**VIETNAM NATIONAL UNIVERSITY, HANOI**

**UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

[](http://test.uet.vnu.vn/)

**Thi-Hue Nguyen**

**RTL MODEL AND IMPLEMENTATION OF RECONFIGURABLE FABRICS FOR THE ROUTER USED IN NETWORK-ON-CHIPS.**

**Major: Electronics and Telecommunication Engineering**

## HA NOI - 2017

*(Upper case, bold, 12pt, centered)*

**VIETNAM NATIONAL UNIVERSITY, HANOI**

**UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

# **Thi-Hue Nguyen**

**RTL MODEL AND IMPLEMENTATION OF RECONFIGURABLE FABRICS FOR THE ROUTER USED IN NETWORK-ON-CHIPS.**

**Major: Electronics and Telecommunication Engineering**

**Supervisor: Dr. Hung Kiem Nguyen**

## HA NOI - 2017

*(Upper case, bold, 12pt, centered)*

**AUTHORSHIP**

*“I hereby declare that the work contained in this thesis is of my own and has not been previously submitted for a degree or diploma at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no materials previously published or written by another person except where due reference or acknowledgement is made.”*

Signature:………………………………………………

**SUPERVISOR’S APPROVAL**

*“I hereby approve that the thesis in its current form is ready for committee examination as a requirement for the Bachelor of Electronics and Communications Engineering degree at the VNU University of Engineering and Technology.”*

Signature:………………………………………………

**ACKNOWLEDGEMENT**

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Hung-Kiem Nguyen. Without his assistance and dedicated involvement in every step throughout the process, this thesis has never been accomplished. I am so deeply grateful for his valuable guidance and understanding over these past two years.

I would also like to give special gratitude to Professor. Xuan-Tu Tran for providing me the opportunity of working in VNU Key Laboratory Smart Integrated Systems and giving the helpful advice when I get stuck.

I also thank all member of SIS Lab who always facilitates us to do the thesis, answer my question in a family way and share their experience for me as well as make me comfortable and better in studying.

Subsequently, I want to give my sincerely thanks to the faculty members and staffs of the Faculty of Electronic and Telecommunication, VNU-UET for their enthusiasm to guide me to for the background of knowledge and create the best studying conditions for us.

This work has been supported by Vietnam National University, Hanoi (VNU), under Project No. QG.16.33

Sincerely,

Thi-Hue Nguyen

**ABSTRACT**

With the advances in semiconductor technology, it allows to increase the number of IP cores, memory and processing elements which is integrated in a small chip. In this trend, the traditional communication architectures are more inadequate to meet increasing challenges of the System on Chip (SoC) design in terms of latency, power performance, cost and reliable data transmission. Network-on-Chips (NoC) are considered as a promising communication paradigm for the Ultra Large-Scale Integration System-on-Chips. A NoC offers much higher flexibility compared with the traditional communications paradigms. In order to implement a competitive NoC architecture, the router micro-architecture has a big impact to the performance and implementation cost of NoC. The router micro-architecture that not only guarantees the required performance but also improves the efficient use of the hardware resources is a key issue in NoC design. Recently, researchers have been paying a lot of attention to the design methodologies for reconfigurable routers. These methodologies focus on the techniques that allow NoCs to autonomously adapt their structure and behavior to the dynamic environment at runtime.

In this thesis, I propose and implement reconfigurable fabrics for a multi-mode router. The router is dynamically reconfigurable to exchange between switching schemes, arbitrating strategy and routing algorithms at run-time in order to meet the performance goals. I modeled the reconfigurable fabrics of router at Register Transfer Level in VHDL language and then synthesized on Xilinx Virtex-7 FPGA technology. The experiment results show that this router can operate reliably and improve the performance when compared with the generic router.

**Keyword**: **Network-on-Chip, System-on-Chip, reconfigurable router, virtual channel, router micro-architecture.**

**TABLE OF CONTENTS**

[List of Figures ix](#_Toc482635745)

[List of Tables xi](#_Toc482635746)

[ABBREVATIONS xii](#_Toc482635747)

[INTRODUCTION 1](#_Toc482635748)

[BACKGROUND AND RELATED WORK 3](#_Toc482635749)

[1.1. Overview of Network-on-Chip 3](#_Toc482635750)

[1.1.1. NoC architecture 3](#_Toc482635751)

[1.1.2. Network topology 5](#_Toc482635752)

[1.1.3. Switching techniques 6](#_Toc482635753)

[1.1.4. Credit-based flow control 9](#_Toc482635754)

[1.1.5. Routing technique 10](#_Toc482635755)

[1.2. Reconfigurable Architecture 12](#_Toc482635756)

[RECONFIGURABLE ROUTER ARCHITECTURE 14](#_Toc482635757)

[2.1. Reconfigurable Router Micro-architecture 15](#_Toc482635758)

[2.2. Reconfigurable fabrics 16](#_Toc482635759)

[2.2.1. Reconfigurable VC Buffer 16](#_Toc482635760)

[2.2.2. Reconfigurable Route Computation Unit 19](#_Toc482635761)

[2.2.3. VC Arbiter 23](#_Toc482635762)

[2.2.4. Switch arbiter 25](#_Toc482635763)

[SIMULATION AND EVALUATION 26](#_Toc482635764)

[3.1. Simulation script 26](#_Toc482635765)

[3.2. Simulation result 28](#_Toc482635766)

[3.2.1. Evaluation of reconfigurable fabrics 28](#_Toc482635767)

[3.2.2. NoC Evaluation 31](#_Toc482635768)

[3.3. Synthesis 33](#_Toc482635769)

[CONCLUSION 36](#_Toc482635770)

[REFERENCES 37](#_Toc482635771)

List of Figures

[Figure 1.1: A 2D-mesh topology NoC . 4](#_Toc483386192)

[Figure 1.2: Common network-on-chip topologies. 5](#_Toc483386193)

[Figure 1.3: Message composition. 6](#_Toc483386194)

[Figure 1.4: Switching technique 7](#_Toc483386195)

[Figure 1.5: Virtual channel. 9](#_Toc483386196)

[Figure 1.6: Credit-based flow control. 10](#_Toc483386197)

[Figure 1.7: XY routing alogorithm. 11](#_Toc483386198)

[Figure 1.8: A basic 4-stage pipelined router 12](#_Toc483386199)

[Figure 1.9: Pipelined stage with Look-ahead routing 12](#_Toc483386200)

[Figure 2.1: (a) Router I/O interface, and (b) Model of 2D-mesh NoC. 14](#_Toc483386201)

[Figure 2.2: Router micro-architecture 15](#_Toc483386202)

[Figure 2.3: Multi FIFO architectre. 18](#_Toc483386203)

[Figure 2.4: Routing computation unit. 20](#_Toc483386204)

[Figure 2.5: XY routing architecture. 22](#_Toc483386205)

[Figure 2.6: VC arbiter model. 23](#_Toc483386206)

[Figure 2.7: Switch arbiter model. 25](#_Toc483386207)

[Figure 3.1: A 8×8-router NoC evaluation platform. 26](#_Toc483386208)

[Figure 3.2: A NI with dummy IP core. 27](#_Toc483386209)

[Figure 3.3: VC\_buffer simulation 29](file:///C:\Users\admin\Google%20Drive\Khoaluan_NguyenHue_20170515.docx#_Toc483386210)

[Figure 3.4: Look ahead routing algorithms 30](file:///C:\Users\admin\Google%20Drive\Khoaluan_NguyenHue_20170515.docx#_Toc483386211)

[Figure 3.5: VC arbiter simulation 30](#_Toc483386212)

[Figure 3.6: Switch arbiter Simulaton 31](file:///C:\Users\admin\Google%20Drive\Khoaluan_NguyenHue_20170515.docx#_Toc483386213)

List of Tables

[Table 2.1: Description I/O signals of Multi-FIFO 18](#_Toc481938970)

[Table 2.2: Description I/O signal of RC block 20](#_Toc481938971)

[Table 2.3: VCID caculating 22](#_Toc481938972)

[Table 2.4: VC arbiter I/O signal 24](#_Toc481938973)

[Table 3.1: Comparison of Throughput between generic router and reconfigurable router. 32](#_Toc481938974)

[Table 3.2: Comparison of Latency between generic router and reconfigurable router 32](#_Toc481938975)

[Table 3.3: VC buffer resoucre by using Xilinx Virtex-7 technology 33](#_Toc481938976)

[Table 3.4: RC resource by using Xilinx Virtex-7 technology 34](#_Toc481938977)

[Table 3.5: Switch arbiter and VC arbiter resource by using Xlilinx Virtetx-7 FPGA 34](#_Toc481938978)

[Table 3.6: Comparsion resource by using xilinx Virtex-7 technology between generic and reconfigurable router 34](#_Toc481938979)

ABBREVATIONS

2D 2-Dimensions

CFS Conflict Sensing

IP Intellectual Property

NI Network interface

NoC Network- on- Chip

PE Processing Element

QoS Quality-of-Service

RTL Register Transfer Level

SAF Store-and-Forward

SoC System-on-Chip

VC Virtual Channel

VCT Virtual-Cut-Through

INTRODUCTION

Nowadays, the development of semiconductor technology allows to integrate a large number of transistor to the complex System-on-Chip (SoC) for many applications with the required dimension of a single chip as small as possible. The traditional communications such as shared-buses, point-to-point interconnections was used as communications infrastructure for SoC, but as silicon technology advances further, they are no more suitable because of their poor scalability, low performance and communication bottleneck in highly connected multi-core system. In bus-based communication, using common bus can reduce the amount of resources needed in the system. However, the additional delay increased by the bus arbitration, can drastically affect the performance of the system. With point-to-point interconnection, number of links may increase exponentially with the increase in number of cores, so this increase its complexity and makes the approach only possible for an offline temporal placement, where all the configurations can be defined and implemented at compile-time. Power grows for each communication event as more units attached will increase the capacitive load.

Network-on-Chip (NoC) has been suggested an alternative communication to provide a low-latency, low power, high bandwidth by using the benefits of a packet-based communication network. Especially, when the on-chip integration is not only more and more increasing with billion transistors, but also apply the increasingly applications, ensured Quality-of-Service (QoS). The on-chip communication should aim at providing high throughput, low latency, scalability, low power consumption. However, standard NoCs architecture seems not flexible enough to support dynamic environment where communication characteristics can strongly changing in run-time. Recently, researchers concentrate to the development of run-time methodologies for reconfigurable NoCs that allow NoCs autonomously adapt their structure and behavior during the period of their operation [3].

In this thesis, I proposed a reconfigurable router for NoCs which can support different multi-mode in order to ensure the NoC bounds between the cost (area, power) and performance (latency, throughput and reliability). My main contribution is to implement RTL model of reconfigurable fabrics and evaluate the reconfigurable router in comparison with the generic router.

The rest of this thesis is organized as follows. Chapter 1 provides theoretical background, giving an overview of the NoC and its basic concept. Besides, the knowledge about reconfigurable computing is also reviewed in this chapter. In chapter 2, I give the reconfigurable router micro-architecture with the general operation and I show the design of each reconfigurable fabric in detail. Chapter 3 describes the simulation of each module which I perform and synthesized result by using Xilinx Vertex-7 FPGA technology. I compare the evaluation of reconfigurable router with the generic router’s to show the difference and advances of reconfigurable computing. Finally, the conclusions and directions for future work are addressed in the final section of the thesis.

# 

BACKGROUND AND RELATED WORK

In this chapter, we will discuss on basic concepts of Network–on-Chip (NoC) paradigm and reconfigurable computing. The NoC architecture and communication control mechanism will also be presented.

## Overview of Network-on-Chip

The Network–on-Chips are abstraction of communication among on-chip components and must satisfy QoS requirements and implementation cost. Network QoS consist of latency, throughput, and reliability, whereas the implementation cost consist of power consumption, area size and efficient using of resource. In order to provide reliability and high performance, NoCs are usually built on the packet-based communication mechanism [3]. In this chapter, we discuss the basic background for NoC architecture such as topology, routing technique, switching technique, flow control and overview about reconfigurable computing.

### NoC architecture

There are three basic components in a generic NoC architecture: routers, processing elements and physical links. In NoC, each of routers is attached to the next router and processing element through physical links with a specified topology. Routers and links are employed for providing communication infrastructure for processing elements in other words they organize a NoC. The example NoC architecture in 2D mesh topology is shown in Figure 1.1. The design of an on-chip network can be featured by its communication mechanism used for packet forwarding such as topology, routing algorithm, flow control and switching technique.

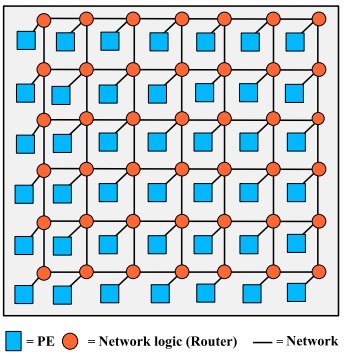


Figure 1.1: A 2D-mesh topology NoC [1].

A physical link connects between router and processing element in the network. Processing element can be either an IP core, a memory block, an embedded programmable logic or any custom hardware block. Each processing element has a unique address, which is similar to the address of router to which it is connected. Router is the most important building block in NoC. A router is responsible for routing data from a source port to its destination port according NoC topology. It consists of a given set of components such as the buffer to temporarily store packets, a controller that determines how to forward the packet and arbiter to arbitrate when congestion happens. Therefore, the design of router architecture is very important which has efficient impact to the NoC’s performance.

### Network topology

A NoC can be characterized by the structure of the router connections and this structure or organization is called topology. The on-chip network topology determines the arrangement of routers and processors on the device and the way those processors are connected together. The router can be connected in direct or indirect topologies. In this section, I will concentrate the direct topologies. Figure 1.2 shows three commonly used on-chip direct topologies such as ring topology, mesh topology and torus topology.

In direct topology, each router is associated to a processor and this pair can be seen as a node in the network. Each node is directly connected to a fixed number of neighbor nodes. Only the routers are involved in the communication in a direct topology and the communication is based on the routing algorithm implemented by the routers.



Figure 1.2: Common network-on-chip topologies (a) Ring, (b) 2-D mesh, (c) 2-D tourus [2].

Due to advantage and disadvantage of each topology, the physical metrics are used to compare between them such as router degree, maximum channel load, path diversity and bisection bandwidth. In this thesis, with the packet-based communication network, we proposed to use 2-D mesh network topology (Figure 1.1) because it naturally fits the tile-based architecture on the chip. For instance, it is easy to be implemented because of simple routing strategies and good network scalability. It is also a popular choice for the designers want to minimize the router diameter of network as well as improve its bandwidth.

### Switching techniques

Switching techniques define the flow of data through routers in the network transmitted from source to destination node. When a message is injected into the network, it is first segmented into packets, which are then divided into fixed-length flits. The breakdown of messages to packet and packets to flits is depicted in Figure 1.3. The packet will consist of a head flit that contains the destination address and routing information, body flits and a tail flit that indicates the end of a packet. Flits can be further broken down into phits, which are physical unit and correspond to the physical channel width.

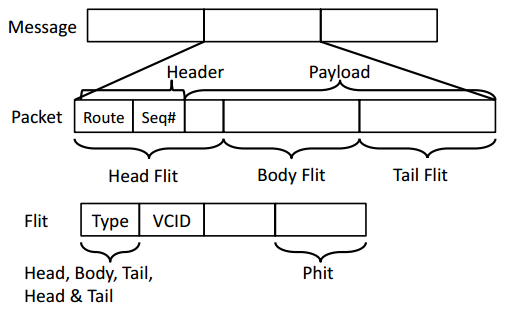


Figure 1.3: Message composition.

Switching techniques are classified by the level of message transferred through the node (a technique that operates on message, packet and flit level), so switching techniques can be divided into two objects: message-based switching (circuit switching) technique and packet-based switching technique (Figure 1.4). In message based switching technique, circuit switching reverses a physical path from source to destination before it starts transmitting data and holds it till all data have been delivered. Once the problem reaches the destination (having successfully allocated the necessary links) and acknowledgement message will be transmitted back to the source. When the source receives the acknowledgement message, it will release the message which can then travel quickly through the network. Once the message has completed its traversal, the resources are deallocated. Circuit switching gives assurance of bandwidth throughout the duration of data delivery.



**Figure 1.4: Switching technique.**

Packet-based switching technique first break down messages into packets, then interleave these packet on the links, thus improving link utilization. Unlike circuit switching, the remaining techniques will require per-node buffering to store in flight- packets. Due to the complex of packet-switching technique, the problem called communication mode is the way a packet is forwarded from one network node to the next one. Three communication modes are common used as Store-And-Forward (SAF), Virtual-Cut-Through (VCT) and Wormhole.

* Store-and-Forward: In the strategy, the node waits to store the entire packet before forwarding any part of the packet to the next node. Therefore, Store-and-Forward ensure that the buffer size at each node is sufficient to store the whole packet. The size of buffer may increases in the increase of packet’s size.
* Virtual-Cut-Through: To reduce the delay packets at each node, VCT flow control allows transmission of a packet to processed in the next node before the entire packet is received at the current router. Although latency experienced by a packet is thus drastically reduced over Store-and-Forward flow control, buffer size is enough to store whole packet in the case packet must not transferred to the next node.
* Wormhole: Like VCT, wormhole cuts packet through flits, allowing flits to move on to the next router before the entire packet is received at the current node. The node makes the routing decision and forwards the packet as soon as the header arrives. The subsequent flits follow the header as they arrive. However, unlike VCT and SAF, wormhole allocates storage and bandwidth to flits instead entire packets. The wormhole routing schemes just offers size of the flits buffer that is smaller than the packet size. This reduces the latency within the router, but in case of packet stalling, many links risk to be locked at once.

In NoC, they usually use Wormhole and Virtual-Cut-Through because of buffer response. In order to ensure the QoS in a NoC, each packet is also assigned a certain priority level so that the arbiter can grant the switch to only one specific packet when many packets require the connection to the same output port. The problem with the above NoC is starvation. The starvation happens when a packet does not reach its destination because some resources do not grant access it due to its low priority. This problem becomes extremely serious in the NoC using the wormhole switching scheme. If a packet is blocked, all of resources occupied by it cannot be released for other packets. VCT packet switching can deal with the above problem by always ensuring enough buffer size in the all routers which the packet passes through. However, according to the research of Pratomo et al. in [9], VCT switching wastes a big amount of recourses in each router. In summary, there are significant variables in the performance between wormhole switching and virtual cut through. An implementation way is to propose the hybrid switching routers that combines some switching schemes in the same micro-architecture. Modarressi et al. in [10] proposed architecture of a hybrid switching router that dynamically changed between circuit switching and packet switching technique. However, in the thesis, we proposed the hybrid switching router that combines both virtual cut through and wormhole switching for the reconfigurable NoC [3]. The proposed router can dynamically reconfigure its switching scheme to the wormhole switching, the virtual-cut-through switching or combination of both depending on the traffic load and network status. Therefore, the physical path will be established and moved from source to destination during the transmitting time of packet. This router can reconfigure itself to adapt to the dynamic status of communication environment

In addition, Virtual channel have been first proposed as a solution for deadlock avoidance in flow data in NoC, but also applied to skirt around head-of-blocking in flow control, thus extending throughput. In input port of router, there is queue to store data before forwarding this data to the next router. Head-of-line blocking occurs where there is a single queue at each input when a packet at the head of the queue is blocked. Virtual channel is basically a separate queue in the router; multiple VCs share the physical link between two routers.

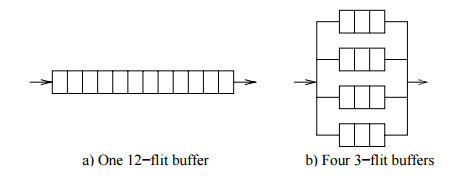


Figure 1.5: Virtual channel.

### Credit-based flow control

Network on chip are designed to provide high bandwidth and parallel communication, while minimizing the number of employed interconnection resources. However, NoC performance can significantly degrade in absence of an effective follow control mechanism. This follow control is responsible for preventing resource starvation and congestion inside the network. This goal is achieved by managing the flow of the packets that compete for resources, such as links and buffers in order to avoid deadlock or live lock locally in the node. Data flow control technique determines the way which packet is transferred in network. In this thesis, we proposed to use one of technique in end-to-end flow control is credit-based flow control.

The main concept of credit-based communication is to enable data forwarding only when there is a free data space in the destination buffer. Once there is no data space at the destination, the data should not be moved, enabling other data to utilization the communication resources. Before sending a packet, the transmission router sends credits to the received router and then received router send credits back to transmission router to confirm storing information status of router. The credit-based flow control model is described in Figure 1.6. Credits keep track of the number of the buffers available at the node 2, by sending a credit to the node 1 when a buffer is vacated (when a flit/packet leaves the router), and incrementing the credit count at the previous hop upon receiving the credit. When a flit departs the current router, the current router decrements the credit count for the appropriate downstream buffer.

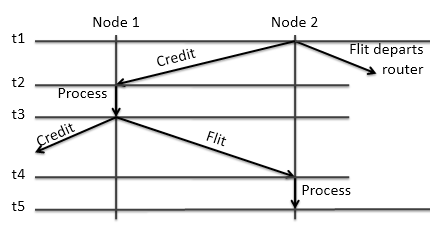


Figure 1.6: Credit-based flow control.

### Routing technique

The architecture of NoC based on topology, switching technique and routing algorithm. The routing algorithm is one of key factor which affects NoC network communication. Each router should respect the properties of the routing algorithm and forward the incoming packets to the appropriate output following the path decided by the routing algorithm (Duato et al. 1997). Another way, a routing algorithm determines how a path the data is routed from a source router to a destination router. Routing algorithm decides within each intermediate router which output channels must be selected for the incoming packet. According to how a path is defined to transmit packets, routing algorithm can be classified as deterministic or adaptive algorithm. Deterministic routing algorithm always chooses the same path between a pair of nodes and they are commonly used due to easy implementation. Adaptive routing algorithm use information about the network‘s state, which are taken into consideration.

Dimension-order-routing (DOR) algorithm is considered to be one of the most popular deterministic routing algorithms due to its simplicity for implementation and good performance according to average packet delay and throughput metrics. The algorithm determines to what direction packet are routed during every stage of the routing.



Figure 1.7: XY routing alogorithm.

XY routing (Figure 1.7) is a dimension-order-routing which routes packets first in X horizontal direction until it reaches in the router located in the same column (X) with the destination core and then via the Y axis until it reaches the destination. In YX routing, the implementation is reserved, means YX routing first route packets along Y axis and then X axis. The XY and YX routing suit well on 2D Mesh network. A packet has Head flit, Tail flit and Data flit where the Header flit has the destination node address. In 2D-mesh topology, each router is defined by a pair of co-ordinates (x axis, y axis). XY and YX routing has advantages such as simplicity of implementation, produce minimal paths without redundancy and never runs into deadlock and livelock since it is not possible for a cyclic channel dependency to arise if channels are acquired XY order. However, in dealing with real-life application, it has low power due to the ignoring of network status.

Commonly, all flit have to go through all stages: BW (Buffer Writing), RC (Routing computation), SA (Switching Arbitration), and CT (Crossbar Traversal). Therefore, it is reason why the communication latency and power consumption especially when we design a large network on chip size. To solve this problem, a smart pipeline design is proposed called “Look-Ahead routing”. Look-Ahead routing algorithm is calculating the next router which the packet will be forward to. Nadera Najib et. al. [6] proposed a partially adaptive look ahead routing algorithms for a low latency. In this design, the Look-Ahead routing is based on the target address and the deterministic routing algorithms. In Look-Ahead routing, the preferred output direction is pre-computed one clock cycle in advance using a congestion aware flow control. After pre-computing the next-port direction of the downstream, routing information will be routed and then embeds it in the flit. When arriving to the downstream node, this hot encoded next-port identifier will be used by SA to ask the grant for using the selected output port and reach the neighbor node. At the same time SA and RC calculate parallel. Figure 1.8 and Figure 1.9 have described the basic stage and stage with Look- Ahead routing, respectively. Therefore, Look-ahead routing is proposed to reduce the limitation of deterministic routing which the router’s increases the flit latency and power consumption, since any flit have to go thought all stage from source to destination.



Figure 1.8: A basic 4-stage pipelined router.



Figure 1.9: Pipelined stage with Look-ahead routing.

## Reconfigurable Architecture

Reconfigurable computing is rapidly establishing itself as a major discipline that covers various subjects of learning, including both computing science and electronic engineering. Reconfigurable computing is defined as the study of computation using of reconfigurable devices, such as field programmable gate arrays (FPGAs) for computing purposes. If a new application has to be computed, the device structure will be modified again to match the new application. The structure of reconfigurable device is changed by modifying all or part of the hardware at compile-time, design-time or run-time [1] in order to fit their new applications.

Reconfigurable computing systems often have impressive performance. In a reconfigurable system, the circuitry is optimized for the application then reduces the dimension and component amount to improve flexibility and performance. Several design methodologies have been proposed to deal with NoCs and can be classified into two main categories, including design-time methodologies and run-time methodologies. Design-time methodologies are generally aimed at designing the NoC for a specific application. Unfortunately, the application-specific NoCs are not flexible enough to support dynamic environments where communication characteristics need to adapt strongly to various contexts at runtime. Nowadays, run-time methodologies for reconfigurable NoCs are concentrated with a lot attention. These methodologies focus on the techniques that allow NoCs to autonomously adapt their structure and behavior during the period of their operation. The elements of a NoC that can be modified at run-time include reconfigurable topology, reconfigurable links and reconfigurable router [3].

In this thesis, we present the reconfigurable router that contains reconfigurable fabrics to dynamically change between other different operation modes to provide higher achievable throughput values. We proposed a hybrid switching scheme that combines both Virtual-cut-through and wormhole switching. The arbiter can alter between Round Robin and Priority mode to change priority of request in each time. The routing computation block implements Look-ahead routing based on XY routing or YX routing. These operation modes are varied in order to avoid congestion and improve higher performance.

A reconfigurable computing system typically contains one or more reconfigurable fabrics and a reconfigurable controller upon which custom functional units can be built. The controller executes sequential and noncritical code, while code that can be efficiently mapped to hardware can be executed by controller unit and then have mapped to the reconfigurable fabrics. The operation implementation in the reconfigurable fabrics can change over time to match better the new environment.

RECONFIGURABLE ROUTER ARCHITECTURE

|  |  |
| --- | --- |
| (a) | (b) |

Figure 2.1: (a) Router I/O interface, and (b) Model of 2D-mesh NoC.

In this chapter, we will discuss about the reconfigurable architecture router. Figure 2.1 show the I/O interface of each router proposed 2D-mesh topology NoC. Each router consists 5 input/output ports for four primary directions (labeled North, South, West, East) to connect with the neighbor routers and another port (labeled Local) for connecting with the processing element (PE). In this thesis, we proposed the reconfigurable router that is featured by combining of wormhole switching technique and virtual-cut-through, credit-based flow control, look-ahead routing algorithm based on XY routing or YX routing and four variable-size asynchronous virtual channels. Each port has its input channel and output channel for transmitting data with neighbor routers and processing element. Data packet moves into one input channel port of router by which it is forwarded to the output channel of other port. In this chapter, micro-architecture of the router will be described more in detail.

## Reconfigurable Router Micro-architecture

A router’s performance, cost and efficient primarily depends on the micro-architecture. Micro-architecture of reconfigurable router is determined by only once couple of input channel and output channel. The reconfigurable router’s microarchitecture is shown in Figure 2.2 this architecture model is proposed by the VNU Key Laboratory for Smart Integrated System [3].



Figure 2.2: Router micro-architecture.

The basic microarchitecture consists of three modules: *input channel*, *output channel* and *crossbar*. In addition, each module contains logic blocks that decide the overall routing strategy for moving data through routers in NoC. Firstly, in each router port, a couple of *I/O channel* is responsible for exchanging the data with next router and processing element by connecting to physical links via the *Receiver* and *Transmitter* block. After that, the packet is first collected and stored in *Input buffer*. Data from *Input buffer* is transferred to the *Output channel* through *Crossbar* by the physical channel established by *Flow control*. After traversing across the *Crossbar*, the granted packet reaches to the next router and placed in the *output buffer* before moving to the next router. This process repeats until the packet reaches the destination.

## Reconfigurable fabrics

The uniqueness of the reconfigurable router is the reconfigurable fabrics. The reconfigurable fabric consists of a set of re-configurable functional units, a reconfigurable interconnection and a flexible interface to connect the fabric to the rest of the system. In Figure 2.2, compared with the generic router, the reconfigurable router offers reconfigurable fabrics such as configurable input buffer, reconfigurable route computation unit, reconfigurable VC arbiter and switch arbiter, with the support from conflict sensing (CFS) unit and configuration controller. These improvements allow the router to be configured between Wormhole and VCT switching schemes, round-robin and priority arbitrating strategies, and routing algorithms at run-time. The configurable buffer is composed of an array of storage elements that can be flexibly changed into four VCs with variable size depending on applications. The routing computation can be implemented to reconfigure the operation mode in run-time. The VC arbiter and switch arbiter choses the served VC and input port, respectively, depend on each priority in run time.

### Reconfigurable VC Buffer

The reconfigurable input buffer is implemented by four asynchronous Virtual Channels, each virtual channel has its own FIFO (First-in-First-out) queue to store data temporarily. The each virtual channel size is variable in the range of one to the maximum packet size, which is defined by the NoC. In the case, there is a virtual channel become to blocked, DFC unit will forward the header flit of packet from VC to the configuration controller. The configuration controller encodes the information about the length of the current packet from header flit and then makes a decision to reconfigure the size of this virtual channel. After right the virtual channel has been expanded to fit the packet length, whole the blocking packet will fit into this virtual channel and blocking has been resolved. The FIFO supports some main features as follows:

* Each FIFO has got two interfaces, one for writing the data into the FIFO and the other for reading data out FIFO. Two interfaces Reading and writing data, one for writing data in FIFO, one for reading data from FIFO:
* Reading and writing data activities are controlled by 2 clocks and enable signal for reading and writing are independent. Reading data bus and writing data bus are independent.
* FIFO Full state: Writing activity to FIFO is allowed when FIFO is not full (Signal state “FULL” is not active).
* FIFO Empty state: Reading activity from FIFO is allowed when FIFO is not empty (Signal state “EMPTY’ is not active).
* FIFO Reset state: When signal “Reset” of system is active, FIFO come in Empty state, contemporary Reading and writing pointers is deleted to ‘0’.
* Capacity and width of data bus:
* FIFO Depth (number of memory elements in FIFO): determined by ADD\_WIDTH parameter of design.
* Data bus width: determined by WIDTH parameter of design.

The reconfigurable VC buffer architecture is presented in Figure 2.3. Four FIFOs has shared a same dual ported RAM memory to store data. In each dual–clock asynchronous FIFO, there are two gray counters for Writing and Reading operation to perform write address and read address through two pointer Wptr (Writing pointer), Rptr (Reading pointer). Gray-code counters to count two value pointers and then lead to show the FULL/ EMPTY state to maintain correct operation of FIFO. In addition, the declaration of input and output signal of VC buffer is represented in Table 2.1.



Figure 2.3: Multi FIFO architectre.

Table 2.1: Description I/O signals of Multi-FIFO.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signals** | **Direction** | **Width (bits)** | **Reset Value** | **Description** |
| **Write Interface** | | | | |
| IN\_DATA | IN | FLIT\_SIZE | Don’t care | Input FIFO data |
| WE | IN | 1 | 0 | Writing Enable FIFO data  WE = ‘1’: Enable Writing |
| W\_CLK | IN | 1 | Undefined | Writing Clock |
| FULL | OUT | 1 | 0 | State FIFO  FULL = 1: Full FIFO |
| **Read Interface** | | | | |
| Dataout | OUT | FLIT\_SIZE | Don’t care | Output FIFO data |
| RE | IN | 1 | 0 | Reading Enable FIFO data  RE = ‘1’: Enable Reading |
| R\_CLK | IN | 1 | Undefined | Reading Clock |
| EMPTY | OUT | 1 |  | State FIFO  EMPTY = 1: Empty FIFO |
| RAVC\_SEL | IN | 2 |  | Signal for choosing one VC to reading data, created by VC reading controller. |
| Reset | IN | 1 | 0 | Asynchronous reset signal. Low level active. Clear memory and status register inside FIFO |
| Generic | | | | |
| DATA\_WIDTH |  | FLIT\_SIZE |  | FIFO data width in bit |
| ADDR\_WIDTH |  |  |  | Address width |
| VC\_NUM |  |  |  | Numbers of Virtual Channels |

### Reconfigurable Route Computation Unit

In the Route computation (RC) unit, we proposed to use the Look-Ahead routing technique in order to reduce the communication latency and the power consumption while taking advantage at the same time of the simplicity of the conventional XY routing or YX routing which can be changed in run time. Routing computation unit can calculate the PortID of next router or current router which packet will move through. As Figure 2.1, each router has 5 ports and each port is identified by three properties: Port Name, Port ID and Priority. There are 2 steps in implementation Look-ahead routing.

|  |  |
| --- | --- |
| ***Algorithm 1: Pseudocode for assigning Next address***  *-- Evaluate node X address*  **IF** (cPortID = EAST) **THEN**  Xnext <= Xcur + 1;  **ELSIF** (cPortID = WEST)**THEN**  Xnext <= Xcur – 1;  **ELSE**  Xnext <= Xcur;  **END IF**;  *--Evaluate node Y address*  **IF** (cPortID = NORTH) **THEN**  Ynext <= Ycur - 1;  **ELSIF** (cPortID = SOUTH) **THEN**  Ynext <= Ycur + 1;  **ELSE**  Ynext <= Ycur;  **END IF**; | ***Algorithm2: Pseudo code fining new next Port , based on XY\_routing***  **IF** (Xnext = Xdest) **THEN**  **IF** (Ynext = Ydest) **THEN**  nPortID <= LOCAL;  **ELSIF** (Ynext > Ydest) **THEN**  nPortID <= NORTH;  **ELSE**  nPortID <= SOUTH;  **END IF**;  **ELSIF** (Xnext < Xdest) **THEN**  nPortID <= EAST;  **ELSE**  nPortID <= WEST;  **END IF;** |



**Figure 2.4: Routing computation unit.**

Table 2.2: Description I/O signal of RC block.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signals** | **Direction** | **Width (bits)** | **Reset Value** | **Description** |
| **I/O Interface** | | | | |
| CF\_RQST | IN | 1 | 0 | Reconfigurable signal for routing algorithms:   * 0: Using available signal * 1: Reconfigure routing information |
| RC\_EN | IN | 1 | 0 | Enable signal for RC unit, high level active:   * 1: enable active * 0: no active and reset all RC unit state follow rising edge CLK signal. |
| RC\_Done | OUT | 1 | 0 | Invalid signal for finishing calculating VC in RC unit, high level active:   * 0: not finish * 1: finished |
| nReset | IN | 1 | 0 | Asynchronous reset signal, low level active. |
| CLK | IN | 1 | - | Clock signal |
| Target\_X | IN | 3 | U | X-coordinate of destination router. |
| Target\_Y | IN | 3 | U | Y-coordinate of destination router. |
| Layer | IN | 1 | U | Packet signal information:  data layer or configuration layer:   * 0: Data Layer * 1: Configuration Layer |
| cPortID\_in | IN | 3 | U | PortID of output port (current router) which packet is forwarded through. Signal is taken from Header flit of packet. |
| cPortID | OUT | 3 | 000 | PortID of output port (current router) which packet is forwarded. Signal is generated when there is reconfigurable signal CF\_RQS= ‘1’). |
| nPortID | OUT | 3 | 000 | PortID of output port in next router which packet is sent. |
| VCID | OUT | 2 | 00 | Virtual channel ID of input port of next router, which is buffered for nPortID. |
| **Configuration** | | | | |
| XY-Mode | IN | 1 | 0 | Reconfigurable mode:   * 0: XY routing * 1: YX routing |

1. **XY-Routing block**

The RTL architecture of XY/YX-routing block is described in Figure 2.5, which calculates the Port ID of next router based on XY-routing algorithm or YX-routing algorithm. The routing mode (XY or YX) depends on the control signal XY\_Mode.



Figure 2.5: XY routing architecture.

1. **VCID Logic**

VCID Logic is implemented to calculate which VC in the input port of the next router will be corresponding to store packet. In this thesis, we use Look-ahead routing technique so the calculating VCID depends on both PortID of current router (cPortID) and PortID of next router (nPortID).

Table 2.3: VCID caculating.

|  |  |  |
| --- | --- | --- |
| **cPortID**  **(in the current Router)** | **PortID of the input port**  **(in the next router)** | **nPortID in the next router** |
| 000 | IP | Don’t care |
| 100 | 110 | If nPortID = 100 then VCID = ‘10’  Else VCID = nPortID[1:0] |
| 110 | 100 | VCID = nPortID[1:0] |
| 001 | 011 | * 100: VCID = ‘11’ * Others: VCID = nPortID[1:0] |
| 011 | 001 | * 100: VCID = ‘01’ * Others: VCID = nPortID[1:0] |

### VC Arbiter

An arbiter is responsible for granting the shared resource to only one of N requests. The arbiter has a very important role in the NoC router. The main object of arbitration algorithms is to ensure that only one object has access to the shared resource at any given time, all the others are forced to remain in the idle state until they are granted. Therefore, it resolves contention problem and enhance high performance. Arbiter can be operated in two modes: Round-Robin mode and Priority mode. With round-robin arbiter, the request that has just been serviced will have the lowest priority in the next round of arbitration. In Priority mode, each request will be assigned with one priority level that can be configured at run time.

The data from VC has been chosen to send the next router. However, in the case there is more than one VCs of an input channel has request to send data concurrently, the conflict will occur. As one VC is allowed to access the output port, contention is removed. In order to arbitrate the available VC, we can configure the VC arbiter to exchange between two modes, therefore to change the priority of each VC. The architecture of VC arbiter is presented in Figure 2.6 and the description of VC arbiter signal is shown in Table 2.1.



Figure 2.6: VC arbiter model.

Table 2.4: VC arbiter I/O signal.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Signals** | | **Direction** | | **Width (bits)** | **Reset Value** | | **Description** | |
| **Interface for Network** | | | | | | | | |
| RQST(i)  i={1,2,3,4} | IN | | 1 | | | 0 | | High active signal from VC(i) |
| Grant(i)  I={1,2,3,4} | OUT | | 1 | | | 0 | | Assert a grant signal for VC(i) |
| CLK | IN | | 1 | | | Undefined | | Clock |
| Enable | IN | | 1 | | | 0 | | Enable Arbiter  Enable = ‘1’: Enable. |
| PRI\_UPD | IN | | 1 | | | Undefined | | Control signal, able to upgrade priority level of each VC. |
| nReset | IN | | 1 | | | 0 | | Reset signal. Low level active. Clear memory and status register in FIFO. |
| **Interface for Configuration** | | | | | | | | |
| Din | IN | | Log2(VC\_NUM) | | |  | | Select input RQST |
| Mode | IN | | 1 | | |  | | Arbiter mode:  1’: Round-Robin mode  ‘0’: Priority mode |
| **GENERIC** | | | | | | | | |
| ARBITER\_  WIDTH | Constant | | Integer | | |  | | The number of bits used to encode the number of states that the arbiter can arbitrate. |
| ARBITER\_  SIZE | Constant | | Integer | | |  | | Arbitrator status can be arbitrated.  ARBITER\_SIZE < 2\*\*ARBITER\_WIDTH. |

### Switch arbiter



Figure 2.7: Switch arbiter model.

In Figure 2.2, *Crossbar* is called as the heart of router, which transfers data from input port to output port of router before forwarding to the next router. In *Crossbar*, switch arbiter is a logic block which chose which input ports should be connected to which output ports. In some case, when there is more than one request of input ports which demand to send data to the same output port, the switch arbiter is used to choses which input port is served in that time depend on its priority. It improves the connectivity between the input and output ports and used to determine the operation sequence of the routing paths so as to solve the problem of conflicting requests. The operation of switch arbiter is similar with the operation of VC arbiter, it means switch arbiter can be configured in two mode in order to change priority of each input. The switch arbiter architecture is shown in Figure 2.7. The description of switch signal is same as VC arbiter. However, in each router, there are five input ports so the request signal and grant signal are set to 5 bits. In the special router, when a port is chosen to forward the data, the input port of this port must not send the request to this output because the packet cannot transfer reversed in the port which it come on. There are just four requests from four other input ports.

SIMULATION AND EVALUATION

## Simulation script



Figure 3.1: A 8×8-router NoC evaluation platform.

In order to evaluate the operation and the performance of proposed router, we use the HDL-based simulator. Firstly, we implement reconfigurable fabrics and we simulated the general proposed router by composing each module. The proposed router had been built 2-D mesh 8×8 NoC (Figure 3.1). We also developed a Network Interface (NI) with a built-in dummy IP core as shown in Figure 3.2. The NI can be configured as either Master, Slaver or both.

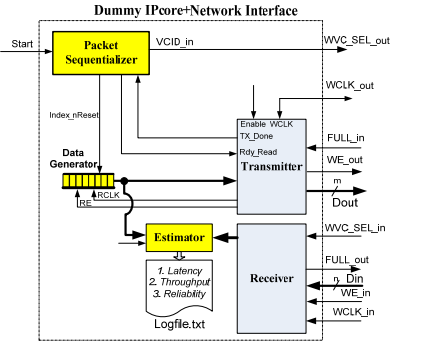


Figure 3.2: A NI with dummy IP core.

For evaluation, one or more couples of NIs will be mounted to the NoC at source router and destination router. At source router, NI will act as a Master (e.g. M1[0][0], M2[0][1], M3[0][2] in Figure 3.1) that will inject packets into NoC at a certain rate. The size of packet and the maximum packet injection rate (PIR) is controllable by setting the corresponding parameters of the *Packet Sequentializer*. Data for packets will be generated randomly by the *Data Generator*. At the destination router, a similar NI is also attached to the NoC but configured as a Slaver (e.g. S1[7][7], S2[7][2], S[2][7] in Figure 3.1).

Estimator in the Slaver will get packets from the NoC via the *Receiver* and compare them with those generated by the *Data Generator* in terms of correctness and delay. After that, Estimator calculates and exports network performances including latency and throughput to a *logfile.txt*. Here, the latency is defined as the time (in cycles) taken for one packet to travel from a source node to a destination node (from M1 to S1) while the throughput is the network’s transfer rate and is evaluated as a number of FLIT per cycle. In implementation, we defined a script in order to evaluate the performance including latency and throughput of proposed reconfigurable router and then we compared with performance of generic router to declare the efficient with using reconfigurable router. In the implementation process, we can change three parameters: Message size (PACKET\_NUM), Number of flits in one packet (PACKET\_SIZE) and packet injection Period (PIR). The packet size is set to ten 35-bit FLITs. The buffer size of each input port is equal to sixteen 35-bit FLITs, therefore, the size of each virtual channel is 4 FLITs in normal operation. Because of the conflict detection mechanism of the DFC unit, the packet injection rate can be self-adjusted depending on the status of the NoC. The script includes three scenarios as follows:

* Scenario 1: Just M1 transmits data to S1 at rate of 0.01 packets per cycle. The message size (PACKET\_NUM) is set to 25 packets.
* Scenario 2: In addition to the settings for Scenario 1, M2 is also enable to transmit data to S2 at rate of 0.0067 packets per cycle. The message size is set to 50 packets.
* Scenario 3: In addition to the settings for Scenario 2, M3 is also enable to transmit data to S3 at maximum rate of 0.005 packets per cycle. The message size is set to 25 packets.

The highest priority is assigned to the packet sent from M3 and the lowest priority is assigned to the packet from the M1.

## Simulation result

### Evaluation of reconfigurable fabrics

1. **Reconfigurable VC buffer**

From the Figure 3.3, each virtual channel has own size and base address, can be configured in run time. There are two asynchronous clocks for writing and reading data. In each time, If the signal “write\_en” is equal ‘1’ and the signal “full” of this VC is equal ‘1’ contemporary, it means the data can be written in the VCs. Similarly, if the signal “read\_en” is equal ‘1’ and the signal “empty” of VC is equal ‘1’, the data can be read out the VCs follow the written order. The writing in and reading out data can be processed in parallel in two scenarios: In the same FIFO channel or in different FIFO channels.

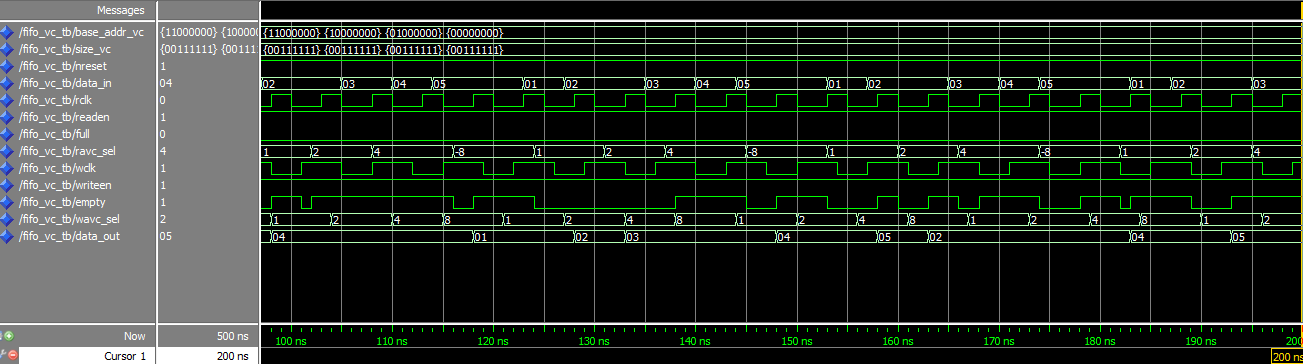
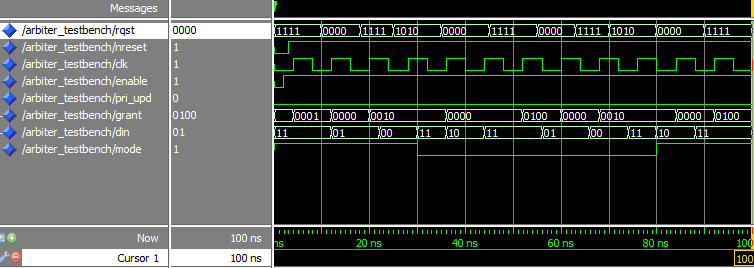
The signal “RAVC\_SEL” and “WAVC\_SEL” present which VC is chosen to read and write, respectively, in this time.

Figure 3.3: VC\_buffer simulation.

1. **Route Computation Unit**

The simulation result of Look-ahead Routing is shown in Figure 3.4. The next router address which is calculated by cPortID and source address, compared with the destination router address (x,y) to find nportID. PortID is labeled for each port Local: 000, North: 100, East: 001, South: 110, West: 011. The signal “cPortID” is reconfigured and given out when signal “CF\_RQST” is active. The signal “VCID” presents which VC of next input port used to store data. When calculating of next portID is finished, “RC\_done” signal is set to 1 to announce the finish.

1. **VC arbiter**

****

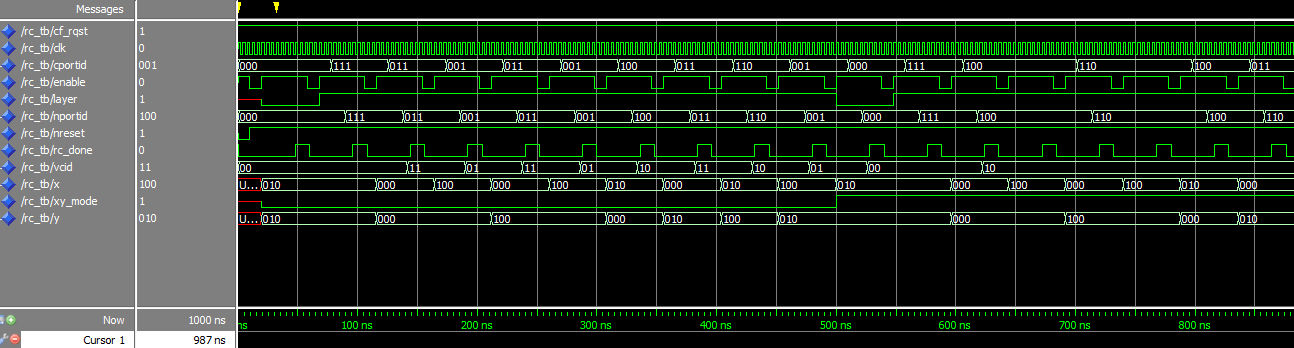
****

Figure 3.4: Look ahead routing algorithm.

Figure 3.5: VC arbiter simulation.

In VC arbiter, signal “PRI\_UPD” is control signal to allow upgrade priority level of VCs. Signal “gclk” is clock of a counter which has response to execute round robin mode, depend on PRI\_UPD or Grant. In this case, we proposed the value PRI\_UPD always equal ‘0’. In the rising edge of “gclock”, the value of the “dout” of the counter will be changed to perform the priority of VC. The mode signal will determine which mode of arbiter is used, round-robin or priority mode. IF mode = ‘1’ means round-robin arbiter will be executed then signal “count” increase 1 unit in each rising edge of gclock. If mode = ‘0’ means priority arbiter is active and signal “count” equal input “Din” in each rising edge of gclock.

1. **Switch arbiter**

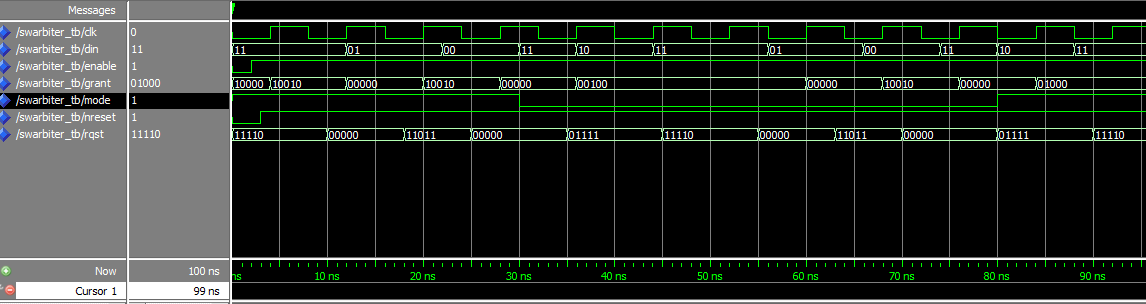


Figure 3.6: Switch arbiter Simulation.

In each router, there are five ports and due to routing algorithm, we define which output port the packet is moved through. Therefore, switch arbiter are response to arbitrate the requests from 4 input ports to an output port. Similarly with VC\_arbiter, the mode operation depend on MODE signal, however in switch arbiter, request signal is 5 bit represent 5 ports, in each specific router, the bit represent output is keep stable and chose one request from other input ports.

### NoC Evaluation

The comparison in respect of throughput and latency between the generic router and the reconfigurable multi-mode router is presented in Table 3.1 and Table 3.2. In this simulation script, we proposed Scenario 1, Scenario 2 and Scenario 3. These values of M1-S1, M2-S2, M3-S3 communication are measured at the target nodes S1, S2, S3, respectively.

In the description of Scenario 1 with only one M1-S1 communication, it is distinct no conflict between source and destination router, so there are no difference between generic router and reconfigurable router. In Scenario 2, M1-S1 communication has to compete with M2-S2 communication and because the priority of M2-S2 communication is higher than M1-S1 communication, the performance of M1-S1 is decreased and the congestion happens. However, there are significant difference between reconfigurable router and generic router. In reconfigurable router, the throughput and latency in M2-S2 communication is improved to 22.81% and 22.62%, respectively, when compared with generic router. It shows the effect of reconfigurable router in resolving contention.

Table 3.1: Comparison of Throughput between generic router and reconfigurable router.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario | Average throughput (FLITs/cycle) | | | | | |
| Generic router | | | Reconfigurable router | | |
| S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 0.08194 | - | - | 0.08194 | - | - |
| 2 | 0.041315 | **0.041315** | - | 0.041315 | **0.05074** | **-** |
| 3 | 0.03577 | **0.03577** | 0.04932 | 0.03577 | **0.04633** | 0.04932 |

Table 3.2: Comparison of Latency between generic router and reconfigurable router.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario | Average latency (cycles) | | | | | |
| Generic router | | | Reconfigurable router | | |
| S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 300.76 | - | - | 300.76 | - | - |
| 2 | 472.6 | **326.6** | - | 472.6 | **252.7** | **-** |
| 3 | 566.8 | **364.1** | 245.76 | 566.6 | **271.44** | 245.76 |

In Scenario 3, M1-S1 communication also has to compete with M3-S3 communication. Theoretically, there is no conflict between the M2-S2 communication and M3-S3 communication. However the appearance of M3-S3 communication affect the performance of not only M1-S1 communication but also M2-S2 communication. Because M1-S1 packet is blocked by a M3-M3 packet due to the higher priority of M3-S3 packet. After that, a demand of M2-S2 communication appears. Consequently, although M2-S2 packet’s priority has higher than M1-S1 packet, M2-S2 transmission cannot be done because the resource occupied by M1-S1 has not been released. This problem is resolved in the case of reconfigurable router. By reconfiguring router to the M1-S1 communication, the M2-S2 communication is performed in parallel with the M3-S3 communication. As a result, the reconfigurable router causes the throughput and latency of the M2-S2 communication to be improved about 29.52% and 25.44%, respectively, when there is congestion. From results of Scenario 2 and Scenario 3, it is clear to see that the efficient of reconfigurable router in resolving congestion to reform better performance of NoC.

## Synthesis

The RTL model of the proposed reconfigurable fabrics and the entire router has been synthesized by using Xilinx Vivado Design Suite. The synthesis result of the proposed router using Xilinx Virtex-7 xc7vx485 device is shown in Table 3.3, Table 3.4, Table 3.5, and Table 3.6.

Table 3.3: VC buffer resource by using Xilinx Virtex-7 technology.

|  |  |  |
| --- | --- | --- |
| **Resource** | **Utilization** | **Utilization ratio (%)** |
| Flip Flops | 321 | 0.05 |
| LUT | 719 | 0.24 |
| BRAM | 0.5 | 0.05 |

Table 3.4: RC resource by using Xilinx Virtex-7 technology.

|  |  |  |
| --- | --- | --- |
| **Resource** | **Utilization** | **Utilization ratio (%)** |
| Flip Flops | 6 | 0.01 |
| LUT | 17 | 0.01 |
| BRAM | 1 | 3.12 |

Table 3.5: Switch arbiter and VC arbiter resource by using Xilinx Virtex-7 FPGA.

|  |  |  |
| --- | --- | --- |
| **Resource** | **Utilization** | **Utilization ratio (%)** |
| Flip Flops | 6 | 0.01 |
| LUT | 11 | 0.01 |
| BUFG | 1 | 3.12 |

Table 3.6: Comparsion resource by using Xilinx Virtex-7 technology between generic and reconfigurable router.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resource Type** | **Used resource** | | **Area overhead (%)** | **Used/Available Ratio (%)** |
| **Generic router** | **Reconfigurable router** |
| Flip-Flops | 1716 | 1928 | +12.35 | 0.32 |
| LUTs | 3055 | 3518 | +15.15 | 1.16 |
| Memory LUTs | 175 | 199 | +13.7 | 0.15 |
| Power(W) | 0.245 | 0.246 | +0.4 |  |

The synthesis resource of reconfigurable router is shown in Table 3.6 and compared with generic router. The proposed reconfigurable router takes about 0.32%, 1.16% and 0.15% of the xc7vx485 in term of Flip-Flop, LUT and memory LUT, respectively. Compared with the generic router which use wormhole switching technique, the reconfigurable router which support both wormhole switching and virtual-cut-through techniques, the resource utilization overhead of reconfigurable router is about 12.35%, 15.15% and 13.7%. The power consumption is measured about 0.246W, increased 0.4% with generic router.

CONCLUSION

In this thesis, I proposed and implemented reconfigurable fabrics of a reconfigurable router for the Network-on-Chip. The router has been modeled at Register Transfer Level by using VHDL language and then synthesized on Xilinx Virtex-7 FPGA technology. A NoC platform includes 8×8 of the proposed routers and Network Interfaces (NIs) with a built-in dummy IP core are also developed for simulating and evaluating the router performance in terms of throughput and latency. The experiment results is shown and compared with the result in generic router, in order to present reconfigurable router is reliability and reduce the latency and increase the throughput when the congestion happening. In terms of implementation cost, our router consumes an insignificant ratio of the Virtex-7 XC7VX485 FPGA chip’s resources compared with generic router. The proposed router is feasible to apply for high-flexibility and high-performance on-chip embedded systems. In the future, we will continue optimizing the proposed router and then synthesizing and evaluating it on CMOS technology.

REFERENCES

1. Christophe Bobda, “*Introduction to Reconfigurable Computing – Architectures, Algorithms, and Applications*”. Springer, 2007.
2. Natalie Enright and Li-Shiuan Peh “*On-Chip Networks*”, 2009. A Publication in the Morgan & Claypool Publishers series: Synthesis Lectures on Computer Architecture.
3. Hung K. Nguyen, Xuan-Tu Tran, Vnu Key Laboratory for Smart Integrated System (SISLAB), University of Engineering and Technology “*Design and implementation of a Hybrid switching router for the Reconfigurable Network-on-Chip*”. Supported by Vietnam National University, Ha Noi under Project NO.QG.16.33.
4. T.J Todman, G.A. Constantinides, S.J.E. Wilton, O. Mencer, W. Luk and P.Y.K. Cheung “*Reconfigurable computing: architecture and design methods*” IEE Proc.-Comput. Digit. Tech., Vol. 152, No. 2, March 2005.
5. Érika. Cota, Alexandre de Morais Amory “*Reliability, Availability and Servicebility of Networks-on-Chip*” DOI 10.1007/978-1-4614-0791-1\_2, LLC 2012.
6. Nedera Najib, Alireza Monemi, Muhammad Nadzir marsono. “*Partially Adaptive Look-Ahead Routing for Low latency Network-on-Chip*”. University Teknologi Malaysia, 978-1-4799-6428-4/14/2014. IEEE.
7. Atagoziyev, Maksat, M.Sc., Department of Electrical and Electronics Engineering “*Routing algorithms for on chip Networks*”, December 2012.
8. Daniel U. Becker “*Efficient microarchitecture for Network-on-Chip routers*”, August 2012. Re-distributed by Stanford University.
9. I. Pratomo and S. Pillement, “*Impact of design parameters on performance of adaptive network-on-chips*”, International Conference on High Performance Computing and Simulation, pp 724-725, July 2012.
10. Modarressi, M, Sarbazi-Azad, H.; Arjomand, M. “*A hybrid packet-circuit switched on-chip network based on SDM*” Conference & Exhibition on Design (DSD), pp21-28.
11. Ruchika Chandravanshi, Viek Tiwari “*Network on Chip Router architecture performance analysis by using VHDL*”. International journal of Computer applications (0975-8887) Volume 140-No.13, April 2016.