HIGH PERFORMANCE AND ENERGY EFFICIENCY IN NETWORK ON CHIP (NOC) DESIGN

by

Pengzhan Yan
June 2018

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, State University of New York
in partial fulfilment of the requirements for the
degree of

Doctor of Philosophy

Department of Computer Science and Engineering
Acknowledgments

In the past five years, I really enjoyed my life in Buffalo and the great experience with many wonderful people.

I was not quite sure about my career path when I just received my Master’s degree five years ago. I am deeply indebted to my advisor, Dr. Ramalingam Sridhar, for bringing me into the Ph.D. program at that time, and for making my Ph.D. such a rewarding experience. Here, I can only thank him for a subset of the tremendous amount that he has given me. I would like to thank him for the independent research environment that he provided for me. Being an independent researcher and engineer is the most valuable lesson that I have learned in my Ph.D. study. Furthermore, his meticulous attitude toward research has impacted my work significantly. Finally, I am grateful for his deep and broad knowledge that have stimulated many insights on my work, as well as his continuous and unconditional support for my career development.

I also like to express my deep gratitude to Dr. Shambhu Upadhyaya and Dr. Wenyao Xu who serves as my Ph.D. dissertation committee members. I am thankful for the time they spent on discussing and commenting on my dissertation work. Their valuable feedback and insights have significantly improved the dissertation.

My appreciation also goes to my fellow Ph.D. students and friends in UB VLSI Lab. Special thanks go to Shixiong Jiang, James Clay and Na Gong for the discussion, feedback. Thanks to Na Going for helping get started in my Ph.D research. I really appreciate Shixiong Jiang’s help in both my research and personal life. He is a big brother to me. I am also thankful to James for helping me review the paper.

I appreciate our grind and work at nights and weekends in Davis 340. Our technical discussion have enriched my research from a diverse knowledge background. I also enjoyed the lunch, dinner, and all kinds of fun we had in the past four years.

Last but not least, I would like to thank my family for their consistent and unconditional support. My girlfriend April has been supporting all the time in my Ph.D. career. Thank her for making me a better and happier person every
day. I want to give my thankful to my two sisters Qing Yan and Li Yan. I cannot achieve this without their encouragement and unconditional love. My parents’s understanding for my personal and professional life have been instrumental to what I have accomplished.
# Table of Contents

 Acknowledgments ....................................................... ii  
 List of Tables ......................................................... vii  
 List of Figures ......................................................... viii  
 Abstract ........................................................................ x  

Chapter 1  
 Introduction ..................................................................... 1  
 1.1 Network-on-Chip ......................................................... 1  
 1.1.1 NoC Topology ....................................................... 2  
 1.1.2 Routing Algorithm ............................................... 4  
 1.1.3 NoC Infrastructure ............................................... 5  
 1.2 Router Microarchitecture ............................................. 5  
 1.2.1 Packets and Flits .................................................. 6  
 1.2.2 Router Block Diagram ......................................... 6  
 1.3 Dissertation Contributions .......................................... 7  
 1.4 Dissertation Organization ........................................... 8  

Chapter 2  
 Background ...................................................................... 9  
 2.1 Research Trends in NoC .............................................. 9  
 2.1.1 NoC Topology ..................................................... 9  
 2.1.2 Router Architecture .......................................... 10  
 2.1.3 NoC Routing Algorithm ..................................... 12  
 2.1.4 NoC Reconfiguration .......................................... 13  

Chapter 3  
 Router: Virtual Channel Allocation .................................... 15  
 3.1 Introduction ................................................................. 15
Chapter 5
Routing Algorithm 51
  5.1 Introduction .............................................. 51
  5.2 Indirect Adaptive Routing .............................. 54
      5.2.1 Globally Adaptive Routing ......................... 54
      5.2.2 Predictive Passing Time Routing ............... 59
  5.3 Evaluation ................................................. 61
      5.3.1 Experimental Setup .............................. 61
      5.3.2 Uniform Random Traffic .......................... 63
      5.3.3 Worst Case Traffic ............................ 63
      5.3.4 Transient Traffic Response .................... 64
          5.3.4.1 Uniform Random to Worse Case ........... 65
          5.3.4.2 Worst Case to Uniform Random .......... 65
  5.4 Conclusion ................................................. 65

Chapter 6
Fault Tolerant Router Design 68
  6.1 Introduction .............................................. 68
  6.2 Fault Tolerant Router Architecture ................. 70
  6.3 Fault Tolerant XY-Routing Algorithm ............... 72
  6.4 Performance Analysis .................................... 74
      6.4.1 Reliability Analysis ............................ 75
      6.4.2 Mean Time to Failure Analysis ............... 76
  6.5 Power Analysis on Application Reconfiguration .... 78
      6.5.1 Power Modeling ................................... 78
      6.5.2 Faulty Router Connection Topology .......... 79
      6.5.3 Power Consumption ............................ 80
      6.5.4 Area Overhead .................................. 81
  6.6 Conclusion ................................................. 82

Chapter 7
Conclusion 83
  7.1 Dissertation Summary .................................... 83
  7.2 Future Work ............................................. 85

Bibliography 87
# List of Tables

3.1 Theoretical VA Matching Quality Analysis ........................................ 31
3.2 VA Matching Quality Analysis ......................................................... 33
3.3 SA Matching Quality Analysis ......................................................... 33
3.4 Area ............................................................................................ 37
3.5 Power Delay Product ..................................................................... 37
3.6 Time ............................................................................................. 38

4.1 SA Matching Quality Analysis ......................................................... 47
4.2 SA Arbiter Characteristic Analysis .................................................. 48

6.1 Power Consumption with Spare PEs ............................................... 80
6.2 Power Consumption with Limited PEs ............................................ 81
6.3 Area Overhead Compared Against the Baseline Router ................. 82
## List of Figures

1.1 Hardware Solution for Future Computing .................................... 2  
1.2 Traditional NoC Topology ..................................................... 3  
1.3 High Radix Router NoC Topology ............................................. 3  
1.4 Baseline Network-on-Chip .................................................... 6  
1.5 Packets and Flits Structure .................................................. 7  
2.1 Generic NoC Synthesis Flow[1] .............................................. 10  
2.2 Router Power Consumption [2] ............................................... 11  
3.1 Canonical router architecture proposed in Dally’s Model [3] .......... 18  
3.2 Virtual Channel Allocation .................................................. 20  
3.3 Fixed-Priority Arbiter ......................................................... 21  
3.4 Round-Robin Arbiter .......................................................... 22  
3.5 CMPA Arbiter ................................................................. 23  
3.7 Separable output-first VC Allocation [4] ................................ 26  
3.8 Centralized Priority Management Allocation ............................ 28  
3.9 Area Comparison ............................................................... 34  
3.10 CMPA Layout ................................................................. 35  
3.11 Power-Delay Product Comparison ........................................ 36  
3.12 Timing Comparison ............................................................ 36  
4.1 Round-Robin Arbiter ........................................................... 41  
4.2 Switch Allocation Arbiter ..................................................... 41  
4.3 Switch Allocation ............................................................... 42  
4.4 Priority Output Ports Rotation: In the first clock cycle the priority  
output ports inside each arbiter will be P1,P2,P3...Pp. In the next  
cycle it will be Pp,P1,P2...P(p-1). In the third cycle it will be P(p-1),P(p),P1...P(p-2).  
4.5 Virtual Channel number:2 ..................................................... 45  
4.6 Virtual Channel number:4 ..................................................... 45  
4.7 Virtual Channel number:8 ..................................................... 46  
4.8 Throughput Gain at Saturation Injection Rate .......................... 47
4.9 Power consumed by router buffers, arbiter, crossbars, links, and clock [2] ........................................ 49
4.10 Power Consumption ....................................................... 50

5.1 Comparison of the average number of hops (uniform traffic) between different network topology [5] ....................... 52
5.2 Dragonfly Topology is an all connected hierarchical network: all groups are connected with each other and all routers in the same group also get connected with each other. ...................... 53
5.3 Credit Flow Control: router decreases CR when it sends out a flit and increases CR when it receives a credit .................. 55
5.4 Source Router cannot get the global queue information directly from its CR ......................................................... 55
5.5 Predictive Passing Time. R0: source router, R1 is an intermediate router in the source group, R2 is an intermediate router in destination group ......................................................... 57
5.6 Credit Delay Hardware Implementation .................................. 58
5.7 Latency under uniform random traffic pattern ......................... 62
5.8 Latency under worst case traffic pattern ................................. 64
5.9 UR to WC traffic pattern change at 0.45 injection rate ............... 66
5.10 WC to UR traffic pattern change at 0.45 injection rate ............. 66

6.1 Homogeneous 2D NoC Architecture ...................................... 71
6.2 Comparison of Baseline Router and Fault Tolerant Router ........ 72
6.3 Proposed router structure and its application in NoC ................. 73
6.4 Case 1: Source router is the faulty router. Case 2: Destination router is the faulty router. Case 3: Faulty router is in the path or not in the path ......................................................... 74
6.5 Reliability and Gain Analysis of Various Size of Mesh from 1 10 Years ................................................................. 77
6.6 MTTF Comparison .......................................................... 78
6.7 Faulty Network Topology .................................................. 79
6.8 Architecture of MPEG4 and VOPD ....................................... 80
Abstract

With Moore’s law fading, presence of billions of transistors on a single chip, and diminishing performance from uniprocessor architectures, multicore chips are emerging as the prevailing architecture in both application-specific and general-purpose markets. As the core count increases, the need for a scalable on-chip communication fabric that can deliver high bandwidth continues to gain importance, leading recently to multicore chips interconnected with on-chip networks. Networks-on-Chip (NoC) is widely regarded as a promising approach for addressing communication challenges affiliated with Chip Multi-Processors (CMPs) in the face of further increases in integration density. However, throughput, energy efficiency and routing algorithms become more challenging in NoC design.

To address throughput and energy efficiency, we evaluate Virtual-channel Allocation (VA), Switch Allocation (SA) in terms of matching quality, delay, area and power using RTL implementation. Based on the results of this study, we propose centralized priority management allocation (CPMA) router architecture to improve matching quality, delay, area, and energy efficiency. By coordinating arbiters’ priority in the first allocation stage, CPMA increases the matching quality. The centralized priority unit can also reduce each arbiters design complexity. The improvement of matching quality and reduction of design complexity make the router more energy efficient with less area consumption.

The dissertation next focuses on NoC routing algorithms. A parameter that can indicate the traffic congestion is needed for the routing algorithm to select the best path. In this dissertation, we propose predictive passing time as a parameter for the routing algorithm. Instead of indicating the traffic congestion, predictive passing time reflects the time it will take the message to go through the path. This parameter makes the routing algorithm more accurate in choosing among different paths. In NoC, a broken router can isolate a functional processing element (PE) from other nodes, severely restricting the performance of the system. To solve this issue, we present a new routing algorithm that can
transmit the message to the isolated PE even when the router is broken.

The approaches used here in Network on Chips can be adapted to Chip Multiprocessor and multicore systems.
Chapter 1

Introduction

1.1 Network-on-Chip

The continued scaling of transistor size has led to dramatic increase in the number of transistors on a single silicon die. More and more processor cores and large reusable components have been integrated on a single chip. Traditional buses and point-to-point global interconnections based computer architecture have reached their bottleneck to use the resources efficiently. New architectures are needed to use the large computation power efficiently. The era of big data and cloud computing also trigger the needs for a more powerful computing system. Networks-on-Chip (NoC) is proposed to replace global interconnections to alleviate this problem and considered as a next generation computer architecture.

Many NoC research had made significant achievement in the past ten years. Some research labs also tapeout the NoC chip based on their design. In 2008, Intel builds its 80-core processor prototype in TeraScale Research program [6]. The goal for the 80-core processor was to explore tiled architectures, 2D on-die meshes, and other issues about the circuits used on the chip. In 2010, the
same research group built the 48-core SCC processor to explore a scalable many-core architecture that does not use a cache-coherent shared address space [7]. In 2012, MIT also presented their chip prototype of a 16-node 4x4 mesh NoC fabricated in 45nm SOI CMOS that aims to simultaneously optimize energy-latency-throughput for unicasts, multicasts and broadcasts[8].

Research in NoC can be categorized into various domains such as NoC Topology, Router Architecture, Routing Algorithm, Switching Techniques, Flow Control and Buffer Implementation [9, 10, 11, 12, 13, 14, 15, 16, 17].

1.1.1 NoC Topology

When NoC infrastructure was first proposed, 2D mesh and torus-based network topologies were wildly used because of the smooth implementation and low radix on-chip router limitation. As the size of a network grows larger, low radix network topology requires messages to potentially go through many hops, which severely reduces performance and can increase the power consumption. To meet the throughput and latency requirement, researchers proposed high
radix router. The high radix routers were designed for large-scale NoC [18, 19]. New topologies using high radix routers like Flattened Butterfly, Dragonfly, and Hypercube, are then proposed to increase the scalability and reduce the cost.
1.1.2 Routing Algorithm

In NoC, routing algorithm determines how the data transfer between different NoC components. Ordinarily, different topology adopts the best routing algorithm that can handle their traffic pattern. In the simplest case, XY-routing algorithm can achieve high efficiency for a small 2-D mesh based topology. For large 2-D mesh based topology, dynamic XY-routing will have a better performance [20]. But for large Butterfly or Dragonfly topologies, XY-routing can severely reduce the throughput [21]. In most cases, an algorithm is only suitable for one or a few topologies. However, some critical problems exist in all the routing algorithms and need to be solved. Routing issue raises when the network begins to block the traffic. Deadlock and starvation are critical issues need to be resolved during the algorithm design.

1. **Deadlock.** When two or more resources need the other to be transmitted before itself, the routing can meet the deadlock issue. Routers do not release the resources before they get the new resources and so the routing is locked [22, 23, 24]. One can solve deadlock problems in many ways. For example, routing algorithm can allow the preemption of packets involved in a potential deadlock situation by rerouting or discard the resources.

2. **Livelock.** Livelock issue only occurs in some non-minimal routing algorithm or adaptive routing algorithm. The resources keep routing around its destination without reaching it. In some algorithms, transmitted resources try to avoid the heavy traffic path thus cannot reach the hot spot destination. This issue can severely reduce the transmit throughput. Researchers have proposed many solutions for livelock issue [25, 26, 27]. In these proposals, the resources that are detected and cannot reach the destination will either be discarded or assigned a higher priority during rout-
3. **Starvation.** The starvation in routing is very likely the thread starvation issue in the operating system. Routing algorithm may assign some important resources a higher priority and some other resources lower priority. If the resources are transmitted from the source, the resources will have a very low opportunity to reach their destination. The starvation can be avoided by using a strong fairness routing algorithm.

### 1.1.3 NoC Infrastructure

The NoC infrastructure includes three major components: Network Adapters, Router, and Links [15]. The functionality of links is very similar to the bus in traditional computer architecture design. It connects the nodes and routers to achieve the communication. The router receives resources from the links and forwards it to the core attached to it or another link based on the destination information stored in the resources [28]. Network Adapter (NA) makes the logical connection between the router and IP cores, since each IP may have different interface protocol with each other. The NA also make IP cores easier to transplant into other system-on-chip. Figure 1.4 shows a traditional 4-by-4 mesh based NoC architecture. Each IP core connects to a router via NA. And each router is also connected with four other routers through links.

### 1.2 Router Microarchitecture

One of the most important characters of NoC is the communication property, and router implements the communication protocol. In NoC design, the performance of the system depends critically on the performance of the router.
1.2.1 Packets and Flits

In NoC, resources are transmitted as messages. A message is broken into multiple packets (each packet has header information that allows the receiver to reconstruct the original message). A packet itself can be divided into many flits as shown in Fig. 1.5. Flit is the minimal unit transmitted in NoC. The packaging and un-packaging are done by Network Adapter. A packet is composed of different types of flits. The head flit is the first part of the packet. It contains the packet’s destination information. In router’s flow control, some arbitration and computation are only done for the head flit. The body flit is an intermediate chunk of a packet and contains most of the message. The tail flit is used to indicate the final stage of the current packet. The router can use it to release some resources such as virtual channel.

1.2.2 Router Block Diagram

A router is composed of control logic, function units, switches, registers and that collectively implement the routing and flow control functions required to buffer and forward flits en route to their destinations. Many router architectures
have been proposed, and in this thesis, we mainly focus on the architecture of input-queued virtual-channel routers. The router architecture comprises four components [29]: (a) routing computation unit, (b) virtual channel allocation logic, (c) switch allocation logic, and (d) switch traversal. Routing computation unit is responsible for extracting the destination information for the head flits and compute the output port and return the info to the next stage. Virtual channel allocation accepts requests from all VCs and arbitrates among them to the corresponding output physical channel. Switch allocation schedules the access requests to the crossbar. Switch traversal is responsible for passing the flit to appropriate output channel.

1.3 Dissertation Contributions

This dissertation analyzes the design of NoC from topology level to circuit level. In particular, it makes the following contributions:

1. Based on Dragonfly topology, we proposed a new Indirect Adaptive Routing Algorithm: Predictive Passing Time Routing Algorithm. The proposed
algorithm shows a more accurate decision when choosing between minimal routing and non-minimal routing. The new algorithm can also increases the NoC saturation injection rate and shows a shorter response time when the traffic pattern fluctuates.

2. We developed a new Virtual Allocation and Switch Allocation mechanism for router architecture. In the new allocation mechanism, we proposed a specific arbiter for NoC. The new design shows a high matching quality in router’s allocation stage which can help to increase NoC’s throughput.

3. We proposed a new router design and revised XY routing algorithm for the Network on Chip fault tolerance. The newly designed router will automatically link its PE to the neighboring router when it fails.

1.4 Dissertation Organization

This dissertation is organized as follows:

Chapter 1 presents the introduction of NoC and Router architecture.
Chapter 2 shows the current research trend in NoC and the challenges.
In Chapter 3, we evaluate Virtual Allocation design and presents new allocation mechanism.
Chapter 4 presents two new allocation methods used in Switch Allocation.
Chapter 5 explores Predictive Passing Time Routing Algorithm in dragonfly topology.
In Chapter 6, we show a fault-tolerant router architecture in NoC design.
Chapter 2

Background

2.1 Research Trends in NoC

As a new communication architecture, many design choices about NoC need to be made. The topics range from application and platform, all the way down to the circuit-and layout-level consideration [1].

NoC topology, router architecture and routing algorithm are the essential part of NoC and researchers have proposed many new designs on these topics.

2.1.1 NoC Topology

The ability of the network to efficiently disseminate information depends on the underlying topology [1]. In fact, NoC topology profoundly determines the network latency, bandwidth, floor-plan, routing algorithm, etc. The choice of topology is mainly determined by the communication latency, network bandwidth, power consumption and area requirements. For the guaranteed service traffic, the application traffic patterns and average latency need to be analyzed. The average hop count is usually used as an indicator for the data transfer la-
2.1.2 Router Architecture

A router is composed of control logic, function units, switches, registers and that collectively implement the routing and flow control functions required to buffer and forward flits en route to their destinations. In router architecture, buffer organization plays an essential role. Figure 2.2 shows the router power.
consumption distribution using an event-driven NoC simulator [2]. Since buffer consumes most of the area and energy budget, a significant part of research focuses on buffer management. Besides power, buffer also consumes most of the area budget. To reduce the buffer power and area consumption, dynamic buffer allocation (DBA) mechanism is used in many research [32, 33]. In conventional design, the buffer size is fixed at a high level to maintain the system working. However, DBA can dynamically allocate buffer resources according to network conditions. For example, if one input port has four virtual channels and each virtual channel need a buffer size of eight. In the baseline design, 32 buffers are needed. However, not all the VC needs eight buffers all the time. DBA can dynamically allocate the buffer for each VC if it is needed. In router design, Virtual Channel Allocation and Switch Allocation are responsible for resources allocation and arbitration. Row-Column (RoCo) Decoupled Router [34, 35] is proposed to reduce the content probability. RoCo split the routing into two modules. One module is responsible for X-dimension, and another is for Y-dimension. In NoC design, five port router is used in most cases. One port is the
local port can connect to the processing element. The other four ports connect to the neighbor routers [32, 36, 37]. Asynchronous NoC router is also proposed in [38]. The new router design combines the low-latency input buffers using a circular FIFO with a different end-to-end credit-based virtual channel flow control for a replicated switch architecture. It achieved a 55% less area and 28% latency improvement compared with the synchronous router. In NoC router design, resources allocation directly determines the performance of a router. VC and SA allocate resources for the router. In this dissertation, we redesign SA and VA to make router work in a more efficient way.

2.1.3 NoC Routing Algorithm

In NoC design, each topology has its special requirements and the routing algorithm adopted is also different from each other. One of the main purposes of the routing algorithm is to keep the traffic balance. Uneven traffic can not only increase the latency but also can consume more power. In some designs, the traffic feature can be known at design time and make it possible to implement the routing deterministically. When the traffic pattern cannot be predicted, dynamic routing is often used. Also the routing issue like deadlock, livelock, starvation are more critical in dynamic routing [22, 23, 24]. In small mesh based topology, xy-routing can meet most requirements. However, in Large-scale on-chip interconnection networks like flattened butterfly, dragonfly and hypercube network, new routing algorithm are still needed. Kim and Balfour [39] presented non-minimal routing for dragonfly and flattened butterfly network. Ahn and Binkert [40] shown an adaptive routing algorithm DAL for Hypercube network.

As the size of semiconductor chip keeps scaling, the fault-tolerant design is becoming more and more critical. When CMOS approaches nanometer, the
signal integration noise and miss-connect during fabrication are making chip tape-out more and more difficult. The types of faults can be categorized as transient and permanent faults. Transient fault refers to fault introduced by temporary environmental conditions, like cosmic rays and electromagnetic interference. Permanent fault means irreversible physical change. To overcome the increasing error rate and reduce the verification cost, the fault-tolerant design is proposed and becoming more and more popular [41, 42]. In a higher level, fault-tolerant routing is an excellent choice for researchers.

Research in NoC fault-tolerant design mainly focuses on routing algorithm and router architecture. Ahmed and Abdallah [43] proposed the routing algorithm of how to routing around the broken device and keep the communication change at minimal. In architecture level, Liu and Ouyang’s research [44, 45] shows how to improve router stability in NoC communication instead of using additional hardware as spare for the broken router.

### 2.1.4 NoC Reconfiguration

Some devices such as smartphones run diverse applications with diverse usage. These applications often have different requirements for hardware and software. For different implementations, the same IP core may have different usages, and the traffic pattern may also change. Thus application reconfiguration is needed to optimize resource usage [46]. To address this issue, NoC platform should be able to analyze the application requirements and make the reconfiguration in a cost effective way. Also, the transition between configurations must follow the routing rules: deadlock free, livelock free and starvation free [47]. Reconfiguration can be done by optimizing parameters such as communication change or power consumption. Ahmad and Erdogan [48] presented
a mechanism to utilize the resources effectively by dynamically remapping NoC based on the communication requirements. They also proposed reconfigurable router design to effectively switch between wormhole switching and virtual cut-through to achieve throughput. The future pattern estimation and the effective reconfiguration are also challenging work to be done. Ababei and Kavasseri [49] proposed an efficient routing algorithm to speeding-up the computational runtime for pattern analysis and reconfiguration. As the fault-tolerant design is becoming more and more critical for NoC, reconfiguration can also be done to solve the broken component [11]. In this dissertation, we present a new fault-tolerant router design for NoC reconfiguration.
Chapter 3

Router: Virtual Channel Allocation

3.1 Introduction

With Moores law supplying billions of transistors, and uniprocessor architectures delivering diminishing performance, multicore chips are emerging as a prevailing architecture in both general-purpose and application-specific chip markets. As the core count increases, the need for a scalable on-chip communication fabric that can deliver high bandwidth continues to gain importance, leading recently to multicore chips interconnected with sophisticated on-chip networks. Networks-on-Chip (NoC) is widely regarded as a promising approach for addressing the communication challenges associated with future Chip Multi-Processors (CMPs) in the face of further increases in integration density. However, as the number of cores on a single chip continues to grow, NoC’s area, transmission latency, and power consumption also increase which consume many chip resources and severely limit the system performance [50]. NoC efficiency can be improved at different level such as routing topology, algorithm and router architecture. Many researchers have demonstrated their solutions to the these problems. For NoC topology, traditional routing topologies
such as 2-D grid, 2-D torus and Octagon [51, 52, 53] are not able to keep their efficiency with the core count increasing. With the development of optical signaling technology, channel data rate is increased to another level which enables the research in high radix networks and topologies. Flattened Butterfly, Dragonfly and HyperX topologies have been proposed to reduce cost and increase throughput of large-scale network [21, 40, 54, 55]. In most cases, topologies using high radix routers can reduce the Manhattan distance between IPs which efficiently cut transmission latency.

Routing algorithm also changes with NoC topology. The basic XY routing method [20] shows its efficiency and low implementation cost with topologies like 2-D grid and 2-D torus. But in the new topology with high radix router, traditional routing algorithms often fail in finding the best path. Various kinds of routing algorithm have been proposed for topology using high radix router. Dynamic XY-routing, MIN, Non-MIN, DBAR and Globally adaptive routing are used to reduce the routing traffic and reduce the transmission latency with the new topologies [12, 56, 57]. In most cases, each topology has its own best routing algorithm. The routing algorithm needs to choose the best path depending on the topology and traffic.

Router plays a vital role in NoC infrastructure. It not only consumes the most of NoC area budget, but also directly reduce the power consumption and NoC efficiency. The primary function of NoC router is to forward each flit that arrives on one of its inbound channels to an appropriately selected outbound channel. Many techniques are used to achieve this. Flow control technique, buffer management, switching control and fault-tolerant techniques can significantly improve the router performance [28, 58, 59, 60, 61]. In router architecture, arbitration addresses the problem of coordinating access to a single shared re-
source between multiple competing agents. Allocation extends the problem to situations where agents compete for multiple resources simultaneously [4, 62]. Separable input-first allocation technique (SIFA) decompose the allocation into two successive phase of arbitration. In the first arbitration stage, each virtual channel or input port selects a single output port or output virtual channel to request. In the second stage of arbitration, the resource chooses a winner among all incoming requests. In contrast, separable output-first allocation technique forwards all the requests to the required output port or output virtual channel. Then the arbitration is performed among all incoming requests and send grants back to the winning agent. In this case, multiple output port or output virtual channel may choose the same winning requests. The second stage of arbitration is then used to make sure only one winning request get the resource. In this chapter, we propose CPMA technique for router allocation. This technique can be applied to all separable allocators. In the first allocation stage, CPMA can help the arbitration winners have less request conflicts in the second allocation stage which can help increase allocation’s matching quality. The improvement of allocation matching quality also means the less transmit latency and efficient resource usage. A new feature in a design normally introduces area or energy trade-off. But the implementation of CPMA shows the area and energy are reduced at the same time.

3.2 Router Structure

Figure 3.1 illustrates the major components of a canonical router. The router has five input ports and five output ports, supporting $n$ Virtual Channels (VCs) per port. The router comprises of four components [29]: (a) Routing computation unit, (b) Virtual channel allocation logic, (c) Switch allocation logic, and (d)
3.2.1 Routing computation

The packet’s routing information stored in the head flit. RC unit is responsible for returning information about the output port and the candidate VCs according to the destination information stored in the head flit, and the routing algorithm adopted. It can also return multiple output ports or single output port based on whether the algorithm is adaptive or deterministic. The output VCs can also refer to the input VCs of the next router. RC unit is a “per-packet” operation: it performs only on the head flit.
3.2.2 Virtual Channel allocation

VA accepts requests from all VCs and arbitrates among them to the corresponding output physical channel. Normally, two stages are required to achieve this arbitration, as shown in Fig. 3.2. In the first stage, a total number of \( P_v \) v:1 arbiters are implemented. For each VC there is one v:1 arbiter to arbitrate among candidate output VCs that processed by the RC unit. This design ensures only one VC at a particular output channel is assigned to the input VC. The second stage has a total number of \( P_v \) Pv:1 arbiters. The Pv:1 arbiter is adapted to deal with the worse case where all input VCs request for the same output VC. So each output VC will only accept one request. VA is also a “per-packet” operation: it performs only on the head flit.

3.2.3 Switch allocation

The SA unit schedules the access requests to the crossbar in two stages. The first stage has a total number of \( P \) v:1 arbiters. Each input port has an arbiter to arbitrate among the VCs in the port. So each input port will only generate one VC request per cycle. The second stage has \( P \) P:1 arbiters. Each output port has a P:1 arbiter that can arbitrate among all the winning input port request. This can ensure that only one input port can be connected to one output port during a cycle. SA unit is a “per-flit” operation unlike RC and VA.. All the flits need to go through SA unit to get permission and schedule to go to the crossbar.

3.2.4 Switch traversal

Based on the VA and SA’s granted passage, switch traversal reconfigures the crossbar connection per cycle. Flits that have been granted passage on the cross-
bar are passed to the appropriate output channel.

3.3 Arbiter Design

The functionality of arbitration and allocation is mainly done through arbiters. Arbitrator or allocator is different from each other by how to manage the arbiters.
3.3.1 Fixed-Priority Arbiter

Arbiter is designed to select one or more resources among requests for limited resources [63, 64]. Its function is to prevent two requesters reserve the same resource when they should not. Arbiter is the fundamental unit inside an allocator and its arbitration mechanism directly decides the performance of allocation. Figure 3.3 shows a basic arbiter structure using a linear array of basic function processing cells $P$. Inside each cell, the output will be granted if both request input $r_i$ and incoming priority signal coming from the previous cell are asserted. Also, the priority signal is propagated to next cell if the current request is not granted. The fixed priority arbiter design reduces the implementation complexity. However, this design shows low fairness and its critical path delay scales with the number of inputs.

3.3.2 Round-Robin Arbiter

Round-robin arbiter is proposed to achieve strong fairness and reduce critical path latency [65, 66]. Figure 3.4 shows the round-robin arbiter structure. Compared with fixed priority arbiter, each function cell $P$ has an independent pri-
ority input $p_i$. The priority input is provided by function cell $R$ which stores the priority bits in the registers and rotate the registers during each clock cycle. Since a different priority bit is provided to arbiters, strong arbitration fairness can be achieved, and the critical path length is also shortened.

### 3.3.3 CPMA Arbiter

The CPMA arbiter is shown in Figure 3.5. Compared with Round-Robin arbiter, CPMA arbiter removes the rotated register priority function and keeps the simplicity. Compared with the Fixed-Priority arbiter, CPMA arbiter takes the priority signal from outside. If the priority bit is provided appropriately, CPMA arbiter can keep strong fairness.

### 3.4 VC Allocator

In Virtual Channel Allocation, flits in each virtual channel requests for their output port and candidate virtual channel. VA arbitrates among them to the corresponding output physical channel per flit. Typically, Virtual Allocation can
be categorized into two types: static allocation and dynamic allocation. Static allocation can make sure that requests from all directions have the equal opportunity for grants. However, Static Allocation cannot handle uneven traffic and is not used as much nowadays. If the traffics with unpredictable or burst property, Dynamic Allocation, on the other hand, can provide a more efficient arbitration. For most cases, Static Allocation has a straightforward design and Dynamic Allocation has a more complex model. Currently, more and more designs try to integrate features of both schemes: keep fairness and the ability to arbitrate uneven traffic.

### 3.4.1 VC Allocator Design

Virtual channel arbitration is decomposed into two phase. In the first phase, the arbiter in each input VC grants one output virtual channel. In the second phase, the output VC grant one request among $P^*V$ requests. For each phase, multi arbiters are used. The organization of the two phases is called the alloca-
Allocators are generally categorized as input-first allocator and output-first allocator. But for these two allocations, each phase do its arbitration and can lead to low efficiency. In the worst cases, only one flit will be granted among all requests. Consider all the input virtual channel for all input ports grant the request to the same output virtual channel. So, in the second phase, only one request will be granted which means at most one flit will be transported in this cycle. In some other cases, it may also be possible that the grants come from the first phase request for the same output virtual channel. Separable Input-first Allocation and Separable Output-first Allocation are currently most used allocation schemes.

### 3.4.2 Separable Input-first Allocation

Before VC allocation, the destination port is known by RC unit. For Separable Input-first Allocation, it can select among the V output VCs for each output port [4]. As shown in Fig. 3.6, a multiplexer expands the results from the first arbitration to a P*V wide vector of output side requests. But, the requested output VCs can also be used by the other packet. To avoid this, a multiplexer checks the availability of the requested VCs at the destination output port, and the results vector is then used to mask the candidate input VCs ahead of input-side arbitration.

In output-side arbitration, a requested output Virtual Channel is marked as busy if it receives requests from one or more input Virtual Channel. Then the vector results are sent back to the input VC arbiters. So, the input VC arbiter will know which output VCs are unavailable.
3.4.3 Separable Output-first Allocation

Figure 3.7 describes the Separable Output-first Allocation. In output-first allocation, the first arbitration stage is very likely in the input-first allocation. Each input Virtual Channel requests the destination output virtual-channels. However, different from input-first allocation, the arbitration among the output virtual channel is not done until output side selection. For the input side, the grants
Figure 3.7. Separable output-first VC Allocation [4]

from different output ports are combined as the input-first allocation. But, a specific input Virtual Channel can receive grants from different output Virtual Channels, in which an extra arbitration stage is needed to arbitrate among them. However, the disadvantage of output-side arbitration is obvious. Since the output Virtual Channels that are allocated must be marked as busy, the input side final grant signals must be transmitted back to the output port. This further transmission can severely reduce the delay in generating the input side grant
3.4.4 CPMA in Virtual Allocation

The design architecture can be seen in Fig. 3.8. The arbitration unit $P$ in the proposed design is similar to the round-robin arbiter. But CPMA shows a different way to provide the priority bits. In round-robin arbiter, the priority bit is provided by itself through the output feedback. Processing unit $R$ in Fig. 6.3b keeps the priority vector and provide it to each arbitration unit $P$. To keep strong arbitration fairness, the priority vector is rotated one bit during each cycle. However, this independent arbitration may reduce the allocation matching quality. For example, in the first arbitration stage, if all the input virtual-channel’s arbiters choose the same output port VC, then in the second arbitration stage, only one input to output virtual-channel will be matched. This is not only a waste of resources but also severely reduces router’s throughput.

In CPMA, instead of providing the priority bits by each independent arbiter, the centralized priority management unit (CPMU), as shown in Fig. 3.8, give the priority bits to all arbiters. When a new packet comes to Virtual Allocation, the destination output port information will be sent to CPMU directly. After receiving input VC’s requested output port, CPMU can assign a different output VC priority bit to input VC’s arbiters if these VCs request for the same output port. For example, if input VC1 and VC2 request for the same output port, the traditional design may allow VC1 and VC2 to request for the same output VC at the first arbitration stage. If both requests win, then only one of the two winners from the first arbitration can win at the second stage. In this way, only one pair of input VC and output VC will be matched. However, in CPMU, if two input VCs ask for the same output port, they will be assigned two different output VC signals.
priority bits at the first arbitration stage. If both of them win in the first stage, then both of them will be granted in the second stage. In this case, two pairs will be matched.

In router design, CPMA can be used in both VA and SA. To describe the design better, we assume the router has P ports and each port has V virtual-channels.

As shown in Fig. 3.2, in the first stage of VA, each arbiter selects one VC among the destination output port’s all available VCs. Since there are P*V arbiters in the first stage, it is highly possible that different input VCs who request for the same output port (INVCSO) be assigned with the same output VC. However, CPMU gives the INVCSO a different priority bit from each other. So the possibility for INVCSO to be assigned with same output VC is reduced. Then

Figure 3.8. Centralized Priority Management Allocation
Algorithm 1 CPMU Algorithm

procedure PRIORITY MANAGEMENT
    \( P \) = router’s port number
    \( V \) = port’s virtual-channel number
    \( \text{PriorityBits}[P \times V][V] \)
    for \( i = 0; i < P \times V - 1; i++ \) do
        \( P_{\text{req}} = \) requested output port
        if VC is available then
            for \( j=0; j < V-1; j++ \) do
                \( \text{PriorityBits}[i][j] = 1 \)
                mark the VC reserved
                break
            mark \( P_{\text{req}} \) as reserved
        else
            for \( j = 0; j < V-1; j++ \) do
                if VC is available then
                    \( \text{PriorityBits}[i][j] = 1 \)
                    mark the VC reserved
                    break
    in the second stage of arbitration, there is a higher possibility for INVCSO to get the resources. The same analysis logic can be used in SA. As shown in Fig. 4.3, in the first stage of SA, each input port needs to select one VC out of its \( V \) VCs. The selected VC then gets the chance to compete with winners from other ports in the second stage of SA. In traditional design, all the arbiters in the first stage are independent. So the winners from different input ports may ask for the same output port which will make competition in the second stage arbitration and reduce the matching possibility. However, CPMU can make sure that each input port arbiter has a different output port priority bit. Thus during each cycle, more flits can be transmitted through Switch Traversal.
3.4.5 Matching Quality Analysis

Matching quality is a normalized metric to assess the quality of all kinds of allocation [62]. It is obtained by dividing the total number of grants generated by each allocator implementation by the number of grants a maximum-size allocator would generate for the same sequence of requests. For this design, we build a probability model to calculate the matching quality. We can use this model in VA as an example.

\[ P_{RR}(i) = \left( \frac{1}{PV} \right)^i \left( \frac{PV - 1}{PV} \right)^{PV-i} \binom{PV}{i} \quad i \in PV \] (3.1)

\[ grants = \sum_{i=1}^{PV} (P_{RR}(i) \times 1) \] (3.2)

\[ P_{CPMA}(i) = \left( \frac{1}{P} \right)^i \left( \frac{P - 1}{P} \right)^{PV-i} \binom{PV}{i} \quad i \in PV \] (3.3)

\[ grants = \frac{\sum_{i=0}^{V} (P_{CPMA}(i) \times i) + \sum_{i=V+1}^{PV} (P_{CPMA}(i) \times V)}{V} \] (3.4)

Equations 3.1 and 3.2 are used to analyze the matching quality of SIFA allocator implemented with round-robin arbiter. In Equation 3.1, \( P_{RR}(i) \) is the probability for \( i \) number of input VC choose a specific output VC and other input VC choose the other output VCs. For each input VC, the probability for it choosing a specific output VC is \( \frac{1}{PV} \). The probability for it to choose the other VC is \( \frac{PV-1}{PV} \). The first item in equation 3.1 represents the possibility of \( i \) input VCs choose this specific output VC. The second item represents the remaining input VCs choose this specific output VC. The second item represents the remaining input VCs choose the other output VCs. The third item is the combination of all the input VCs. Equation 3.2 represents the average grant per each output VC.
Equation 3.3 and 3.4 are used to calculate allocator implemented with proposed mechanism. Different from allocator implemented with round-robin arbiter, the analysis is based on each output port grants. Similar to Equation 3.1, Equation 3.3 represents the probability for each output port to have \( i \) number of input VC requests. Equation 3.4 shows the advantage of our proposed design: For a specific output, if the number of input VC requests is less than \( V \), CPMA can make sure that all the requests are assigned to a different output VC. In this way, \( i \) requests can be granted per port if the requests are less than \( V \). If the number of input VC requests is larger than \( V \), then only \( V \) requests can be granted.

For a router with five ports, the VA matching quality result from our mathematic analysis as shown in Table 3.1.

<table>
<thead>
<tr>
<th>VC Num</th>
<th>Allocator(R)</th>
<th>Allocator(CPMA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC=4</td>
<td>0.6415</td>
<td>0.8254</td>
</tr>
<tr>
<td>VC=8</td>
<td>0.6368</td>
<td>0.8752</td>
</tr>
<tr>
<td>VC=16</td>
<td>0.6344</td>
<td>0.9113</td>
</tr>
</tbody>
</table>

As can be seen from the theoretical analysis result, the matching quality of allocator implemented with round-robin arbiter keeps stable with different VC number. The proposed design shows a higher matching quality when VC number equals to 4. As the VC number increases, the matching quality also increases. The result is expected, for Allocator(R), the arbitration is random and not related to number of VCs. In the proposed design, as the VC number increases, CPMU has more opportunity to distribute the requests to different output VC thus increase the matching quality.
3.5 Evaluation

3.5.1 Experimental Setup

To evaluate the efficiency of proposed design, we develop parameterized RTL implementations for different allocators. The RTL design is synthesized by Cadence Genus Synthesis solution using TSMC 45nm standard cell library. In particular, we considered three different implementations:

- Separable input-first allocation implemented with fixed priority arbiter (Allocator(F)).
- Separable input-first allocation implemented with round-robin arbiter (Allocator(R)).
- Centralized priority management allocation (Allocator(CPMA)).

To present the scaling behavior, we compare router implemented with four, eight and sixteen virtual-channels.

3.5.2 Matching Quality

To evaluate the matching quality, we created a set of 20000 pseudo-randomly generated requests for each virtual-channel and counted the clock cycle needed to grant all the requests. The matching quality was then compared to the number of requests each virtual-channel can grant in one cycle.
Table 3.2. VA Matching Quality Analysis

<table>
<thead>
<tr>
<th>VC Num</th>
<th>Allocator(R)</th>
<th>Allocator(CPMA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC=4</td>
<td>0.6410</td>
<td>0.8260</td>
</tr>
<tr>
<td>VC=8</td>
<td>0.6359</td>
<td>0.8757</td>
</tr>
<tr>
<td>VC=16</td>
<td>0.6348</td>
<td>0.9119</td>
</tr>
</tbody>
</table>

Table 3.3. SA Matching Quality Analysis

<table>
<thead>
<tr>
<th>VC Num</th>
<th>Allocator(R)</th>
<th>Allocator(CPMA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC=4</td>
<td>0.6791</td>
<td>0.7375</td>
</tr>
<tr>
<td>VC=8</td>
<td>0.6739</td>
<td>0.8545</td>
</tr>
<tr>
<td>VC=16</td>
<td>0.6769</td>
<td>0.9720</td>
</tr>
</tbody>
</table>

Table 3.2 shows that the simulation results have a good consistency with our theoretical results. As we can see from Table 3.2 and 3.3, both VA and SA shows significant improvement in matching quality using proposed design.
3.5.3 Area

Figure 3.9 shows an area comparison for the three allocators. Allocator(F) shows the minimum area due to its simple design. Usually, people don’t use this design if strong fairness is required. When VC number equals to four or eight, allocator(R) and allocator(CPMA) shows a similar area consumption. But when VC number rises to 16, allocator(CPMA) shows less area consumption. In allocator(R), each arbiter keeps its priority selection unit. So the area of priority selection unit grows linearly with the number of arbiters. However, in allocator(CPMA), the centralized priority management unit can have a better use of the shared resources and shows a slower area growth when the number of VC increases. The layout of 16 virtual channel CPMA is shown in Fig. 3.10.
3.5.4 Power-Delay Product

The Power delay product (PDP) shown in Fig 3.11 largely mirror the results of their area. Allocator(F) shows the minimal PDP for all configuration. For small configuration, Allocator(R) and Allocator(CPMA) show a similar PDP. Alloca-
tor(CPMA) become more efficient as the number of ports increases.

![Figure 3.11. Power-Delay Product Comparison](image)

3.5.5 Efficiency

![Figure 3.12. Timing Comparison](image)

The PDP comparison of different design cannot reflect the design performance. Since Allocator(CPMA) has a higher matching quality, the time/cycle it used to finish the same amount of job is less compared with Allocator(R).
To evaluate the design’s performance, we generated synthetic uniform traffic for each input port and measured the time needed to transmit all the packets. For each input port, we injected 2000 packets, and each packet contains four flits. As can be seen in Fig. 3.12, Our design shows a 24% productivity improvement.

### 3.5.6 Results

The results of the area, PDP, and time are shown in the tables below.

**Table 3.4. Area**

<table>
<thead>
<tr>
<th>VC Num</th>
<th>Allocator(R)</th>
<th>Allocator(R)</th>
<th>Allocator(CPMA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC=4</td>
<td>156um</td>
<td>197um</td>
<td>211um</td>
</tr>
<tr>
<td>VC=8</td>
<td>712um</td>
<td>813um</td>
<td>809um</td>
</tr>
<tr>
<td>VC=16</td>
<td>3833um</td>
<td>4767um</td>
<td>4172um</td>
</tr>
</tbody>
</table>

**Table 3.5. Power Delay Product**

<table>
<thead>
<tr>
<th>VC Num</th>
<th>Allocator(R)</th>
<th>Allocator(R)</th>
<th>Allocator(CPMA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC=4</td>
<td>0.892pj</td>
<td>1.136pj</td>
<td>1.495pj</td>
</tr>
<tr>
<td>VC=8</td>
<td>4.638pj</td>
<td>6.051pj</td>
<td>6.102pj</td>
</tr>
<tr>
<td>VC=16</td>
<td>25.563pj</td>
<td>33.732pj</td>
<td>30.574pj</td>
</tr>
</tbody>
</table>
### Table 3.6. Time

<table>
<thead>
<tr>
<th>VC Num</th>
<th>Allocator(R)</th>
<th>Allocator(R)</th>
<th>Allocator(CPMA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC=4</td>
<td>16493</td>
<td>16755</td>
<td>12923</td>
</tr>
<tr>
<td>VC=8</td>
<td>16428</td>
<td>16267</td>
<td>12658</td>
</tr>
<tr>
<td>VC=16</td>
<td>15863</td>
<td>15920</td>
<td>12211</td>
</tr>
</tbody>
</table>

#### 3.5.7 Conclusion

In this chapter, we presented a new allocation mechanism called centralized priority management allocation (CPMA). By coordinating all arbiters priority bit in allocators first stage, CPMA can help the winners from the first allocation stage have different output requests in the second stage. Thus the allocators matching quality can be increased. In NoC router design, CPMA can be used in both VA and SA. In the VA stage, CPMA can increase the packet matching quality. In SA stage, it can increase the flits matching quality. Our results show that the new design can increase performance and reduce the power consumption at the same time. Also, the area consumption is slightly changed based on the number of VCs in the router.
Chapter 4

Router: Switch Channel Allocation

4.1 Introduction

In this Chapter, we present a novel mechanism in the first stage of SA that can increase router’s throughput by adopting priority based dynamic arbiters using predefined round-robin algorithm. In this structure, SA’s first stage arbiters will have a higher opportunity to choose requests asking for different output ports. So in the second stage of the SA, the competition for the output port will be reduced, which means that more flits will travel through the crossbar in one cycle. The throughput thus can be increased due to the higher number of flits traversing the crossbar per cycle. To better evaluate the performance, we build a Verilog HDL based FPGA implementation.
4.2 Arbiter Design

4.2.1 Cooperation Priority Arbiter

In Round-Robin arbiter, the priority bit vector is provided by the feedback of previous grants as shown in Fig. 4.1. However, in the Cooperation Priority Arbiter (CPA), the priority bit vector is provided by Cooperation Priority Generator directly as shown in Fig. 4.2. Also, each arbiter’s Cooperation Priority Generator also works with other arbiter’s Cooperation Priority Generator using round-robin algorithm. In this way, we can make sure that each output port Switch Allocation arbiter will have a different priority bit. Also compared with CPMA arbiter in Chapter-2, Cooperation Priority Arbiter has a more straightforward implementation but can keep high match quality at the same time. The proposed Switch Allocation Arbiter can be regarded as a tradeoff design round-robin arbiter and CPMA arbiter.

4.3 Switch Allocator

4.3.1 Switch Allocator Design

The SA unit schedules the access requests to the crossbar. It also performs in two stages. The first stage has a total number of $P\cdot v:1$ arbiters as can be seen in Fig. 4.3. Each input port has an arbiter to arbitrate among the VCs in the port. So each input port will only generate one VC request per cycle. The second stage has $P\cdot P:1$ arbiters. Each output port has a P:1 arbiter that can arbitrate among all the winning input port requests which can make sure that only one input port can be connected to one output port during a cycle. Different from RC and VA, SA unit is a ”per-flit” operation. All the flits need to go through SA unit to get
permission and schedule to go to the crossbar.

![Round-Robin Arbiter](image1)

**Figure 4.1.** Round-Robin Arbiter

![Switch Allocation Arbiter](image2)

**Figure 4.2.** Switch Allocation Arbiter

### 4.3.2 Round-Robin Arbiter Inside the Switch Allocation

Baseline router adopts the round-robin arbiter as shown in Fig. 4.1. R module receives grants from the output port, and then it will rotate the received grants and reset them as the new priority setting. This algorithm can ensure fairness among requests. Figure 4.2 shows the predefined priority cooperation based round-robin arbiter. The difference is that in the new structure the R module will not receive and rotate the grants from the output ports. Instead, the initial
priority bits are predefined in all arbiters inside the first stage of switch allocation. The arbiters will set the priority bit following a specific sequence when the allocation stage arbitrates among requests. Each arbiter starts with a different priority bit, and all of them will follow the same sequence to reset the priority bits as can be seen in Fig. 4.4. This mechanism makes sure that all the arbiters will have a different priority bit during the operation. Also, the priority cooperation is predefined which means that the arbiters do not need to communicate with each other. Through the priority cooperation, grants from different arbiters will have a higher probability to be different from each other.
In the second stage of switch allocation, this diversity can reduce competition and increase the throughput. Based on this structure the throughput will have a nearly linear relationship with the number of virtual channels. However, it is not the most efficient way to reduce the competition in the second stage. A dynamic cooperation mechanism can be built between \( P \) arbiters to choose the best priority bit. However, dynamic cooperation will dramatically increase the complexity of the design and the time used to implement the algorithm will also increase the latency. Our model not only increases the throughput but also reduces the design complexity.

### 4.3.3 CPA Inside the Switch Allocation

In the first stage of SA, we have to arbitrate among \( V \) virtual channels for each port as can be seen in Fig. 4.3. The winning virtual channel will get the chance to compete with winners from other ports in the second stage of SA. Since all the arbiters in the first stage are independent, the problem will arise: the winners from different input ports may ask for the same output port which will make competition in the second stage arbitration and reduce the throughput. Let’s take the worst case as an example: inside each input port, different virtual channels request for different output ports. At input port 1, \( v \) virtual channels ask for \( v \) different output ports. Assume that same situation happens in other input ports. Inside all these input ports, if the virtual channels request for output port one win in the first arbitration stage, then the worst scenario will happen in the second stage. The arbiter in the second stage of output port one then have to choose one from \( p \) requests, and other output ports will have no requests which mean that only one flit will be transferred during the clock period. To solve this kind of problem, we come up with a predefined priority
Figure 4.4. Priority Output Ports Rotation: In the first clock cycle the priority output ports inside each arbiter will be P1,P2,P3...Pp. In the next cycle it will be Pp,P1,P2...P(p-1). In the third cycle it will be P(p-1),P(p),P1...P(p-2).

cooperation based round-robin arbiter. In the design, we implement the priority cooperation based round-robin arbiter in SA1. P different arbiters are assigned P different priority output ports, and the P different priority output ports will rotate between these P arbiters during the execution as can be seen in Fig. 4.4. This way the worse scenario can be avoided. Assuming that all virtual channels in each arbiter request for different output ports, then the arbiters will choose the virtual channels which request for the priority outputs. This is how arbiters cooperate with each other to select different output ports. In the second stage of switch allocation, each arbiter will only have one request which means that P flits will be sent out during one clock cycle. In this way, we can reduce competition in the second stage resulting in more flits getting access to the crossbar.
4.4 Evaluation

4.4.1 Experimental Setup

To evaluate the efficiency of proposed design, we implement a two clock-cycle pipelined wormhole router. In the first pipeline stage, look ahead route com-
putation, virtual channel allocation and switch allocation are achieved. In the second pipeline stage, we perform the crossbar traversal. To better measure the throughput, we build a 3x4 regular mesh based network. Each input port of a router has a four deep 32-bit wide buffer. The router is implemented using synthesizable Verilog HDL. The code was simulated using Modelsim and synthesized in Quartus II.

### 4.4.2 Throughput

In the simulation, we use uniform random traffic pattern to generate workload. As seen in Fig. 4.5 to Fig. 4.7, there is no significant improvement in the throughput when the virtual channel number is 2. However, when we increase the number of virtual channels, it shows a greater throughput increase. These results are expected, the larger the virtual channel number, the higher opportunity for the arbiters to choose the request for priority output port and the higher the throughput.
4.4.3 Throughput Gain

The throughput gain is illustrated in Fig. 4.8. In this figure, it shows a significant improvement in throughput gain when we change the virtual channel number from 2 to 4. There is also an improvement when we change the virtual channel number from 4 to 8, but at a slower rate. This is because our router is a five-port router, so when the VC number is around 5, then the throughput gain will increase with the VC number if the VC number is much larger than five the throughput gain will reach a saturation value. For our design, the throughput has an improvement up to 13% for the router with eight virtual channels.

4.4.4 Matching Quality

Table 4.1. SA Matching Quality Analysis

<table>
<thead>
<tr>
<th>VC Num</th>
<th>Allocator(R)</th>
<th>Allocator(CPA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC=4</td>
<td>0.6791</td>
<td>0.7023</td>
</tr>
<tr>
<td>VC=8</td>
<td>0.6739</td>
<td>0.7530</td>
</tr>
<tr>
<td>VC=16</td>
<td>0.6769</td>
<td>0.8680</td>
</tr>
</tbody>
</table>
Cooperation priority arbiter is a tradeoff between round-robin arbiter and CPMA arbiter. CPA has a more straightforward design compared with CPMA arbiter but shows a better matching quality than round-robin arbiter in switch allocation design. Table 4.1 shows the matching quality comparison between switch allocation implemented with round-robin arbiter and cooperation priority arbiter.

### 4.4.5 Area and Delay

<table>
<thead>
<tr>
<th>Structure</th>
<th>Area Consumption($\mu m^2$)</th>
<th>Worst Case Delay($ps$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round Robin Arbiter</td>
<td>77.9</td>
<td>739.6</td>
</tr>
<tr>
<td>CPA Arbiter</td>
<td>125</td>
<td>457.8</td>
</tr>
</tbody>
</table>

Table 4.2. SA Arbiter Characteristic Analysis

To better analyze the arbiter characteristic, we synthesized the design using Cadence Design Compiler with TSMC 45nm standard cell library. As can be seen from Table 4.2, compared with baseline arbiter, the proposed arbiter shows a 61% reduction in the worse case delay. Since switch allocation is on the critical path of the router, the router frequency can also increase. Also, the proposed arbiter presents an increase in the area consumption. However, the arbiter area can be ignored when it is compared with the router buffer.

### 4.4.6 Power Consumption

In any design, there is always a tradeoff between area budget, power consumption, and performance. Usually, high throughput design will come with higher power consumption. To achieve a 19% higher throughput a DSB200 router [37] consumes 35% more power and occupies 58% more area. For the router ar-
Figure 4.9. Power consumed by router buffers, arbiter, crossbars, links, and clock [2]

architecture in [67], the improvement of 20% on throughput costs a 52% – 100% power penalty. This is because most of these designs change the buffer allocation mechanism or the crossbar as a way to improve the throughput. However, these components are the major part of power and area consumption. As seen in Fig. 4.9, which is simulated by Ababei [2] by using power models in Orion 2.0 [68], the power consumed by arbiters plays a minimal role in the entire router. Using the simulator provided by Ababei [2], the power consumption comparison is shown in Fig. 4.10. As we can see from the Figure, there is only 0.16% change in the power consumption. Compared with other mechanisms which increase the throughput at the expense of power consumption, our design provides a better solution.
4.5 Conclusion

In this chapter, we presented a router architecture that can increase the NoC throughput with a minimal area and power overhead. In this new architecture, a predefined communication mechanism is implemented for arbiters in the first stage of switch allocation. This mechanism can efficiently reduce the competition in the second stage of switch allocation. We also observed that the advantage of proposed architecture would be better shown when the virtual channel number is higher than the number of output ports.
Chapter 5

Routing Algorithm

5.1 Introduction

Due to fixed packaging constraints, low radix router networks were widely used to offer lower packet latency [69]. For 2D, 3D mesh and torus network topologies with limited terminals [70], low radix routers are a good choice. However, when the size of a network grows larger, a low radix network topology requires messages to potentially go through many hops, which severely reduces performance and increase the power consumption. Today’s processors with more than 100 cores are already in the market and a processor with kilo-cores may soon become reality [71]. High radix router and new topologies are needed in future multi-core design.

Due to power and area constraints, high radix routers were used to be impractical for on-chip design. Swizzle-Switch network design [18][19] shown that on-chip high radix router are feasible. The router can scale up its radix up to 64 while supporting 128-bit channels, consuming less than 2W of power and operating at a frequency of 1.5GH in 32nm technology.

Large-scale off-chip interconnection networks using the high radix off-chip
Figure 5.1. Comparison of the average number of hops (uniform traffic) between different network topology [5].

routers have been used in data center architecture [72][73][74][75]. Many novel network topologies have been proposed to increase the scalability and reduce the cost. Flattened Butterfly, Dragonfly and Hypercube topologies show their capability to work on on-chip multi-core network [39][21][40][76]. The dragonfly topology, which provides a low network diameter and large bisection bandwidth, is being explored as a promising option for building multi-Petaflop/s and Exaflops/s systems [77][78]. In Figure 5.1, Besta [5] shows that under uniform traffic, dragonfly topology has a 2.6 average number of hops which is better than 3.5 of flat butterfly and 10.5 of torus 3D at network size of 3000. As can be seen from Figure 5.2, the dragonfly topology is a hierarchical network: high-radix routers are used to organize the network into a two-level, all to all connected system. First, neighboring routers are connected with each other via the local network to form a group. Second, each router in the group builds connections, via a global channel, with routers in other groups. In the global net-
Figure 5.2. Dragonfly Topology is an all connected hierarchical network: all groups are connected with each other and all routers in the same group also get connected with each other.

work connection, we enforce a fully connected graph, connecting each group with every other group. There are three types of connections for each router. The terminal channels are used to connect with Processing Elements (PE), local channels are used to connect with local routers, and global channels are used to connect with routers in other groups. In the dragonfly topology, a minimal routing algorithm takes a flit, the minimum transmission unit of a packet, three hops from source to destination. First, the flit travels from the source router to the router that has a connection to the destination group through a local channel. Next, the flit travels through the global channel from its source group to its destination group. Finally, the flit travels within the destination group from the received router to the destination router. As each flit will travel through two local channels and one global channel providing an effective load balance for the dragonfly topology entails maintaining a ratio of two local channels for every
one global channel.

5.2 Indirect Adaptive Routing

In large interconnection networks, routing methods can be categorized as minimal (MIN) and non-minimal. MIN routing chooses a path that takes the minimal number of hops from the source router to the destination router. For benign traffic pattern, dragonfly topology will have the best performance using MIN. But MIN will create severe congestion on the global channel under adversarial traffic. To solve this problem, researchers proposed Non-Minimal routing algorithm. Non-minimal routing can avoid congested links on specific channels by distributing the packets on different paths which take more hops than the minimal routing path. In dragonfly network, Non-Min algorithms such as Valiant’s Algorithm (VAL) [79] can balance the load at the cost of double the hop count. With VAL, the packet will be sent to a random selected intermediate group first and then go to the destination group.

5.2.1 Globally Adaptive Routing

The Universal Globally Adaptive routing algorithm (UGAL) [80][81] was proposed to optimize performance and load-balancing by switching between MIN and Non-MIN routing on a per-package basis. UGAL performs MIN routing under low network traffic and changes to Non-MIN routing when there is congestion on the path. For example, under benign traffic patterns, such as a uniform random traffic pattern, UGAL chooses minimal routing to achieve the best performance. For an adversarial traffic pattern, UGAL will perform VAL to perform load-balancing.
Figure 5.3. Credit Flow Control: router decreases CR when it sends out a flit and increases CR when it receives a credit.

Figure 5.4. Source Router cannot get the global queue information directly from its CR.

The UGAL can be summarized as follows:

\( Q_m \) is a minimal queue depth that represents the router output congestion. \( H_m \) is the minimal hop count. \( Q_{nm} \) is the non-minimal queue depth, and \( H_{nm} \) is the non-minimal hop count. In dragonfly network, non-minimal routing will take two times the hop count of minimal routing, so the UGAL can be simplified as:

In a traditional mesh-based network, source router can get its neighbor router’s input queue length directly, and the downstream router’s input queue length is used in above equation. In dragonfly topology, global channel limits the network performance due to its limited number and high latency. So
it will be a good idea to use global channel queue length in above equation [82]. However, in most cases, the source router cannot get the global channel queue length directly. Since the router only connects with a few routers in other groups, it needs to send the packet to another router which has a connection with the destination group first. Then the intermediate router will forward the packet to the destination group through the global channel. So, source router can not get global queue information. In Figure 5.2, if the left router in group 0 wants to send a packet to a router in group 3, it should transmit the packet to the right router first. The right router then will forward the packet to group 3. Before a router transmitting a flit, it should determine whether space exists in the downstream router. A credit-based flow control route provides one method of determining whether space exists in downstream routers. In the credit-based flow control router, a credit counter (CR) monitors the downstream router buffer status. As shown in Figure 5.3, whenever the downstream router sends out a flit, a credit will be sent to the upstream router. Then the upstream credit counter (CR) will increase by one, indicating one more space is available in the downstream router. If the upstream router sends one flit to the downstream router, then the credit counter will decrease by one. A credit counter works fine for a direct adaptive routing algorithm, but when it comes to the vast interconnection network, it performs poorly. In a dragonfly topology, global channels can bottleneck the network performance: as only one-third of all channels should be global. More importantly, the source router cannot get global channel information directly. As shown in figure 5.4, the source router can only gets the downstream router R1’s and R2’s buffer occupancies rather than the global channels’ occupancies. This severely increases the difficulty to choose between MIN and VAL.
Indirect adaptive routing (IAR) [82] is then adopted by UGAL when congestion information is not directly available at the source router. In IAR, the global channel queue length or congestion information will be transmitted to source router by the intermediate router or by adding an extra link. The router can also stores all the global channel information in the group at the cost of area overhead. Credit timestamp queue (CTQ), Progressive adaptive routing (PAR), Piggyback routing (PB), and Reservation routing (RES) [82] are all good IARs for high radix router network topologies.
For RES routing, every global channel maintains a reservation counter to record the flit queue length. When the source router decides to send a packet, it will send a reservation flit to minimal global channel router. The reservation flit contains source router’s average global channel queue length. Then the intermediate router will compare the minimum global queue length with the average queue length by $Q_m \leq 2Q_{avg}$. If the result is correct, the intermediate router will send a positive signal to the source router and the source router route the packet minimally. Otherwise the source router will route the packet non-minimally.

In PB routing, the global channel state information will be sent to all the routers through an extra link in the same group whenever a packet is trans-
mitted. The packet will be routed minimally only if the received information shows the global channel is uncongested. Otherwise, the packet is routed non-minimally. To reduce the extra link size, the global channel information is stored as one bit based on $Q_m \leq 2Q_{avg}$. $Q_{avg}$ is the average global queue length in the same group. $Q_m$ is the channel’s queue length.

However, all of these methods use global queue depth as a congestion indicator and end up using $Q_mH_m \leq Q_{nm}H_{nm}$ as criteria for deciding between MIN and Non-MIN. On the other hand, we wish to compare which paths take a shorter time for the packet to traverse. Buffer occupancy is a good indicator of efficiency but can suffer from inaccuracies. For example, during a channel stall, it may take longer for the packet to traverse a path even if the buffer occupancy is at a low level. Also, traffic patterns are always switching to each other. And methods mentioned above don’t have a proactive mechanism when the traffic pattern suddenly changes. In this chapter, we compare our algorithm with the state-of-art indirect adaptive routing algorithm. The result shows that our algorithm outperforms PB routing under both uniform random traffic and adversarial traffic. To deal with the sudden change of the traffic, we implement congestion prediction mechanism. For example, prediction of congestion occurrence in min-path router can be used to trigger the source router choice non-min routing earlier. Also, our design shows a faster response time when the traffic changes.

5.2.2 Predictive Passing Time Routing

Instead of using global queue depth as a parameter to choose between MIN and Non-MIN, we present predictive passing time routing. The fundamental idea of predictive passing time routing is to choose the path that takes the minimal
time to transmit the message to destination.

As we mentioned, upstream router credit counter is decremented when a flit is sent to the downstream router. And once the packet leaves downstream router, credit is sent back to the upstream router and the credit counter is increased. The time period between a flit sent out and a credit received is a good indicator of congestion. It is real time that it takes to go through one hop. Figure 5.5 can better describe the approach. We use R1 to record the time period between a flit is sent out, and a credit is received. The monitored delay can reflect the time for flit to go through next router. In this design, R1 will not immediately send back the credit to source router when it sends out a flit. Instead, it uses processed delay information to delay its credit. So, the source router can know the global congestion according to the delayed credit information. Since the global channel is the bottleneck in the network, credit returned from the global channel will not be delayed. To avoid source routers make a decision based on single delay information, the design has eight registers to store the most recently monitored delay time and use their average to delay the returned credits. The hardware implementation is given in Fig. 5.6. When the router receives credit, the delay time will be calculated and stored in a register. Also, the sum of all delay time will be re-calculated. Since division will generate large area overhead, we use shift registers to calculate the average. The shift registers will shift three bits to the right which means that the sum is divided by 8. Then, we subtract a base value from average. The base value is used to adjust the delay time of credits. Otherwise, the delay will be too long for the source router to wait which can generate additional latency. The base value is an experimental result and hard to choose. Based on the by analysis the simulation results, we choose the base value that can postpone the credits transmission and with-
out introducing too much latency. Finally, before sending credit back upstream router, the credit is delayed based on the value from the subtraction unit.

To make our routing more proactive, we implement history-based prediction module in our design [83]. History-based prediction works with a predefined history window (control period), during which of the variable of interest, $t$, is sampled and the averaged at the end of the window. To predict the average value of passing time, $t_{pred}$, for the next history window, the following equation is used:

$$t_{pred} = \frac{W \cdot t_{curr} + t_{past}}{W + 1} \quad (5.1)$$

Where $t_{curr}$ is the computed average value of the delay in the current history window, $t_{past}$ is the previous prediction made during the history window, and $W$ is a user set parameter.

When source router receives the delayed credit from the downstream router, it will simplify the delay information and calculate the predicted passing time based on the above equation. Then the routing algorithm will work as follows:

$t_m$ is predicted passing time return from a minimal path, and $t_{nm}$ is predicted passing time return from the non-minimal path. Since the flit will go through one global channel in the minimal path and two global channel in the non-minimal path. We doubled the predicted passing time for non-minimal path.

5.3 Evaluation

5.3.1 Experimental Setup

We implement the proposed dragonfly network using an event-driven NoC simulator [2] and perform simulations under uniform random (UR) traffic, worst-
case (WC) traffic and the transition between each other. The network makes
fully inter-group interconnection in fifty-one groups of routers. Each group con-
tains ten routers that are also fully connected to each other through local ports.
Each router has five ports connect with processing element(PE), nine ports con-
nect with local routers and the rest connect with routers in other groups through
a global channel. We adopt credit flow control router architecture [60]. To make
the routing deadlock free, two VCs are needed for minimal routing, and three
VCs are required for non-minimal routing [21]. The input buffer depth is set as
16. Based on a previous study [84], the on-chip traffic is composed of approxi-
mately 80% short packets and 20% long packets. In our simulation, 80% of the
packets are 64-bit wide, 20% of the packets are 512-bit wide, and we set out link
width 128-bit. We set the delay of global channel 100 cycles and the delay of
local channel ten cycles.
5.3.2 Uniform Random Traffic

We compare four different IAR algorithm under UR as shown in Fig. 5.7. VAL algorithm chooses the non-minimal routing path all the time and creates the highest latency under uniform traffic. MIN algorithm, on the other hand, can provide the lowest delay since it chooses minimal routing path all the time.

The PB algorithm broadcasts link state information of the global channels to all adjacent routers. To avoid additional hardware overhead, only one bit is transmitted to all routers. And this state bit is set according to the global channel buffer depth. The link state bit vector is piggybacked on every packet and broadcast on idle channels. The source router will choose minimal or non-minimal path according to the state bit. However, one-bit state bit information derived from the global channel buffer depth is not accurate compared with predictive passing time routing. As can be seen from Figure 5.7, predictive passing time routing begins to shows a lower latency than PB after the injection rate goes beyond 0.5.

5.3.3 Worst Case Traffic

Figure 5.8 shows the results of four algorithms under WC traffic. During the simulation, all routers will generate packets to some specific destinations. MIN makes all source routers choose shortest routing path and cause severe congestion on the global path. As a result, MIN reaches its saturation state at a very low injection rate. VAL shows the best performance. In VAL algorithm, packets randomly choose one router in the same group, then they go through the intermediate group to the destination. This method can effectively distribute the traffic.

PB and predictive passing time routing have a better performance at low in-
jection rate since they can choose a minimal path when the workload is light. As the injection rate increase, VAL outperforms PB and predictive passing time routing. When the injection rate comes to 0.3, predictive passing time routing has a better performance than PB. The reason is apparent: when severe congestions occur, the buffer depth is very close between different channels which make it very difficult for PB to choose between VAL and MIN accurately.

Under both traffic conditions, predictive passing time routing shows better performance.

5.3.4 Transient Traffic Response

Traffic patterns are always switching to each other. In this section, we measure the performance of PB and predictive passing time routing on a cycle-by-cycle
basis as traffic is switched between UR and WC. To better evaluate the response we set the injection rate at 0.45.

5.3.4.1 Uniform Random to Worse Case

As we can see from Figure 5.9, when the traffic change from UR to WC, PB’s latency increase quickly to a very high level in the first 30 cycles. And another 30 cycles are needed for PB to return to the stable state. However, predictive passing time routing’s latency increases gradually to its saturation state. The result shows that PB makes a more unwise choice between VAL and MIN than predictive passing time routing when there is an immediate traffic change. Also, predictive passing time routing shows a fast response time. It takes predictive passing time routing only 30 cycles to return to its stable state, but PB needs about 60 cycles. The reason is that predictive passing time routing needs a base of 8 cycles to update the delay time.

5.3.4.2 Worst Case to Uniform Random

Figure 5.10 shows the two methods’ transient responses when the traffic pattern is switched from WC to UR. Both predictive passing time routing and PB show an immediately respond to this change. Because under WC, only a few channels in the network are congested, when traffic pattern is switched to UR, most packets will choose other paths immediately.

5.4 Conclusion

In this chapter, we proposed a novel indirect adaptive routing method predictive passing time routing that increases the performance of the large-scale inter-
Figure 5.9. UR to WC traffic pattern change at 0.45 injection rate

Figure 5.10. WC to UR traffic pattern change at 0.45 injection rate
connection networks by reducing global network congestion. We do this without affecting the improvements to scalability or cost that the dragonfly topology with indirect adaptive routing naturally provides. Our contribution deals explicitly with the problem of choosing between minimal routing (MIN) and non-minimal routing (Non-MIN) according to global channel queue length. We show that efficient packet routing in the network cannot rely on any individual nodes queue length. Hence, we propose a predictive passing time routing method and show simulation results that outperform state of the art piggyback (PB) routing.
Chapter 6

Fault Tolerant Router Design

6.1 Introduction

The development of deep sub-micron technology has highlighted the importance of IP interconnect. A high number of Silicon IPs are often integrated onto a single chip [6]. Conventional bus and crossbar architectures are often unable to meet the multi-core communication requirement. The NoC architectures have been proposed to replace traditional global interconnects. However, a complex system such as NoC based System-on-Chip consists of billions of transistors which makes it vulnerable to faults; one single transistor failure in one IP can even break down the entire system. Typically, faults refer to circuit malfunctioning or data errors, which can occur due to transistor aging, crosstalk noise, fluxes of neutron and alpha particles, power noise, energetic particle strikes, signal glitch, and skin effect [85]. Also, modern processor manufacturing technology involves many different factors that can influence the quality of a chip when it comes off the production line [86]. To increase the yield and maintain performance, fault tolerance and reconfiguration [87] are becoming important design concerns in the future processor design [59, 88, 89]. Traditionally, reliabil-
ity targets are achieved by employing a fault-avoidance design strategy. First, the sources of possible computing failures are estimated, and then the significant margins and guards are placed into the design to ensure it will meet the intended level of reliability [90]. However, as feature sizes near the atomic scale, extensive variation and wear out inevitably make margining uneconomical or impossible. To deal with this, processor designers have to come out with new designs that can address system reliability. In the area of NoC, many techniques have been proposed to address this problem. Due to NoCs inherent structural redundancy and opportunities for fault diagnosis and reconfiguration, using fault tolerant techniques to avoid faulty nodes is very popular. Dally presented a switching element design that uses an adaptive routing algorithm alone with the link level transmission protocol [91]. In this algorithm, the route will only avoid the broken links or nodes. However, this strategy abandons the functional PE, which can severely reduce the processor performance and may not be acceptable in heterogeneous designs. Also, the adaptive re-routing can increase the link power consumption.

Fault tolerance can also be achieved at the circuit level through self-repairing strategy. Triple modular redundancy (TMR) [90] and error correction codes (ECC) [92] protection mechanisms are often used to detect and recover the fault. In these techniques, cyclic redundancy checkers are implemented in the output channels and routing output channels. To compare the output flit, cyclic redundancy checkers also have to implement additional buffers to store the flits. Swapping and bypass techniques are used to recover the faults [93]. Also End-to-End error detection [94] and built-in self-test mechanisms are used to detect and find the error locations. These kinds of techniques can significantly increase the complexity of the design and the area overhead. Approaches using spare
components and reconfigurable links, along with adaptive or dynamic routing algorithm are commonly used [41, 95]. When routers are the faulty components, backup or multiple paths can be utilized to generate spatial redundancy thus achieving fault tolerance. Designing additional ports which connect multiple routers with one PE can also be used to avoid PE isolation due to a single router failure [96]. Although this kind of approach can reduce the design complexity, it can cause area overhead due to spare components, links or routers and can also result in increased leakage power consumption.

In this chapter, we present a novel router scheme that solves the above problems. Instead of using additional components or complex fault recovery mechanism, our design transfers the workload of the faulty router to the neighboring router. Also, we present a revised XY-Routing algorithm to work with the new router architecture. Compared with other fault-tolerant algorithms, the proposed method can achieve configuration by re-routing instead of re-mapping the PE. Simulation results show that the overall reliability of a NoC is significantly enhanced with comparatively low hardware overhead. We also evaluate our design with fault tolerance metrics including system reliability and mean time to failure.

6.2 Fault Tolerant Router Architecture

Figure 6.1a shows a traditional fault free mesh based NoC architecture. Each PE is connected to a router, and each router is connected to another four routers via links. PEs communicate with each other through routers. Figure 6.1b shows a scenario where a router failure can isolate the corresponding PE from other routers, which means that the PE cannot contribute to the system performance. In a homogeneous architecture, this can result in a waste of computational re-
source; however in heterogeneous architecture, this can directly break down the entire system.

Figure 6.2b shows the architecture of the proposed router. In comparison with the baseline router Figure 6.2a, we connect the local port with one of the other ports directly through NMOS transistors due to its simplicity. Also, we use one-bit fault detection signal to control the transistors. When the router works well, the NMOS transistors are turned off, and they will turn on if the router fails. As shown in Figure 6.3a, the fault detection is mainly achieved through ECC. We use ECC to monitor the flit information. ECC can give us a lower overhead solution for state elements than TMR. Like TMR, ECC provides a unified solution to detection and recovery. If the faults are under the recovery capability of ECC, then the router will still work as a functional one. However, if the error is out of control, then the fault detection unit will generate a fault signal to the control switch. The NMOS transistors will turn on
and then connect the PE with the linked router. The additional components are avoided since the workload is transferred to the neighboring router instead of self-recovery. To lessen the area overhead, we connect the local port to no more than one port during faults. If local port is connected with two or three ports, we need to redesign components such as RC, VC, SA and CS which is another way of component redundancy. As illustrated in Figure 6.3b, if the router fails, then it loses the capability of routing the message between the PE and other corresponding routers. However, the faulty router can connect the PE with one of the neighboring routers directly protecting the PE from isolation.

### 6.3 Fault Tolerant XY-Routing Algorithm

Routing plays a vital role in the efficiency of NoC. Different NoC topologies offer different routing algorithms to achieve desired performance and power
consumption. Routing algorithms can generally be classified into adaptive and deterministic routing [97]. For 2-Dimension mesh based NoC topology, XY routing algorithm [98] is commonly used. In XY routing algorithm, current router address in x-axis Cx is compared with destination router address Dx. If Cx is not equal to Dx, then the flit will be forwarded in X dimension until Cx is equal to Dx. If Cx equals to Dx, then it will compare the router address in y-axis and route the flit until Cy equals Dy. Due to its intrinsic characteristic, XY routing never runs into deadlock or livelock. In this chapter, we present a two-step revised XY-Routing algorithm to work with our design.

**Step 1: Check the faulty router location** We assign the linked neighbor router as the source router or destination router if the source router or destination router is connected with a faulty router.

**Step 2: Insert XY-Routing** Do XY-Routing first, if it meets the faulty router then move one step on another coordinate (There are two cases here. Case 1:
if another coordinate address is already equal to the destination address, then move it randomly. Case 2: if another coordinate is not equal to the destination address, then move it in the direction that can make it closer to the destination), then move back and continue on the original XY-routing.

### 6.4 Performance Analysis

To evaluate the design, we evaluate two key fault tolerance metrics: Reliability and Mean Time to Failure (MTTF).
6.4.1 Reliability Analysis

The reliability of a NoC router $Rr(t)$ [99] is the probability that a router performs its functionalities correctly from time 0 to time $t$. It is decided by the failure rate, $\lambda$, which is measured by the number of failures per time unit. After the router has passed the infant mortality period, we can express $Rr(t)$ using exponential failure law:

$$Rr(t) = e^{-\lambda t}$$

(6.1)

The reliability model for the traditional $NxN$ mesh are given below:

$$Rsys = (Rt)^{n \times n}$$

(6.2)

It is obvious that all routers have to be functional to ensure the systems reliability. Normally the failure rate of the router is $\lambda = 0.00315\text{ (times/year)}$ [100]. Our design will reduce the $\lambda$. Assume that $\lambda$ is reduced to half of it in our proposed design. Since our design is simple, it should be more robust than before which means should be even more smaller. The reliability of different mesh size over 1 to 10 years is shown in Figure 6.5. In our architecture, the reliability of a router is not only determined by itself, but it is also associated with the neighboring router from the system view. Since one router only connects with a specific router when it fails, the system reliability will be adjusted as following:

$$Rsys = (Rt + (1 - Rt) \times Rt)^{n \times n}$$

(6.3)

The reliability of different mesh size NoC over 1 to 10 years is shown in Figure 6.5. As shown in the figure, the NoC implemented with the proposed router outperforms the traditional one. The reliability curve of traditional 1010 NoC
drops from 0.74 to 0.05 over 1 to 10 years. However, the reliability of our design only drops down to 0.91 in the 10th year. Also, we can see from Figure 6.5c that the reliability gain increases exponentially with the year especially for NoC with large mesh size.

### 6.4.2 Mean Time to Failure Analysis

MTTF [101] is the average time before a system fails. For a single router MTTF can be expressed as the area under the reliability curve:

\[
MTTF = \int_{0}^{\infty} R(t) \, dt.
\]  

(6.4)

The above equation is the \( MTTF \) for a single router. We can get the \( MTTF \) for the \( N \times N \) meshed NoC as following:

\[
MTTF = \int_{0}^{\infty} e^{-n \times n \lambda} \, dt = \frac{1}{n \times n \lambda}.
\]  

(6.5)

The MTTF of NoC implemented with proposed router is given by:

\[
MTTF = \int_{0}^{\infty} ((R(t) + (1 - R(t)) \ast R(t))^{n \times n}) \, dt.
\]  

(6.6)

As can be seen from Figure 6.7, the new router architecture has a significant improvement in MTTF. For the mesh size of 10*10, MTTF is nine times higher than the traditional architecture. The MTTF gain increases linearly with the mesh size which means that the presented architecture will show even more significant improvements in the future NoC designs.
Figure 6.5. Reliability and Gain Analysis of Various Size of Mesh from 1-10 Years
6.5 Power Analysis on Application Reconfiguration

6.5.1 Power Modeling

To measure the energy consumption of proposed router architecture, we use the bit energy metric proposed in Yes module [102].

\[ E_{comm} = E_{Lbit} \sum_{a_{ij}} w(a_{ij}) \text{dist}(t_i, t_j) + E_{Rbit} \sum_{a_{ij}} w(a_{ij}) \text{dist}(t_i, t_j) \tag{6.7} \]

In the above equation, \( E_{Lbit} \) is the energy consumed by the link, \( E_{Rbit} \) is the energy consumed by the router, \( w(a_{ij}) \) is the communication volume between two IPs and \( \text{dist}(t_i, t_j) \) represents the Manhattan distance between tile i and tile j. We simulated our design using ORION 2.0 simulator, the results show that the power consumption of presented router is 1.09 times higher than the original one. The power consumption for presented NoC architecture will be:

\[ E_{comm} = E_{Lbit} \sum_{a_{ij}} w(a_{ij}) \text{dist}(t_i, t_j) + 1.09 \times E_{Rbit} \sum_{a_{ij}} w(a_{ij}) \text{dist}(t_i, t_j) \tag{6.8} \]
In our architecture, the flits do not need to go through the faulty router, so the routers' Manhattan distance will decrease 1 unit and the links' Manhattan distance will increase 1 unit.

### 6.5.2 Faulty Router Connection Topology

When a router fails, we need to decide which neighboring router should be connected to the corresponding IP. Two faulty router connection topologies are proposed for mesh-based NoC as shown in Figure 6.7. In Figure 6.7a, the faulty router is connected via x-axis or y-axis. If router 2 fails, the corresponding PE will be connected to router three automatically. However, this topology faces the problem that it can cause congestion for a specific router. For example, if router 8 and 11 fail then router 12 will be responsible for 3 PEs which can severely decrease the overall performance. The circle topology shown in Figure 6.7b can avoid this problem. In the circle topology, if router 8 and 11 fail then router 10 and 12 will be responsible for 2 PEs, this way the traffic congestion can be reduced.
Table 6.1. Power Consumption with Spare PEs

<table>
<thead>
<tr>
<th>Application</th>
<th>Arch NXN</th>
<th>Simulated Annealing</th>
<th>Proposed Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP filter</td>
<td>7x7</td>
<td>12.12%</td>
<td>20.89%</td>
</tr>
<tr>
<td>MPEG 4</td>
<td>9x9</td>
<td>15.09%</td>
<td>18.97%</td>
</tr>
<tr>
<td>VOPD</td>
<td>8x8</td>
<td>6.17%</td>
<td>11.67%</td>
</tr>
</tbody>
</table>

6.5.3 Power Consumption

We evaluate our approach in three commonly used applications: Video Object Plane Decoder (VOPD), MPEG-4 decoder [103], and a DSP filter [104]. For MPEG-4 and VOPD structures, 14 cores are used separately to achieve their function. Their communication architectures are shown in Figure 6.8. In this experiment, we verified the proposed design and revised XY-Routing algorithm using the power model.

To better evaluate the design, we compare the results against an simulated
Table 6.2. Power Consumption with Limited PEs

<table>
<thead>
<tr>
<th>Application</th>
<th>Arch NXN</th>
<th>Simulated Annealing</th>
<th>Proposed Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP filter</td>
<td>3x2</td>
<td>Stop Working</td>
<td>20.89%</td>
</tr>
<tr>
<td>MPEG 4</td>
<td>3x5</td>
<td>37.60%</td>
<td>18.97%</td>
</tr>
<tr>
<td>VOPD</td>
<td>4x3</td>
<td>Stop Working</td>
<td>11.67%</td>
</tr>
</tbody>
</table>

annealing based re-mapping algorithm [105] which is widely used in NoC re-
configuration. The goal of simulated annealing is to reduce power consumption
by minimizing the communication volume change. For the processor with spare
PEs, the simulation results are shown in Table 6.1. As seen here, the power con-
sumption for our architecture is only slightly increased. However, simulated
annealing algorithm achieves the goal by abandoning the PE and remapping
the entire application which has significant power consumption. The shortage
of simulated annealing algorithm can be shown in Table 6.2 for the processor
with limited PEs. In this case, simulated annealing algorithm stops working for
DSP filter and VOPD due to the limited NoC size. Since their re-mapping choice
is limited, the power consumption for simulated annealing is also significantly
increased as can be seen from MPEG4 simulation results.

6.5.4 Area Overhead

To compare the area overhead, we developed a two-stage wormhole virtual
channel router. The router has five ports, and each port has four virtual chan-
nels that can transfer 32 bits flit, and each virtual channel implemented with a
four depth FIFO. The design was implemented in Verilog and synthesized using
Cadence Design Compiler with TSMC 45nm standard cell library. Compared to
the baseline router, the presented router has an area overhead of 12TMR based
design and Vicis [93]. The result is shown in Table 6.3.
Table 6.3. Area Overhead Compared Against the Baseline Router

<table>
<thead>
<tr>
<th>Technique</th>
<th>NMR</th>
<th>Vicis</th>
<th>Presented Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area Overhead</td>
<td>100+%</td>
<td>42%</td>
<td>12%</td>
</tr>
</tbody>
</table>

6.6 Conclusion

In this Chapter, we proposed a new router design and revised XY routing algorithm for the Network on Chip fault tolerance and reconfiguration. The newly designed router will link its PE to the neighboring router when it fails. Compared with traditional router design which will isolate the well functional PE when it fails our new router design can protect the PE from isolation. This way we can protect the computation resource quite well. The new design also significantly improves the system reliability and MTTF.
Chapter 7

Conclusion

7.1 Dissertation Summary

This dissertation investigated techniques to increase the performance and reduce the power consumption of NoC and router architecture. In particular, we focus on Virtual Allocation, Switch Allocation, Routing Algorithm and Fault-Tolerant design. Major contributions of this dissertation are detailed as follows.

1. Router Architecture Design.

(a) Virtual Allocation Design.

We developed a new allocation mechanism called centralized priority management allocation. By coordinating all arbiters’ priority bit in allocator’s first stage, CPMA can help the winners from the first allocation stage have different output requests in the second stage. Thus the allocators matching quality can be increased. In NoC router design, CPMA can be used in both VA and SA. In the VA stage, CPMA can increase the packet matching quality. In SA stage, it can increase the flits matching quality. Our results show that the new design can
increase performance and reduce the power consumption at the same time. Also, the area consumption is slightly changed based on the number of router's VCs.

(b) **Switch Allocation Design.**

We redesigned SAs first stage arbiters to be priority based dynamic arbiters using round-robin algorithm. The modified unit can increase the possibility of SAs first stage arbiters to choose requests for different output ports. Hence, in the second stage of the SA, the competition for output ports will be reduced, leading more flits to travel through the crossbar in one cycle, resulting in increased throughput. Our results show that the new design can improve throughput by up to 13% for a router with eight virtual channels. Also, the new arbiter has lower worst-case latency which can help the system to increase its operational frequency.

2. **Routing Algorithm Design.**

We implemented a new Indirect Adaptive Routing Algorithm to reduce the flit transmit latency. The predictive passing time routing algorithm can increase the performance of large-scale interconnection networks by reducing global network congestion. We do this without affecting the improvements to scalability or cost that the dragonfly topology with indirect adaptive routing naturally provides. Our contribution specifically deals with the problem of choosing between minimal routing (MIN) and non-minimal routing (Non-MIN) according to global channel queue length. We show that efficient packet routing in the network cannot rely on any individual nodes queue length.
3. **Fault Tolerant Router Design.**

We presented a Fault-Tolerant Router Architecture and a modified XY-Routing Algorithm for Network-on-Chip Reconfiguration. In this new architecture, we connect the local port of the router with one of another four ports directly with NMOS transistor. The fault detection signal directly controls the NMOS transistor. If the router fails, then the NMOS will turn on and connect the PE with the neighboring router directly. This way we protect the PE even as the router stops working. The new router architecture shows significant increase in the reliability and MTTF of the NoC. As compared to most fault tolerance router designs which use spare components to achieve the goal, our design will transfer the workload of corresponding PE to the neighboring routers.

### 7.2 Future Work

The failure of Dennard Scaling makes dark silicon a severe problem for multicore design. A large portion of the chip needs to be powered off due to a given thermal design power (TDP) constraint. Esmaeilzadeh et. al. [106] shows that at 22 nm technology, 21% of the circuit must to be turned off to keep the chip working. When the transistor scales to 8 nm, the number of dark silicon grows up to 50%. However, the powered off transistors provided by dark silicon also provide potential design space to solve its problem. Power area trade-off design will be one of the solutions for NoC energy saving solution. Power Gating and Dynamic voltage and frequency scaling have evolved into an integral part of NoC designs. We plan to incorporate these two techniques into our NoC design approach. First, we will use power gating technique to turn off the routers that
attached to ideal cores; then we selectively turn on routers to make sure packets can be transmitted and optimize the latency of the packet at the same time. Second, we implement multiple Voltage Frequency optimized routers for each node, leveraging the extra transistors available due to dark silicon.
Bibliography


[18] Korey Sewell, Ronald G Dreslinski, Thomas Manville, Sudhir Satpathy, Nathaniel Pinckney, Geoffrey Blake, Michael Cieslak, Reetuparna Das,


