائلة (Xilinx) من المصفوفات القابلة للبرمجة أربعة أساليب لتنفيذ إعادة التشكيل الديناميكي الجزئي: SelectMAP، Serial Mode، JTAG، و ICAP. في هذه الرسالة، يتم عرض كل من هذه التقنيات وتقييمها واختبارها باستخدام برنامج التفسير الالتفافي، وهو جزء أساسي من نظم الراديو المحددة ببرامج، والذي أصبح أكثر التطبيقات الواعدة لإعادة التشكيل الديناميكي الجزئي. تم إجراء التجارب باستخدام عائلة (Xilinx Virtex 5) من المصفوفات القابلة للبرمجة لتحديد و قياس المفارقات بين الأداء وتكلفة المساحة الإضافية عن طريق إضافة وحدة تحكم و إعادة تهيئتها داخل أو خارج المصفوفات القابلة للبرمجة. يتبين من النتائج أن أداء كل أسلوب لا يعتمد على نوع التصميم، ولكنه يتناسب فقط مع حجم المنطقة الجزئية المعاد تشكيلها و التي تم اختيارها أثناء مرحلة تحديد الأماكن و المسارات للتصميم.

الهدف الرئيسي من هذه الأطروحة هو تقديم دعم إعادة التشكيل الديناميكي الجزئي إلى شبكة توصيل المعلومات (CONNECT). هذه الأطروحة أيضاً تدرس تأثير إعادة التشكيل الديناميكي الجزئي على أداء الشبكة بمختلف عوامل ضبطها. تعمل قابلية إعادة التشكيل أثناء التشغيل على زيادة مرونة شبكات توصيل المعلومات وتسمح بالتشكيل الكامل لشبكات توصيل المعلومات عن طريق التصاميم القابلة لإعادة التشكيل الديناميكي. بالمقارنة مع شبكات تواصل المعلومات الثابتة، تؤدي شبكات تواصل المعلومات القابلة لإعادة التشكيل الديناميكي إلى توفير المساحة عن طريق استغلال جزء من الشبكة عند عدم الحاجة إليه أثناء وقت التشغيل. تم تنفيذ أداة إعادة التشكيل والتي تسمح للمستخدم بتحديد هيكل الشبكة الأمثل لكل تطبيق. تتطلب أداة إعادة التشكيل من المستخدم الحمل المتوقع و السعة المطلوبة من الشبكة كمدخلين للأداة. هذان المدخلان يتم استخدامهم في تحديد الشبكة الأنسب من ناحية المساحة.



رامي أحمد على محمد

١٩٨٩\٦\٢٣

مصرى

٢٠١٢\١٠\١

٢٠١٩\ ١

هندسة الإلكترونيات و الإتصالات الكهربائية
ماجستير العلوم

مهندس:

تاريخ الميلاد:

الجنسية:

تاريخ التسجيل:

تاريخ المنح:

القسم:

الدرجة:

المشرفون:

أ.د. أحمد حسين محمد

د. حسن مصطفى حسن

المتحنون:

(المشرف الرئيسي)

أ.د أحمد حسين محمد

أستاذ الإلكترونيات بكلية الهندسة – جامعة القاهرة

(المتحن الداخلى)

أ.د أمين محمد نصار

أستاذ الإلكترونيات بكلية الهندسة – جامعة القاهرة

(المتحن الخارجى)

أم عمرو طلعت عبد الحميد

أستاذ مساعد بقسمى الإلكترونيات و الشبكات – الجامعة الألمانية بالقاهرة

عنوان الرسالة:

تصميم لشبكة تواصل معلومات معاد تشكيلها للجيل القادم من المصفوفات القابلة للبرمجة باستخدام إعادة التشكيل الجزئى الديناميكي

الكلمات الدالة:

إعادة التشكيل الجزئى الديناميكي ؛ شبكات توصيل المعلومات على الرقائق الالكترونية ؛ دوائر مصفوفات البوابات المنطقية

ملخص الرسالة:

الهدف الرئيسى من هذه الأطروحة هو تقديم دعم إعادة التشكيل الديناميكي الجزئى إلى شبكة توصيل المعلومات (CONNECT). بالإضافة لذلك، هذه الأطروحة تدرس تأثير إعادة التشكيل الديناميكي الجزئى على أداء الشبكة بمختلف عوامل ضبطها. بالمقارنة مع شبكات تواصل المعلومات الثابتة، تؤدي شبكات تواصل المعلومات القابلة لإعادة التشكيل الديناميكي إلى توفير المساحة عن طريق استغلال جزء من الشبكة عند عدم الحاجة إليه أثناء وقت التشغيل. تم تنفيذ أداة إعادة التشكيل والتي تسمح للمستخدم بتحديد هيكل الشبكة الأمثل لكل تطبيق.

تصميم لشبكة تواصل معلومات معاد تشكيلها للجيل القادم من المصفوفات القابلة
للبرمجة باستخدام إعادة التشكيل الجزئي الديناميكي

إعداد
رامى أحمد على محمد

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة ماجستير العلوم
في
هندسة الإلكترونيات و الإتصالات الكهربائية

يعتمد من لجنة الممتحنين:

المشرف الرئيسى	الاستاذ الدكتور: أحمد حسين محمد
	أستاذ الإلكترونيات بكلية الهندسة - جامعة القاهرة
الممتحن الداخلى	الاستاذ الدكتور: أمين محمد نصار
	أستاذ الإلكترونيات بكلية الهندسة - جامعة القاهرة
الممتحن الخارجى	الاستاذ المساعد: عمرو طلعت عبد الحميد
	أستاذ مساعد بقسمى الإلكترونيات و الشبكات - الجامعة الألمانية بالقاهرة

كلية الهندسة - جامعة القاهرة
الجيزة - جمهورية مصر العربية

٢٠١٩

تصميم لشبكة تواصل معلومات معاد تشكيلها للجيل القادم من المصفوفات القابلة
للبرمجة باستخدام إعادة التشكيل الجزئي الديناميكي

إعداد
رامي أحمد على محمد

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة ماجستير العلوم
في
هندسة الإلكترونيات و الإتصالات الكهربائية

تحت اشراف

د. حسن مصطفى حسن

أ.د أحمد حسين محمد

مدرس

قسم هندسة الإلكترونيات
و الإتصالات الكهربائية
كلية الهندسة - جامعة القاهرة

أستاذ

قسم هندسة الإلكترونيات
و الإتصالات الكهربائية
كلية الهندسة - جامعة القاهرة

كلية الهندسة - جامعة القاهرة
الجيزة - جمهورية مصر العربية

٢٠١٩



تصميم لشبكة تواصل معلومات معاد تشكيلها للجيل القادم من المصفوفات القابلة
للبرمجة باستخدام إعادة التشكيل الجزئي الديناميكي

إعداد

رامى أحمد على محمد

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة ماجستير العلوم
في
هندسة الإلكترونيات و الإتصالات الكهربائية

كلية الهندسة - جامعة القاهرة
الجيزة - جمهورية مصر العربية
٢٠١٩