New circuit switching techniques in on-chip networks

SHAO TENG LIU

Doctoral Thesis in Electronic and Computer Systems
Stockholm, Sweden 2015
Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologi doktorsexamen i datalogi torsdagen den 04 Dec 2015 klockan 9.00 i Sal A, Electrum, Kista, Stockholm.

© Shaoteng Liu, June 2015

Tryck: Universitetservice US AB
Abstract

Network on Chip (NoC) is proposed as a promising technology to address the communication challenges in deep sub-micron era. NoC brings network-based communication into the on-chip environment and tackles the problems like long wire complexities, bandwidth scaling and so on. After more than a decade’s evolution and development, there are many NoC architectures and solutions available. Nevertheless, NoCs can be classified into two categories: packet switched NoC and circuit switched NoC. In this thesis, targeting circuit switched NoC, we present our innovations and considerations on circuit switched NoCs in three areas, namely, connection setup method, time division multiplexing (TDM) technology and spatial division multiplexing (SDM) technology.

Connection setup technique deeply influences the architecture and performance of a circuit switched NoC, since circuit switched NoC requires to set up connections before launching data transfer. We propose a novel parallel probe based method for dynamic distributed connection setup. This setup method on one hand searches all the possible minimal paths in parallel. On the other hand, it also has a mechanism to reduce resource occupation during the path search process by reclaiming redundant paths. With this setup method, connections are more likely to be established because of the exploration on the path diversity.

TDM based NoC constitutes a sub-category of circuit switched NoC. We propose a double time-wheel technique to facilitate a probe based connection setup in TDM NoCs. With this technique, path search algorithms used in connection setup are no longer limited to deterministic routing algorithms. Moreover, the hardware cost can be reduced, since setup requests and data flows can co-exist in one network. Apart from the double time-wheel technique for connection setup, we also propose a highway technique that can enhance the slot utilization during data transfer. This technique can accelerate the transfer of a data flow while maintaining the throughput guarantee and the packet order.

SDM based NoC constitutes another sub-category of circuit switched NoC. SDM NoC can benefit from high clock frequency and simple synchronization efforts. To better support the dynamic connection setup in SDM NoCs, we design a single cycle allocator for channel allocation inside each router. This allocator can guarantee both strong fairness and maximal matching quality. We also build up a circuit switched NoC, which can support multiple channels and multiple networks, to
study different ways of organizing channels and setting up connections. Finally, we make a comparison between circuit switched NoC and packet switched NoC. We show the strengths and weaknesses on each of them by analysis and evaluation.
To mother, father and Jie
Acknowledgment

This is the first chapter of the thesis, but this is the last chapter I write. This is the easiest part for the readers, but this is the hardest one for me — I am indebted. I am indebted to many people who have offered their generous help and genuine concern to me during my PhD study. I would like to try my best to express my sincere gratitude here.

I am especially grateful to my two supervisors: Axel Jantsch and Zhonghai Lu. I would like to thank Axel for giving me the opportunity for being his student and inspiring me with many good ideas and interesting topics. Axel is a very respectable person. He sets a good example to me on how to behave as a researcher. I would like to thank Zhonghai for imparting me the knowledge and skills, for sharing the wisdom of doing research, for motivating me and helping me, for giving the warmth and strength to get through difficulties in both life and research.

I want to specially thank several people who helped my PhD research career. I would like to thank Prof. Elena Dubrova. The talks with her are always very encouraging. She also reviewed this thesis. I would like to thank Graham Schelle for giving me the great opportunity to experience the industrial research in American. I would like to thank my Master supervisor Prof. Jimmei Lai, who inculcated me with the love of research. I would like to thank Prof. Lirong Zheng and Prof. Shili Zhang, who were the first to introduce me about the on-going research topics in KTH and Sweden.

Additionally, I would like to deeply thank many current and former colleagues in my department. I would like to thank Prof. Ahmed Hemani, Ingo Sander, Alina Munteanu and Johnny Öberg for facilitating my PhD study. I would like to thank Ming Liu, Yuan Yao, Jian Wang, Shuo Li, Pei Liu, Yanchen Long, Mohammad Badawi, Xueqian Zhao, Huimin She, Nan Li for the inspiring discussions and laughter we have together. I would also like to thank Jue Shen, Li Xie, Chanying Zhai, Junhe Gan, Ning Ma,
Kunlong Yang, Geng Yang, Botao Shao, Qiansu Wan, Zhuo Zhou, Jia Mao, Tao Sha and Byron Roberto Navas Viera for all the joys and fun shared with me.

Furthermore, I would also like to extend my gratitude to colleagues and friends in KTH, Xilinx and in SICS. Especially, I would like to thank Prof. Mats Bronson, Patrick Lysaght, Rebecca Stainert, Jan Ekman, Per Kreuger for their help and concern.

Moreover, I would like to give my heartfelt thanks to my friends. Thank Song Lu, Zhiyun Pei, Bo Wei, Jing Sun, Ze Ni, Jitong Sun, Kai Yu, Yueyue Lin, Martin Ericsson and Jing Fu for accompanying me in Sweden for so many years.

Finally, and most importantly, I am indebted to my parents and my wife Jie Ji. Without your consistent and selfless supports, this work will never be finished. Many thanks to my parents. Thanks for your permanent love and good-to-excellent care and concern. Many thanks to Jie. Thanks for being with me all along the way. Thanks for making every day of mine special.

Shaoteng Liu
October 2015, Stockholm Sweden
Contents

List of Figures xi
List of Abbreviations xiii
List of Publications xv

1 Introduction 1
  1.1 Background 1
  1.2 Classification of NoCs 4
  1.3 Contributions 7

2 Dynamic Connection Setup 11
  2.1 Introduction 11
  2.2 Parallel probe based connection setup 14
  2.3 Future work 21

3 Time Division Multiplexing 25
  3.1 Introduction on TDM NoCs 25
  3.2 Double time-wheel based dynamic connection setup 28
  3.3 Highway in TDM NoC 36
  3.4 Future work 45

4 Spatial Division Multiplexing NoC 47
  4.1 Introduction 47
  4.2 An allocator for channel allocation 49
  4.3 Sub-networks and sub-channels 56
  4.4 Circuit switching versus packet switching 59
  4.5 Future work 63
CONTENTS

5 Summary 65
  5.1 Thesis summary .................................. 65
  5.2 Future directions ................................ 66

Bibliography 69

Publications 82

A Parallel Probing: Dynamic and Constant Time Setup Procedure in Circuit Switching NoC 83

B Parallel probe based dynamic connection setup in TDM NoCs 91

C Highway in TDM NoC 99


E MultiCS: Circuit Switched NoC with Multiple Sub-Networks and Sub-Channels 117

F Analysis and evaluation of circuit switched NoC and Packet Switched NoC 131
# List of Figures

1.1 The structure of a typical bus ........................................ 2  
1.2 AXI interconnect: an example of point-to-point communication architectures ........................................ 2  
1.3 The overview of NoC .................................................. 4  
1.4 The overview of circuit switched NoC ............................... 5  

2.1 Path diversity expressed by using Pascal’s triangle ............ 15  
2.2 Set up a connection with the parallel probing search algorithm ... 17  
2.3 Live-lock and priority preemption based live-lock avoidance method 19  
2.4 General operating phases of connection setup with parallel probing algorithm ......................................... 20  
2.5 Establish a multicast tree by using the parallel probing algorithm 22  

3.1 Illustration of TDM concept ........................................... 26  
3.2 Source routing based configuration in TDM NoCs needs reserved slots of a connection to be adjacent .................... 26  
3.3 Illustration of the slot-table based configuration method ........ 27  
3.4 The overview of a router which supports the probe based connection setup method ....................................... 29  
3.5 Route back Ack/Nack with the double orientation time-wheel technique ......................................................... 31  
3.6 Router architecture of the double time-wheel TDM NoC ........ 33  
3.7 The slot-table in a router ............................................. 35  
3.8 Illustration of slot utilization in TDM NoC ......................... 37  
3.9 Illustration of the function of a highway ......................... 38  
3.10 The overview of a router with highway support ............... 40  
3.11 Highway setup process .............................................. 41
List of Figures

3.12 Flit storing and forwarding rule ........................................ 43
3.13 The structure of the two-stage arbitrator .............................. 44

4.1 TDM NoC and SDM NoC .................................................. 48
4.2 Illustration of a channel allocation problem in NoC .................. 49
4.3 Illustration for homogeneous resource allocation problem ........... 52
4.4 Reduction of the request matrix ......................................... 53
4.5 Overview of the resource allocation logic ............................... 54
4.6 The loop-free structure of the WTF allocator ........................... 55
4.7 Channel organization methods .......................................... 58
4.8 Unbalanced traffic caused by large packets in packeted switched NoC using round-robin arbitration ......................... 61
4.9 Markov model for circuit switched NoC .................................. 61
# List of Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>AXI</td>
<td>Advanced Extensible Interface</td>
</tr>
<tr>
<td>FIFO</td>
<td>First-In-First-Out</td>
</tr>
<tr>
<td>HRA</td>
<td>Homogeneous Resource Allocation</td>
</tr>
<tr>
<td>HWC</td>
<td>Highway Channel</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>MRR</td>
<td>Massive Round-Robin</td>
</tr>
<tr>
<td>MultiCS</td>
<td>Multi-channel and Multi-network Circuit Switched</td>
</tr>
<tr>
<td>NI</td>
<td>Network Interface</td>
</tr>
<tr>
<td>NoC</td>
<td>Network on Chip</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>SDM</td>
<td>Spatial Division Multiplexing</td>
</tr>
<tr>
<td>TDM</td>
<td>Time Division Multiplexing</td>
</tr>
<tr>
<td>VC</td>
<td>Virtual Channel</td>
</tr>
</tbody>
</table>
List of Publications

Papers included in the thesis


Papers not included in the thesis


Chapter 1

Introduction

This chapter begins with introducing the background and the classification of NoCs. Then, it gives an overview about our works on circuit switched NoC and outlines the author’s contributions in the papers enclosed to the thesis.

1.1 Background

The development of semiconductor technology shrinks the feature size of transistors while enlarging the die size. As a result, more devices and Intellectual Property (IP) cores can be integrated on a single chip. However, on-chip interconnects tend to get worse, since the shrinking feature size will cause the increase of wire delay and crosstalk. Besides, as the number of devices increases, more communication bandwidth is required. Therefore, new communication architectures are required to overcome the limitations on wire delay and bandwidth.

In the past, a single bus was used to connect all the devices (IP cores) on a chip, as illustrated in Figure 1.1. Data sent out by a device is broadcast on the bus and can be received by all other devices. At one time, only one device is allowed to transmit data to the bus. An arbiter decides which one can transmit. Bus is not a scalable solution for communication since the service rate of each device decreases as the number of devices connected to the bus increases. Besides, the broadcast way of transmitting data is inefficient. Moreover, as the number of devices grows, the bandwidth of a bus may decrease due to the growing wiring delay, which is the result of
CHAPTER 1. INTRODUCTION

Figure 1.1: The structure of a typical bus

the increase in wire length and capacitance. Consequently, bus becomes the communication bottleneck as the number of devices on a chip goes up.

Figure 1.2: AXI interconnect: an example of point-to-point communication architectures

To overcome the limitation of the single bus based communication architecture, crossbar based point-to-point communication architectures are proposed. For example, Advanced Extensible Interface (AXI) interconnect [1] allows point-to-point communications via a big crossbar structure, as depicted in Figure 1.2. This architecture enhances the communication bandwidth by allowing several data flows on-going simultaneously inside a crossbar. However, one significant drawback of this solution is that the crossbar structure does not scale up well. Firstly, the number of wires and transistors consumed by the crossbar and the arbiters scales up with $O(n^2)$, where
1.1. BACKGROUND

$n$ is the number of connected devices. Secondly, inside this architecture, the wiring delay between different sources and destinations are non-uniform since the physical distance varies. As illustrated in Figure 1.2, the distance of path $P1$ is longer than that of $P2$. As the number of devices scales up, such non-uniformity issue becomes more severe, resulting in the design of the clock system and synchronization scheme very challenging.

Because of all of these limitations and challenges, more advanced communication architectures are required. Network-on-Chip (NoC) emerges under this background. As a promising technology, NoC is expected to exceed the limitation on the communication bandwidth, overcome the increasing wiring delay problem and become a scalable communication solution. NoC, as its name suggests, intends to build a communication network inside a chip. The concept of communication network is well-known since it has been widely used in our everyday life, for example, telephony network and Internet.

NoC inherits some of the merits from those macro-networks like Internet or Asynchronous Transfer Mode (ATM) network. For example, it also distributes a number of routers between all the terminal devices and interconnects the routers and terminal devices according to a certain topology, as illustrated in Figure 1.3. We name the wires that interconnect two routers/devices as a link. Inside a NoC, point-to-point data delivery is relayed by routers. Thus, instead of connecting source and destination directly by using long wires, data transmitted from a source node takes multiple hops and traverses multiple short links to reach its destination. This approach can solve the complexities caused by long wires and thus increases the scalability. Besides, in many cases, NoCs take a regular topology, e.g. mesh (as illustrated in Figure 1.3) or torus, which are very suitable for expanding. Moreover, inside a NoC, multiple data flows can co-exist on different links, or take turns to use one link. In such a way, the communication resources are exploited very efficiently. When compared with other approaches, NoC can provide larger communication bandwidth with relatively less area cost and less wiring complexity.

NoC also presents some varieties from those macro-networks. For example, NoC design can utilize more wires for a link and operating at very high clock speed since it operates in an on-chip environment. Besides, NoC design focuses more on router delay, area cost, and power consumption. For example, NoC design restricts the usage of complicated routing/arbitration/flow control algorithms, deep buffers and so on. Therefore, existing macro-network solutions are often infeasible for NoC design. We
have to consider new communication architectures and components that are suitable for on-chip environments.

1.2 Classification of NoCs

At present, no well-formulated standard exists for guiding NoC design. There are many published NoC architectures in the literature. Generally speaking, these NoC architectures can be classified into two categories: packet switched NoC [2, 3, 4, 5] and circuit switched NoC [6, 7, 8, 9, 10].

In a packet switched NoC, a data flow is split and wrapped into blocks of a certain size, called packets. Normally, a packet is composed of a header and a payload. The header contains the information needed by the routers to direct the packet to its destination. The payload contains the information a source node wants to deliver. Communication resources, such as buffers and channels inside a router are allocated to individual packets. When a packet arrives at a router, resources are allocated. When the packet leaves, the allocated resources are reclaimed. In packet switched NoCs, flow control is often required between every two hops. This is called hop-by-hop flow control.
1.2. **CLASSIFICATION OF NOCS**

Links in a packet switched NoC are normally shared in a work-conserving manner [11] in the sense that a link is never idle if there are packets to transmit. Packets of different data flows take turns to use a shared link. An arbiter decides the order. When a packet gets its turn, a certain amount of its data is delivered by the link. Otherwise, the packet has to be buffered. Depending on the buffering policy and the amount of data allowed to deliver each time, packet switched NoC can further be classified into sub-categories like wormhole [12, 13, 14, 3], virtual cut-through [15, 16], defective [17, 2, 18] and so on.

The main difference between circuit switched NoC and packet switched NoC is that, a circuit switched NoC pre-allocates a path for a data flow before the data transfer launches. In circuit switched NoC, since all the required communication resources are pre-allocated, there is no contention and thus no need for arbitration during the data transfer process. Besides, circuit switched NoCs often do not need a header for directing individual packets since each data flow is transmitted along a pre-allocated path between the source and the destination.

In a circuit switched NoC, normally links are shared in a non-work-conserving manner [11] between different data flows, if there are link sharings, e.g. in TDM NoC. The bandwidth of a link is pre-allocated to a data flow and each flow can only use its allocated share. In circuit switched NoC, flow control is often only needed between the source and destination. This is called end-to-end flow control [19].

### 1.2.1 Fundamentals about circuit switched NoC

![General operating flow](image1.png)

![Router architecture overview](image2.png)

**Figure 1.4:** The overview of circuit switched NoC
As depicted in Figure 1.4a, the general operating flow of a circuit switched NoC consists of 3 phases: path setup, data transfer, and path release. During the path setup phase, link resources are allocated to a data flow to build up a connection from the source to the destination. During the data transfer phase, data is switched by routers on the established path. After data transfer is finished, the routers reclaim the allocated resources. The three phases are distinctively isolated. Thus, the router architecture of a circuit switched NoC can often be divided into a data path and a control path (plane), as suggested by Figure 1.4b. The control plane is used in path setup and reclaim phase. It controls the allocation and configurations of communication resources such as channels and crossbars. The data plane is used in the data transfer phase and responsible for switching data to the established path.

In the past, packet switched NoCs have been explored more thoroughly and intensively. However, circuit switched NoC has some appealing merits and could be preferable under certain traffic or service scenarios. Compared with packet switched NoC, although circuit switched NoC has path setup overhead time, it can offer guaranteed throughput and latency. Besides, it can also present lower hardware complexity and higher energy efficiency, and may work at a higher clock frequency.

**Link bandwidth sharing techniques**

Time Division Multiplexing (TDM) and Spatial Division Multiplexing (SDM) are the two techniques frequently adopted by circuit switched NoC to share the bandwidth of a link between different connections. With TDM technique, connections can take turns to use a link. Each connection can only use a link for a predetermined fraction of time. With SDM technique, a link is physically divided into sub-links. Each connection reserves a number of sub-links and use them exclusively. We will discuss the two techniques in Chapter 3 and Chapter 4 in detail.

Besides TDM and SDM, some other circuit switched NoC designs [20, 21] utilize per-connection virtual channels and round-robin arbitration to share links. To offer guaranteed throughput for each connection, this technique pre-allocates virtual channels to connections and puts limitations on the virtual channels that share a link. However, virtual channels are expensive resources, since they consist of buffers, multiplexers, demultiplexers and require separate hop-to-hop flow control. Thus, the hardware cost is high with this technique.
1.3 Contributions

In this thesis, we focus on circuit switched NoC. We present our research on advanced path setup methods and data switching techniques. The thesis summarizes a collection of papers, which are grouped into three blocks: Dynamic connection setup, Time division multiplexing, Spatial division multiplexing. Each block corresponds to one chapter. We concentrate on introducing the author’s contributions in these chapters. In the following, we summarize our contributions:

- Dynamic connection setup

  In this paper, we proposed a novel parallel probe based path searching algorithm. This algorithm can search the entire network topology and find an available path from a source to a destination in constant time. We implemented this algorithm in a circuit switched NoC for dynamic connection search and set up, by solving tricky issues like live-lock inside this setup method. We also proposed and evaluated several connection search and setup policies. Compared to previous works, our design can reduce the setup time and enhance the setup success rate. Moreover, the router in our design has a concise structure with an inexpensive and efficient implementation.

  *Author’s contribution:* The author proposed the parallel probing algorithm and utilized this algorithm in a circuit switched NoC for connection setup, made the hardware implementation, established experimental platform to evaluate the design, and wrote the manuscript.

- Time division multiplexing
CHAPTER 1. INTRODUCTION

In this paper, we proposed a double time-wheel technique that used in TDM based circuit switched NoC to facilitate the two-way communication of a TDM connection. A slot-table is shared by both upward and downward messages of a connection. Based on this technique, we introduced a probe based connection setup method into TDM circuit-switched NoC. Our design provides shorter connection setup delay and higher success rate while presenting lower hardware complexity than any previously known method.

Author’s contribution: The author proposed the idea and made the hardware implementation of the double-time wheel technique. The author also proposed a probe based path setup method that can be applied in TDM NoC for connection setup. Besides, the author established an experimental platform to evaluate the design and wrote the manuscript.


In this paper, we proposed a novel highway technique that can enhance the performance of TDM NoCs. The proposed technique can dynamically setup highways without contention. Once highways are built, the delivery of data flows can be accelerated by utilizing unallocated and idle TDM slots of links, while preserving the guarantee on minimum bandwidth and data packet order. This highway technique has no dependency on the TDM NoC architectures and introduces no additional traffic. It is a generic technique that can be applied in many different kinds of TDM NoCs.

Author’s contribution: The author developed the concept of highway, defined detailed procedures for highway setup, transfer and release. The author also solved tricky hardware implementation issues, utilized both synthetic traffic patterns and benchmarks for test and evaluation. The author wrote the manuscript.

- Spatial division multiplexing

1.3. **CONTRIBUTIONS**


In this paper, we exposed a special allocation problem in NoC environment and named it as the Homogeneous Resource Allocation (HRA) problem. We developed a novel Waterfall (WTF) allocator for the homogeneous resource allocation. The WTF allocator provides maximal matching quality while keeping the strong fairness guarantee. It can have a loop-free structure and solve a homogeneous resource allocation problem within one clock cycle. It achieves strong fairness and offers better performance and lower area than known solutions.

*Author’s contribution:* The author proposed the homogeneous resource allocation concept and the idea of the waterfall allocator. The author also efficiently implemented the allocator in hardware, proved its advantages in matching quality and fairness by comparing with existing solutions. The author wrote the manuscript.


In this paper, we explored the methods of organizing multiple physical channels in a circuit switched NoC. We proposed a Multi-channel and Multi-network Circuit Switched (MultiCS) to study channel partitioning and configuration policies. Based on the analysis and experiments results, we revealed the benefits and burden of using different number of channels and configurations.

*Author’s contribution:* The author proposed the MultiCS NoC, designed different channel partitioning and configuration policies, analyzed and evaluated these policies, and wrote the manuscript.


In this paper, we compared circuit switched NoC against packet switched NoC. We showed that performance decreases for packet switched NoC as the packet size increases, whereas it increases for circuit switched
NoC. We revealed that circuit switched NoC can operate at a higher clock frequency than packet switched NoC and thus could be better than packet switched NoC in some cases.

Author’s contribution: The author designed a circuit switched NoC and compared with packet switched NoC based on speculation, circuit-level analysis, and evaluation. The author built the experiment platform, conducted all the experiments and wrote the manuscript.

The remainder of the thesis is structured as follows. Chapter 2 introduces our research on the dynamic connection setup method. Chapter 3 summarizes our works on TDM based circuit switched NoC. In Chapter 4, we present our works related to SDM based circuit switched NoC. Finally, we conclude the thesis and discuss future works in Chapter 5.
Chapter 2
Dynamic Connection Setup

In this chapter, we introduce how to set up connections in circuit switched NoCs. Particularly, we will present our novel parallel probing algorithm, and show how to use this algorithm to dynamically search and set up connections in a circuit switched NoC [Paper A].

2.1 Introduction

How to set up connections is vital to a circuit switched NoC, since data transfer in circuit switched NoC relies on established connections. Connection setup methods affect the performance and resource utilization of a circuit switched NoC. Connection setup methods can be classified into static scheduling methods, and dynamic setup methods.

In this chapter, we focus on connection setup methods. In order to simplify our illustration, in the following discussions, we just consider the case that a physical single directional link can only be used by one communication channel without sharing with others. We will discuss link-sharing techniques such as TDM and SDM in Chapter 3 and Chapter 4, respectively.

2.1.1 Static scheduling method

Static scheduling methods schedule the channel resources of a network for connections at compilation time [22, 23]. During the past decade, many NoC architectures supporting static connection scheduling have been proposed
CHAPTER 2. DYNAMIC CONNECTION SETUP

[24, 8, 25, 26], based on a variety of static scheduling algorithms [27, 28, 29, 30, 31, 32, 33, 34, 35, 36].

Z. Lu and A. Jantsch in [37] has formulated a static connection scheduling problem. According to [37], inside a network, all the possible paths of a connection compose a tree-like search space, and a static scheduling algorithm's responsibilities are the following. 1) Explore the solution space and schedule a path for a connection. 2) Make sure all the connections scheduled on a link summed together does not exceed the link's bandwidth. 3) Try to optimize the path schedule. A good schedule should satisfy as many communication requirements as possible, and consume as less communication resources as possible.

The advantage of static scheduling methods is that advanced algorithms can be applied to optimize the scheduling of connections. However, static schedule methods often suffer from the following disadvantages. Firstly, they assume that all communication needs and connection requirements are known at compile time. Thus, they are not well suited for applications like H.264 [38] with requirements for dynamic communication setups or dynamic traffic mixes of applications. Secondly, those static scheduling algorithms are often complicated and take several iterations to produce an optimal solution. As the number of nodes inside a circuit-switched NoC grows up, the required processing time and storage area for scheduling connections can increase exponentially. Thirdly, connections scheduled by static methods cannot be aware of the dynamic fluctuation of network traffic load. Thus, the network bandwidth utilization is often sub-optimal.

2.1.2 Dynamic setup method

Dynamic connection setup methods [39, 40, 41, 42, 43, 44, 45, 46, 47] were proposed to overcome the shortcomings of static scheduling method. Dynamic connection setup methods allocate and release connections at runtime, according to dynamic communication requirements and network status. With dynamic methods, network resources can be dynamically and efficiently allocated and reclaimed.

Dynamic setup methods can further be classified into centralized methods and distributed methods.
2.1. **INTRODUCTION**

**Centralized methods**

In centralized methods, a resource allocation algorithm is running inside a special coordinator node of a network. This node manages all the network resources and connections. All the other nodes need to send requests to the coordinator node in order to set up or release a connection. When the coordinator node receives a connection setup request, it will make a quick check on available channels and decide how to allocate channels inside the network to the connection. When it receives a connection release request, it will reclaim the allocated channels. The resource allocation algorithm can be either running as a software inside a processor like [39, 40, 41, 48], or running on hardware based accelerators [43, 44]. Normally, hardware accelerators based algorithms are about 100-1000 times faster than running as software inside a processor.

After an allocation decision is made by the allocation algorithm inside the coordinator node, channel reservation process starts. The channel reservation process reserves and configures channels in different routers according to the allocation decision on a connection. Two different channel reservation approaches are frequently used in centralized methods. The first approach utilizes a dedicated network to distribute allocation decisions to routers inside a NoC [8, 26], and directly initiates the configuration processes inside each router. With the second approach [43], the coordinator node firstly sends the channel allocation decision to the source node of a connection. Then, the source node will compose and send out a reservation packet to the destination by using contention-free source routing method. This reservation packet carries all the channel reservation and routing information, and reserves the corresponding channels inside each router on its traveling path. When the reservation request reaches the destination, the connection has been established.

The main limitation with centralized allocation methods is the lack of scalability. The coordinator node needs to handle all the setup/release requests and distribute allocation decisions from/to the entire network. Such multiple-to-one and one-to-multiple traffic patterns can become the performance bottleneck. Besides, resource allocation algorithms cannot be occupied by one setup request for a long time. Otherwise, the following setup requests will be delayed and blocked, resulting in the overall performance degradation. For example, if the allocation algorithm fails to find free channels for a setup request, it has to discard the setup request immediately,
rather than retry the failed request repeatedly.

**Distributed methods**

Distributed setup methods [46, 42, 45, 49] are proposed to overcome the scalability issue in centralized methods. With distributed methods, each source node can setup and release a connection independently, without the inquiry of a coordinator node. To establish a connection, the source node generates a setup request and sends it out. Compared with the reservation packets used in centralized methods, the setup request in this case is more concise. It contains only source and destination addresses, instead of all the routing and channel allocating information. The setup request is routed towards the destination by routers. Each router computes the next hop of the setup request, and forwards the setup request to the next router. A channel inside a router is reserved by the setup request when it passes the router. When a setup request reaches the destination, a connection has been established, since the setup request has reserved channels by the routers along one path from the source to the destination. In distributed methods, the channel allocation and channel reservation happen at the same time.

Routing algorithm inside each router used for routing the setup request is crucial to distributed setup methods. It decides the speed and efficiency of setting up a connection to the destination.

In this chapter, we concentrate on dynamic distributed connection setup methods. In the following discussions, the term *path* refers specifically to *minimal path*.

### 2.2 Parallel probe based connection setup

#### 2.2.1 Problem description

Inside a network, multiple minimal paths exist between a source node and a destination node. As described in Figure 2.1, we use a mesh topology to illustrate such path diversity. For example, from the source node(0, 0) to the destination node(2, 2), there are multiple path choices. Specifically, since there is 1 path from node(0, 0) to node(1, 0) and 1 path from node(0, 0) to node(0, 1), there are in total $1 + 1 = 2$ paths from node(0, 0) to node(1, 1).

With this method, since there is 1 path from node(0, 0) to node(2, 0) and 2 paths from node(0, 0) to node(1, 1), the number of possible paths from
2.2. PARALLEL PROBE BASED CONNECTION SETUP

Figure 2.1: Path diversity expressed by using Pascal’s triangle
CHAPTER 2. DYNAMIC CONNECTION SETUP

node(0,0) to node(2,1) equals $1 + 2 = 3$. Similarly, we can find that there are also 3 paths from node(0,0) to node(2,1). Finally, the number of paths from node(0,0) to node(2,2) is $3 + 3 = 6$.

Therefore, we may conclude that the number of possible minimal paths from a source node to a destination node in mesh topology can be expressed by using a Pascal’s triangle [50], as Figure 2.1 suggests. Suppose that the source node has coordinate $(0,0)$, and the destination node is $(x,y)$. The Manhattan Distance [51] between source and destination node is $D$, where $D = x + y$. We have the number of paths $m$ as

$$m = \binom{D}{x} = \binom{D}{y} = \frac{D!}{x!(D-x)!} = \frac{D!}{y!(D-y)!} \quad (2.1)$$

By using Eq.2.1, we can simply get that there are $\binom{2}{2} = 6$ paths from node(0,0) to node(2,2), which is in accordance with our previous analysis.

As the distance between source and destination increases, the number of possible paths increases dramatically. For example, in a 4 by 4 mesh, the number of shortest possible paths from a corner node to reach the node on its diagonal corner is 20. For a 5 by 5 mesh, this number becomes 70. For an 8 by 8 mesh, it becomes 3432.

The routing algorithm used to route setup requests determines how many possible paths can be searched. Since routing algorithms are normally implemented inside individual routers and in hardware, many previous works [42, 45, 46, 52, 53, 54] just seek to apply simple deterministic routing algorithms such as X-Y routing. However, deterministic routing algorithms can only search one fixed path from source to destination. It fails to explore the aforementioned path diversity and thus significantly limits the connection setup successes probability and the setup efficiency.

In order to exploit the path diversity, powerful routing algorithms need to be developed for the setup request. It is possible to implement powerful distributed routing algorithms with reasonable hardware cost. For example, in [49, 55], Pham et al. proposed a depth-first path search algorithm and implemented it in hardware. With this algorithm, a setup request firstly travels along a certain path from source to destination. If the setup request cannot continue along the path, it will backtrack one or several nodes and try another path. This depth-first algorithm can do an exhaustive search among all the possible minimal paths. If a free path exists, it will find it. However, as a depth-first search algorithm, it takes too long time to search over all the
possible paths. As Eq. 2.1 suggests, the time complexity for the worst case is $O(D!)$, where the $D$ is the distance between source and destination.

Towards a more efficient solution, we propose a more powerful path setup algorithm which is based on a breadth-first way for path search. Unlike the depth-first setup algorithm proposed by Pham et al. [49, 55], our breadth-first setup algorithm can exploit the parallelism of hardware. It can search all the possible paths at one time in parallel. The time complexity for such a search is only $O(D)$. We name our algorithm as parallel probing.

### The parallel probing dynamic setup algorithm

![Diagram of parallel probing dynamic setup algorithm](image)

Figure 2.2: Set up a connection with the parallel probing search algorithm

We use an example to illustrate the key idea of our parallel probing path search algorithm, as illustrated in Figure 2.2. In Figure 2.2, a source node (node 1) tries to set up a connection to the destination node (node 16). At the first cycle, the source node sends out two setup probes along two productive directions toward the destination. Each probe consists of source
address and destination address to guide its routing. At the second cycle, when the two probes arrive at node 2 and node 5, each probe splits into two probes, so there are four probes in total continuing travelling toward destination along all the minimal paths. Following such a way, a number of probes will be triggered out to search all the possible paths in parallel. As a probe travels, it reserves the channels it has passed.

However, since we only need one path from source to destination in the end, our algorithm eliminates redundant channels as soon as possible. This is how our algorithm is different from the flood based setup algorithms [56]. One policy of our algorithm is that when two probes carrying the same setup request meet at the same node, one probe dies, and the channels reserved by the dead probe alone will be canceled. For example, as the second picture of Figure 2.2 describes, two probes meet at node 6 that one probe dies. With this policy, the channel between node 2 and node 6 will be canceled, whereas the channel between the source node and node 2 remains, because it is still needed by the probe that has travelled further to node 3. We identify these released redundant channels with cross markers, as shown in Figure 2.2. Following this way, in an ideal case (minimum paths exist and no contention happened between probes of different setup requests), only two probes can enter into the destination node, which is node 16. Then one probe dies and cancels its reserved path. In the end, only one path is left between the source and the destination.

We summarize the general rules of our parallel probing algorithm as follows:

1. **Probing split rule**
   
   It is inspired by the idea "cell division". At first, only one setup probe is sent out by the resource connected to the source node. When a probe enters into a node and finds multiple productive outputs towards the destination (e.g., it can have two productive outputs in a mesh topology), it will split into multiple probes to travel through all these output channels in parallel, with one probe per output for an exhaustive search and setup.

2. **Probing cancel rule**

   This rule reduces the network resource consumption during a connection setup since the probe split rule generates too many probes and
occupies too many redundant channels. The probe cancel rule mandates the release of the redundant parallel paths as soon as possible. This rule requires that 1) whenever two probes of the same request meet, one of them dies, 2) if a probe cannot proceed because of contention or obstruction, the probe dies as well. The channels reserved by a dead probe is released and reclaimed immediately. The release action proceeds backward hop by hop along the path reserved the dead probe until reaches a channel that is still in use by an active probe.

**Live-lock avoidance**

Our parallel probing algorithm has no deadlock [57, 58] issue. This is due to that if a probe cannot proceed, it will be canceled immediately and all the resources reserved by the probe is released and reclaimed. When probes do not hold resources and wait, there is no deadlock.

However, live-lock [59] may exist in our algorithm, since contentions for resources between probes belonging to different connection setup requests may create live-lock. As illustrated in Figure 2.3a, if the four connection setup requests A: 1 → 3 (it means node 1 trying to set up a connection to node 3), B: 4 → 2, C: 2 → 4, D: 3 → 1 are sent out simultaneously, then they will block each other. None of them can succeed. Retrying in a deterministic manner will not help.

![Figure 2.3: Live-lock and priority preemption based live-lock avoidance method](image)

In paper A, we proposed a priority based preemption [60] method to deal with live-lock. With this method, the live-lock illustrated by Figure 2.3a can
be resolved. E.g., if setup probe A’s priority is higher than C, it can preempt the channel reserved by C between node 2 and node 3.

In paper A, we use a setup request’s retry times and its source node ID as priority values. Actually, we can use anything as a priority, just making sure that two conflicting requests do not have the same priority.

2.2.3 Implementation of the parallel probing algorithm

Operating phases in a connection setup process

![Diagram of connection setup process](image)

Figure 2.4: General operating phases of connection setup with parallel probing algorithm

The general operating phases of the parallel probing algorithm is depicted in Figure 2.4. After probes are sent out from the source node, they proceed toward destination while splitting and dying. When a probe reaches destination, "Ack" signals will be sent back to the source node along the established path to confirm channel reservations and launch data transfer. If probes failed, channels reserved by dead probes will be released by "Nack" signal. If all the probes die, depending on different setup polices, the setup request may be retried or dropped.

According to our implementation in paper A, a probe takes 2 cycles per hop to move forward while the backward Ack/Nack takes 1 cycles per
2.3. FUTURE WORK

hop. Thus, for a single search, it takes $2D$ cycles for a probe to reach the destination and $D$ cycles for sending back the ANS signal. There is also an additional 6 cycles that are spent in the network interface as an overhead ($D$ is the hop distance between a source and a destination). In other words, after sending out probes, a source node is notified about the result of the connection setup endeavour within $3D + 6$ cycles.

Setup policies

We proposed three setup policies, which are no retry, retry for free path and retry until success.

No retry policy only tries once for a setup request, and gives up the request if the endeavor is failed. Therefore, it spends at most $3D + 6$ cycles on a setup request.

Retry for free path policy will retry the failed setup requests several times before give-up. The interval between every two adjacent retries is fixed to $3 \times D_{\text{max}} + 6$ cycles, where $D_{\text{max}}$ is the longest distance between two nodes in a topology. In a $n \times n$ mesh, $D_{\text{max}} = 2 \times (n - 1)$. After each retry, the priority of the setup request is increased by 1. Suppose there are $K$ source nodes inside a network and each node at most serves one setup request at a time. Then after $K$ retries, if the setup request still cannot succeed, it will be dropped. Since after $K$ retries, the setup request will have the highest priority inside the network. It will win every arbitration inside each router according to our priority preemption based arbitration method. If it still cannot succeed, it means that all the possible paths are occupied.

Retry until success policy has no limitation on retry times of a failed setup requests. It keeps retrying a failed request until it becomes successful. The retry interval may be set to zero, which means a source node will retry a failed setup request immediately when it is notified about the failure.

2.3 Future work

Our parallel probing algorithm is an adaptive routing algorithm. It can search over all possible minimal paths and find an available path within a time complexity $O(D)$. Because of these good properties, we can explore the parallel probing algorithm in the following three directions in the future.
1. **Using the parallel probing algorithm in topologies other than mesh.** In Paper A, the parallel probing algorithm is only implemented for a mesh topology. However, dynamic connection setups in regular topologies, such as torus [61], fat tree, benes [62], clos and butterfly [63], should also benefit from our algorithm. The principle of the parallel probing algorithm is to use as many as possible probes to search among all available shortest paths in parallel, while cancelling redundant paths as soon as possible. Thus, on a network topology, as long as each router knows how to forward incoming probes toward the destination along minimal paths, our parallel probing algorithm could work.

2. **Using the parallel probing algorithm for fault tolerance purpose.** It is possible to utilize our parallel probing algorithm for fault tolerance purpose, since our algorithm can search all the minimal paths simultaneously and in parallel. Even if faults happen on some of the paths, our algorithm can still find out the fault-free path to build up a connection.

![Figure 2.5: Establish a multicast tree by using the parallel probing algorithm](image)

3. **Using the parallel probing algorithm for multicast communication purpose.** Multicast communication refers to that information sent by a source node is addressed to a group of destinations simultaneously. To support multicast in circuit switched NoC, a source node needs to be connected to multiple destinations. Instead of establishing multiple point-to-point connections, multicast can take a tree-like structure [64, 65] to connect the source node as root and the destination nodes
as leaves. In such a way, multicast can use communication channel resources more efficiently, since the consumption of channel resources can be reduced.

Our parallel probing algorithm is suitable to establish such a tree-like communication structure for multicast. As illustrated in Figure 2.5, with one setup attempt, our parallel probing algorithm is possible to establish a tree-like structure which connects all the destinations (green nodes) to the source node (the yellow node).
Chapter 3

Time Division Multiplexing

This chapter summarizes our research on TDM NoCs. Particularly, we will introduce our work on double time-wheel technique and its usage in the setup of TDM based connections [Paper B]. We will also introduce our highway technique and present its advantages [Paper C].

3.1 Introduction on TDM NoCs

3.1.1 Time division multiplexing technique

Time Division Multiplexing (TDM) [32, 35, 37, 24, 66, 34] technique has been widely used for guaranteed data transfer in NoCs. In TDM NoC, a physical link can be contention-free shared by different connections, with each connection reserving one or multiple specific time slots in a finite repeating time window. Each slot is exclusively reserved by one connection. A connection can span many links from source to destination, by reserving slot(s) at each of the links in a consecutive manner. For example, as illustrated in Figure 3.1, connection v1 spans link 1 and link 2. If slot 0 and slot 2 of link 1 are reserved by v1, then slot 1 and slot 3 of link 2 must be reserved by v1. TDM circuit switching treats the entire NoC as a single shared resource with a single arbiter. In other words, packets wait only at the ingress Network Interface (NI) until their reserved slots come, after which they progress without contention to the egress NIs, with minimal latency [67].

In the following discussions of this chapter, we define a TDM channel as a simplex link between two routers with associated buffers in a particular
CHAPTER 3. TIME DIVISION MULTIPLEXING

3.1.2 Data path configuration methods

In TDM NoC, data path configuration refers to how to configure the crossbar inside each router to switch data flits of different traffic flows correctly through the pre-reserved channels. Generally speaking, there are two kinds of methods: source routing based configuration [68, 67, 69] and slot-table based configuration [8, 70, 37].

With source routing based configuration method, a data flow is wrapped into packets, each of which contains a head flit. The head flit carries all the crossbars’ configurations of a connection. The head flit configures every
cross-bar as it travels, with one crossbar per router. Body flits of a packet follow the path paved by the head flit. Thus, it requires that the allocated slots of a connection must be adjacent, so that body flits can follow closely without interval. For example, as Figure 3.2 suggests, if a head flit of connection \( v_1 \) is delivered at slot 0 of link 1, then its body flit must be delivered at slot 1 to follow the head flit closely. Otherwise, if head flit of another flow occupies slot 1 of link 1, the configuration of the crossbar in node 1 is altered and thus the body flit of \( v_1 \) cannot be switched correctly.

Source routing does not require a slot-table inside each router to store configurations in a distributive manner. However, the configuration information of each connection needs to be stored inside the network interface of the source node of each connection.

\[ W = \begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array} \]

\[ W = \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]

![Figure 3.3: Illustration of the slot-table based configuration method](image)

In slot-table based configuration method, the configuration information is distributed and stored inside each router, as suggested by Figure 3.3. For example, the slot-table inside node 2 denotes that slot 0 and slot 2 of link 3 are allocated to connection \( v_1 \) and slot 1 and slot 3 of link 2 are allocated to connection \( v_2 \). Inside each router, a slot counter reads information from the slot-table and updates the configuration of the crossbar at each slot. Slot counters of all the routers inside a NoC should update their values together according to a global time notion. With slot-table based configuration, there is no need to use head flits and thus does not require the allocated slots of a connection to be adjacent.

In the following, we will introduce some new techniques for TDM NoCs proposed by us.
3.2 Double time-wheel based dynamic connection setup

3.2.1 Problem description

Traditionally, dynamic distributed TDM channel allocation algorithms cannot be efficiently implemented in TDM based circuit switched NoCs because of the following reasons:

- During the connection setup process, messages such as setup requests, Ack/Nack need to be sent back to the source node to notify the failure or success of a setup request, and confirm/cancel the reserved channels. Traditionally, the delivery of these messages needs an auxiliary network [71, 53, 72], which adds up the costs and lowers the cost efficiency of a TDM NoC.

- The routing algorithms of the auxiliary network has to be deterministic [71, 53] to guarantee that setup, tear-down, and Ack/Nack messages of a connection are routed along the same path. In this way, the booked channels inside the routers along the path can be reserved/removed correctly. However, deterministic routing algorithms limit the exploration of path diversity. Setup requests routed by the deterministic routing algorithm are always traveled on fixed paths, resulting in low connection setup success probability.

- This auxiliary network is often a packet switched NoC [71, 53, 72]. It utilizes best effort packets to carry messages such as setup, release, Ack/Nack. These packets contend with each other, rendering the setup/tear down delay long and highly unpredictable.

Therefore, in Paper B, we propose a probe based connection setup method which can be used in TDM NoCs to eliminate the auxiliary network and overcome above-mentioned limitations. To support the probe based setup method, we developed a double time-wheel technique.
3.2. DOUBLE TIME-WHEEL BASED DYNAMIC CONNECTION SETUP

3.2.2 Probe based connection setup in TDM NoCs

Probe based setup method

Generally speaking, in order to support our probe based setup method, we add control signals request and ANS(answer) to a connection. The request signal is used to denote the setup, data transfer and release requests for a connection. The ANS signal is used to carry "Ack/Nack" messages. These messages notify the source node whether a setup endeavour has failed or succeeded.

![Figure 3.4: The overview of a router which supports the probe based connection setup method](image)

The router used to support our probe-based setup method is illustrated in Figure 3.4. Each link of the router has a 2-bit request signal which is in parallel with the data path, and a 2-bit ANS signal goes in the opposite direction of the data path. Note that, all wires of a link, including the data path, the request signal, the ANS signal are all shared by a number of channels in a TDM manner.

The setup process has two phases, namely, channel reservation phase and acknowledge phase. During the setup process, setup requests are carried by probes. When a probe arrives at a router, if the next slot of a desired output link is free, the probe will reserve this slot and use this slot to continue its movement toward the destination. Therefore, when a probe arrives at the destination, a connection has been successfully built up and then the ac-
knowledge phase begins. An "Ack" message will be sent back to the source node hop by hop, through the backward ANS signal of these reserved channels.

With our probe based setup method, the setup/release request and data flits of a connection can travel on the same TDM channels. This is possible because setup and release stages of a connection do not overlap with each other. This is due to 1) before a connection is established by a setup request, no data flits can appear on that connection, and 2) only after the transfer of data flits is finished, release request can then be sent out to release the slots of links reserved by the setup request.

Note that, if a probe fails to reserve a TDM channel in a router, a "Nack" message must be sent back by using the ANS wires of the reserved TDM channels. As the "Nack" messages travel backward, it can also release the reserved channels.

As a result, with the probe based setup method, the auxiliary network for setup is not needed any more. Since both forward and backward messages are transferred over the same TDM channels and along the same path, path search algorithm is no longer constrained to be deterministic.

However, backward "Ack/Nack" messages constitute a design challenge in our probe based setup method, since the "Ack/Nack" messages consist of only 1-2 bits, contain no address information, and need to be routed back following a given path. Moreover, the "ANS" wires used to deliver the "Ack/Nack" messages are also shared by different connections in a TDM manner, which mandates that the usage of correct slots to deliver the "Ack/Nack" messages of connections in order to avoid contentions. To resolve this challenge, we propose the double time-wheel technique.

**Double time-wheel TDM technique**

The way to route the backward messages (such as "Ack/Nack") of a connection is to rely on the slot table information inside each router along the path of the connection. As illustrated in Figure 3.5, the setup probe reserved time slots of a connection inside each router following the sequence $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$. The slot table of each router on the path records such information to configure its forward crossbar. Thus, if we can correctly read and interpret the information that reside in the reserved slot table cell, the backward crossbar can also be correctly configured to deliver backward messages. For example, if a backward message is sent out at slot 0 in router
3.2. **DOUBLE TIME-WHEEL BASED DYNAMIC CONNECTION SETUP**

5, by reading the information from the slot table to configure the backward crossbar, it can reach router 4 at slot 3. Following the same manner, it can reach router 3 at slot 2, and so forth.

To support this scheme, we propose a double orientation time-wheel technique. With this technique, we apply two slot counters inside each router: one is incremental counter and the other is decremental counter. The incremental slot counter reads the slot table and configures the forward crossbar for data flits and requests. The decremental counter configures the backward crossbar for "Ack/Nack". With the two slot counters, there are two slot access sequences inside a TDM NoC. For example, as Figure 3.5 depicts, the read sequence of a slot-table with an incremental counter is slot 0, 1, 2, 3, 0, 1, 2, 3..., while the sequence is slot 0, 3, 2, 1, 0, 3, 2, 1... with the decremental counter. Following this way, if the "Ack/Nack" message is sent out at the correct time slot, e.g. slot 0 from router 5, by using decremental slot counters to read the slot-table and configure the backward crossbar inside each router, it can reach router 4 at slot 3, reach router 3 at slot 2, and so forth. As a result, "Nack/Ack" messages are all delivered by slots of the ANS wires without contention and miss routing. The backward messages of a connection are naturally traveling on the same path that is reserved by the setup request.

An additional benefit of our double time-wheel technique is predictable

![Figure 3.5: Route back Ack/Nack with the double orientation time-wheel technique](image-url)
delays of all kind of messages. Firstly, all the messages travel at a guaranteed speed of one slot/hop since both forward slots and backward slots along the path of a connection are consecutive, according to the requirements of a TDM NoC. Secondly, for "Ack/Nack" messages, although they need to be buffered at a node for a while to ensure that they are sent out at the correct time slot, such buffering delay is also predictable. The maximum buffering delay is the size of a time window. Thirdly, the delay of a single setup attempt is predictable since all kinds of message delays are predictable.

In general, our probe-based double time-wheel TDM NoC can have the following advantages.

- When probe based connection setup method is utilized in TDM NoCs, auxiliary network for connection setup is no longer needed. Instead, the setup/release messages and data flits of a connection share the same TDM channels for traversal.

- The double time-wheel technique can route Ack/Nack messages which are required in the connection setup phase. With this technique, the Ack/Nack messages of a connection are carried by only a few bits of wires and always attached to the path reserved by the setup request.

- By combining the probe-based method together with double time-wheel technique, we can freely choose routing algorithms for the setup probes to explore the path diversity. Besides, in our design, the delivery of all kinds of messages of a connection have predictable delays.

3.2.3 New features in a TDM router

The details about the TDM router supporting our double time-wheel technique have been described in Paper B. We designed a slot-table based router with the double time-wheel technique that supports the probe based connection setup method. In the following, we emphasize the new features added to a TDM router.

ANS signal management

Ack/Nack messages must be sent out at the correct time slot. As Figure 3.6 illustrates, we designed an ANS table at each input port to buffer Ack/Nack messages of different connections until their correct backward time slots
### 3.2. DOUBLE TIME-WHEEL BASED DYNAMIC CONNECTION SETUP

<table>
<thead>
<tr>
<th>Slot table</th>
<th>Forward Cross Bar</th>
<th>Arbiter</th>
<th>Backward Cross bar</th>
<th>REG</th>
<th>ANS</th>
<th>Data/probe</th>
<th>Request</th>
</tr>
</thead>
</table>

#### Figure 3.6: Router architecture of the double time-wheel TDM NoC

Each ANS table just has one column, with the row number representing different slots of a time window. Thus, if a TDM time window contains $K$ slots, each ANS table should have $K$ rows. In our design, for differentiating different kinds of backward messages, a table cell is 2 bits wide. The ANS signal management principle works as follows.

- When an "Ack/Nack" message needs to send back to the source node to notify the success/failure of a connection setup, it is firstly written into the corresponding ANS table cell pointed by the incremental slot counter. The Ack/Nack message is created in the same slot at which the connection setup probe arrives.

- The read position of the ANS table is pointed by the decremental slot counter. After a cell is read, the buffered message is sent back through ANS wires. Then, the cell will be erased at the beginning of the next cycle.

With this ANS signal management principle, the maximum buffering time of a message is $K$ cycles, where $K$ is the total slot number in the time window.
window.

**Slot table**

Slot table plays a key role in our double time-wheel technique. Our double time-wheel technique requires that a slot table can be read by using both incremental slot sequence and decremental slot sequence simultaneously. It also requires that a slot table is able to handle written/erased requests on different table cells from both forward request signals and backward ANS signals at the same time.

As Figure 3.7 describes, rows in the slot table represent time slots, and columns denote output directions. During the slot reservation process, input port ID is written into a vacant table cell, whose row position is denoted by the increment slot counter and column position is denoted by the aiming output direction of the input probe. For example, the content of row 3 in Figure 3.7 means that at slot 3 the forward crossbar is configured in such a way: input port $i_1$ connects output port $o_0$, $i_2$ connects $o_1$. Meanwhile, it also denotes that for the backward crossbar, output $o_0$ connects input $i_1$, $o_1$ connects $i_2$. As Figure 3.7 suggests, when implemented in a mesh topology, the ID of each input port is encoded into 3 bits (suppose each router has 5 input/output ports). Thus, the slot table width equals 15 bits.

Both incremental slot counter and decrement slot counter simultaneously read a slot table, for the configurations of the forward crossbar and backward crossbar, respectively. To configure the multiplexers inside the forward crossbar, a row of table cells can be read to directly get a vector of configuration bits, with 3 bits per group. Each group configures a multiplexer. To configure the backward crossbar, the slot table’s row content needs to be translated into another format, in which columns denote input ports and each table cell denotes the output port, as denoted by Figure 3.7. We implement a combinational logic circuit to do such transformation.

In our implementation, a slot table functions like a dual-ported RAM. Both incremental slot counter and decremental slot counter can issue the read addresses. Write access on a slot-table is only issued by forward "setup" probes. However, erase accesses can be issued by both forward "release" signal and backward "Nack" signal.

Therefore, we have to enforce rules for slot-table access, to avoid collisions between different access requests. Our rules are as follows.
3.2. **DOUBLE TIME-WHEEL BASED DYNAMIC CONNECTION SETUP**

![Slot Table Diagram](image-url)

**Figure 3.7: The slot-table in a router**

1) Read access to a slot table returns a value within a delta delay (some combinational logic gates' delay), while write/erase access will be delayed until the very beginning of the next cycle (or the next slot). Thus, if read and write access to a cell happen at the same time, the read access gets the old value; if write is issued in the previous cycle and read happens in the current cycle, then read access gets the updated value.

2) We use a connection management mechanism to guarantee that write and erase accesses to the same table cell never happen. This mechanism states as follows.

1. For each connection, its forward "setup", "release", and backward "Nack" signals never collide. This is ensured by the non-overlapping phases in the connection setup and release. Therefore, write access and erase access to a table cell can never happen at the same time. Besides, this also guarantees that the erase access issued by forward "release" action of the connection never collides with the erase access issued by its backward "Nack" action.

2. Write accesses between different connections are also impossible to collide, since a slot table cell can only be exclusively reserved by one connection. Before the connection removes its reservation, no other connections can write or erase the table cell.
Flow control with double time-wheel technique

In TDM NoCs, the absence of contention ensures that no link-level flow control is required, but end-to-end flow control per connection is still required. For example, if the receiver's buffer is full, it needs to notify the sender to pause the data transfer. In previous works [25, 68, 8], TDM connections are bidirectional, one way from source to destination for data, and one way from destination to source to deliver end-to-end flow control messages.

However, bidirectional connections have several limitations:

- It imposes more constraints on the connection schedule/setup algorithms. Bidirectional connection setup requires a forward path as well as a backward path available. Thus, the connection setup algorithm suffers more limitations and thus has less probability to successfully build up such a connection.

- If a connection just has data flow on one direction, it is a waste of communication resources and bandwidth.

With our double time-wheel technique, connections can be unidirectional, the flow control messages can be carried by the backward ANS signal (1-2 bits) of a connection. Unidirectional connections can utilize the channel resources of a TDM NoC more flexibly and efficiently.

3.3 Highway in TDM NoC

3.3.1 Problem description

In TDM NoCs, quite often the utilizations of links are low. Because both idle and unallocated slots commonly exist in TDM NoCs.

Unallocated slots refers to the slots that are not reserved by any connection. The generation of unallocated slots is due to the two reasons:

- Successive links on the path of a connection require consecutive slots. Such mandatory sequence makes it difficult to utilize all the slots of links. For example, as illustrated in Figure 3.8, link 2 has one free slot, which is slot 0 (the TDM window size is 4 slots), and link 3 have two free slots, which are slot 0 and slot 2. Suppose that a connection v4 wants one slot on link 3 and one slot on link 2, although both link 3
and link 2 have free slots, they cannot be allocated to v4 since they are not consecutive.

- When mapping an application onto a TDM NoC, it is common that traffic flows are not uniformly distributed on all the links: some links have more bandwidth reservation requirements and some links less. Such unbalanced bandwidth requirements inside a network also cause slots of some links unallocated.

Figure 3.8: Illustration of slot utilization in TDM NoC

*Idle slots* refers to allocated but unused slots. Considering the practical traffic situations inside a network, it is unlikely that all the connections are busy all the time, especially for those statically scheduled connections. On one hand, idle TDM connections hold all the pre-reserved slots even if they have no data to deliver. On the other hand, busy TDM connections can just use a fixed number of reserved slots for data transfer, no matter how many data flits are buffering and waiting at the traffic sources.

However, is it necessary to enhance the performance of a TDM connection? As we know, TDM connections are used to offer guaranteed services. So, is it necessary to offer more service than guaranteed service?

In the streaming applications such as [73, 74] or H.264 decoding, each connection carries a streaming flow. In these applications, we might want
such a property that a lower bound streaming quality for a flow is always guaranteed, whereas enhanced quality can be offered when the system has free resources. To support such a property, we only need the NoC’s promise on the lower bound of communication bandwidth to offer guaranteed quality, whereas the upper bound of a traffic flow should be adjustable and adaptive to the traffic situations.

Besides, consider a computer system with a guaranteed connection between an L1 cache and L2 cache. Since cache misses are not completely predictable, often connection bandwidth is over allocated [68]. Thus, we might want a mechanism that can keep the guarantee on minimum throughput and on the predictable traffic flow, while offering additional non-guaranteed bandwidth to enhance the overall system performance by absorbing peaks of less predictable traffic flows.

Therefore, in some situations it would be a merit if a system can not only provide guarantee service, but also offer more and better service than the guaranteed whenever possible. From this perspective, we propose a novel highway technique. When applying this technique, a TDM connection can dynamically acquire both idle and unallocated time slots to enhance its throughput while still keeping the guarantee on lower bound of the bandwidth.

### 3.3.2 Highway technique for TDM NoCs

![Figure 3.9: Illustration of the function of a highway](image)

Our proposed highway can accelerate the data transfer of an established TDM connection by using unused time slots along the links of the connection.
3.3. **HIGHWAY IN TDM NOC**

A highway consists of one buffer queue per router, and along the given path of a TDM connection. We name the buffer queue as Highway Channel (HWC). With these HWCs, a connection can use both unallocated and idle slots of the output links on its path.

The function of a highway is illustrated in Figure 3.9. Both connections, A and B, span two routers by using slots of link 1 and link 2 in a consecutive manner. Connection A books slot 0 of link 1 and slot 1 of link 2; connection B books slot 2 of link 1 and slot 3 of link 2. Connection A also builds up a highway path by using one HWC in router 1 and one HWC in router 2, respectively. Thus, besides slot 0, connection A may also use slot 1 and 3, even slot 2 (if connection B has no data) of link 1. In contrast, without HWCs, connection B can only use slot 2 for data delivery.

In contrast to the free slot utilization methods proposed in [25, 66, 72], our highway technique can guarantee in-order data delivery. In [25, 66, 72], free slots are used to deliver best effort packets. However, the transfer delay of best effort packets are highly unpredictable. As a result, if some packets of a flow are delivered in the form of the best effort packets while the others are delivered by using TDM connections, the arrival order of packets at the destination node may be different from the sending order. In comparison, our highway technique can maintain the packet order of a data flow.

### 3.3.3 Router architecture overview

A router supporting the proposed highway technique is depicted in Figure 3.10. The input manager at each input link manages all the incoming flits of that input. Each input manager contains one or more HWCs. To support dynamic usage of the highway technique, each input/output link has a flag signal and a credit signal associated with the data path. The flag signal goes in parallel with the data path, while the credit signal is in the opposite direction. The 3-bit wide flag signal is used in highway setup, data transfer, and release. The 2-bit credit signal is used for Ack/Nack purpose in the highway setup phase and for credit-based flow control when in data transfer phase.

### 3.3.4 Highway setup process

As depicted in Figure 3.11, the procedure of setting up a highway for a 2-hop TDM connection which has 2 reserved slots in a time window of 4 is used as
an example to illustrate the dynamic highway setup process proposed by us. When a reserved slot (slot 0) is coming, the source node of an established TDM connection sends out a data flit carrying the flag signal "setup". This flit travels on the reserved slots of links, at a guaranteed speed of one slot per hop. If it can acquire an HWC in a router, its flag signal remains "setup". If the destination receives a flit with flag signal as "setup", "Ack" message will be sent back to the source node hop by hop, along the credit signal of each reserved HWC. When the source node receives "Ack", data transfer can use the established highway.

If the flit with flag "setup" fails to reserve a HWC inside a router, the setup process stops. The flit continues its traversal with the flag turning into "No HWC" (The default flag for a flit without highway is "No HWC"). At the same time, a "Nack" signal will be sent back to the source node hop by hop, along the credit signals of the already reserved HWCs along the path. As the "Nack" signal travels, it will release the reserved HWCs.

The proposed dynamic highway method has the following features:
3.3. **HIGHWAY IN TDM NOC**

- Highways can be dynamically established and released in a distributive way. When a connection needs to accelerate its data transfer, the source node of the connection can send out a "setup" flag to build up a highway by reserving HWCs in routers along the path of the connection. When a highway is no longer needed, the source node can release it by asserting a "release" flag.

- The highway setup process is guaranteed to be contention free. This is due to the fact that the "setup" flag is delivered by using pre-reserved TDM slots of links. As a result, this contention-free feature simplifies the design of HWC allocation logic since there is no need to do arbitration between setup requests.

- Our highway setup technique is applicable to many kinds of TDM NoCs. It basically has no architectural dependency. We impose no constraints on how connections are established and how crossbars are configured. Thus, this technique can be applied for both statically and dynamically built-up TDM connections. As long as a TDM connection...
delivers a flit to the destination, and the flag of the flit remains as "setup", a highway has been established.

- The highway setup process has no influence on the normal TDM data transfer, since the setup requests are carried by flag signals, and do not affect the data path.

- The setup time of a highway is predictable. This is due to the following reasons. Firstly, a "setup" flag travels at a guaranteed speed of one slot per hop toward the destination, since it is delivered on an established TDM connection with consecutive slots. Secondly, the backward "Ack/Nack" signal also travels at a constant speed per hop, since it is delivered by using the "credit" wires of the reserved HWCs and all these HWCs are allocated exclusive to a connection. Assuming each slot is one cycle and the propagating speed of "Ack/Nack" signal is also one cycle per hop and D is the distance between source and destination, then the total delay of a setup attempt is at most $2D + 2$ cycles. The $2D$ cycles are the time spent on the traversal of the forward "setup" flag and backward "ACK" signal. The additional $2$ cycles are regarded as the overhead time spent at network interfaces.

### 3.3.5 Data transfer with highway

After a highway is built up, accelerated data transfer can be launched. As described in Figure 3.12, when a data flit arrives at a router, the first step is to judge whether the incoming flit has a booked HWC or not. If the flit has no reserved HWC, the incoming flit must be delivered by using pre-reserved slots. Due to the consecutive property of the reserved slots, the incoming flit should be directly forwarded to the downstream node at the next slot.

However, as Figure 3.12 suggests, if the flit has an HWC, then the flit will be dispatched to that HWC first. When entering into an HWC, if the First-In-First-Out (FIFO) buffer queue is non-empty, the flit is pushed into the FIFO. However, if the FIFO is empty and if its requested output link is granted by arbitration, the incoming flit can be directly forwarded by using the bypass way, without being buffered into the FIFO. Otherwise, it still needs to be buffered. The flit arbitration rule is described in the next section.
3.3. HIGHWAY IN TDM NOC

Flit arbitration rule

During the data transfer phase, when an output slot is requested by more than one HWC, arbitration is needed. As suggested by Figure 3.10, each input link of a router has an input manager, and each input manager can have one or several HWCs. Thus, there are two stages of arbitration accordingly: a first stage arbitration between all the HWCs of an input manager, and a second stage arbitration between the input managers, as described in Figure 3.13.

Our arbitration mechanism needs to prioritize flits of a connection which are delivered on the pre-reserved slots of the connection. Only in this way, the lower bound of the bandwidth of a connection can be guaranteed. We use priority based round-robin to implement this scheme.

However, unlike the standard priority based round-robin arbiter, our arbiter design can take advantage of the following properties to simplify its
CHAPTER 3. TIME DIVISION MULTIPLEXING

Figure 3.13: The structure of the two-stage arbitrator

circuit logic.

- For the first stage arbitration at each input, at one time slot there is at most one request claiming that it has reserved the current slot.

- For the second stage arbitration for each output link, at one time there is also at most one request claiming that the current slot is reserved by it.

Due to the above two properties, we can simplify the priority based arbiter design for both stages. Our design just adds 2-3 gate-level delay to a classic no-priority round-robin arbiter.

The implementation of the two-stage arbitration is abstracted in Figure 3.13. An HWC asserts its request signal when it has flits to deliver. Meanwhile, if the HWC meets one of its reserved slots, it will also assert the reserved slot signal. Besides, if the input flit has a flag "No HWC" and no other HWC asserts its reserved slot signal\(^1\), the input port can assert the reserved slot signal. The asserted reserved slot signal of the first stage will be propagated to the second stage.

Both stages of arbitration prioritize the request with an asserted reserved slot signal. If no one asserts the reserved slot signal, it will use round-robin rule to decide a winner from all the requesters.

\(^1\)This actually checks whether the incoming flit has an unreleased HWC or not. Refer to Paper C for more details
3.4 Future work

Both our double time-wheel technique and highway techniques are helpful to TDM NoCs. The double time-wheel technique enables two-way communication on a TDM connection with a low cost. The highway techniques can enhance the performance of a TDM NoC without disordering data flows or introducing additional communication overhead. In the future, we plan to explore the two techniques in the following directions:

- **Apply the double time-wheel technique for advanced control purpose on the connection setup process.** For example, the backward ANS signals can be used to feed back the congestion information about the network to source nodes. Based on the congestion information, a source node can make better decisions on when to send out the new connection setup requests, and how often to retry the failed ones. In this way, we can try to lower down the contention probability between setup requests and enhance the success probability of each connection setup endeavor.

- **Apply the double time-wheel technique for error correction purpose.** When soft errors [75, 76] occur in a data transfer process and are detected by a destination node, the backward ANS signal path can be used to inform the source node about the error. Then, the source node can re-send the data to correct the errors.

- **Design a network interface that can offer better support to the TDM highway technique.** As a complete TDM NoC solution, network interface [77] is always indispensable. In order to support the TDM highway technique, the network interface also needs to add in a few new functionalities. For example, there should be a decision-making unit at the network interface to decide when to setup a highway for acceleration and when to cancel it.
Chapter 4
Spatial Division Multiplexing NoC

This chapter summarizes our research on Spatial Division Multiplexing (SDM) based circuit switched NoC. Particularly, we proposed a new allocator with maximal matching quality and strong fairness guarantee [Paper D]. It can be used for allocating channel resources in SDM NoCs. We also proposed a circuit switched NoC which supports multiple channels (SDM) and multiple networks. Based on this NoC, we explored the effects of several channel partitioning and configuration policies [Paper E]. Finally, we analyzed the respective strengths and weaknesses of circuit switched NoC and packet switched NoC [Paper F].

4.1 Introduction

4.1.1 An overview on SDM NoC

As the feature size in semiconductor technology scales down, more wires can be utilized between on-chip routers. Using all the wire resources between two routers as one wide communication channel becomes awkward and inadequate in many situations. Therefore, we may need to think about breaking down the wire resources into several narrower links, which is called sub-links, to offer more flexibility. One popular way of organizing these sub-links is called spatial division multiplexing (SDM) [78, 79, 80, 81, 82], which interconnects all the sub-links of a router with one crossbar.

The comparison between TDM NoC and SDM NoC is illustrated in Figure 4.1. With TDM technique, a single link is shared by multiple data flows
by dividing the time of the link into many fractions. Each data flow only uses a fraction of the time of the link in an alternating pattern. In comparison, SDM physically divides a link into sub-links. Each data flow can occupy one or several sub-links. All the flows are physically and spatially isolated.

4.1.2 Properties of SDM NoCs

The advantage of SDM NoC is that established connections are physically isolated. In other words, after a connection is established, the sub-links reserved by the connection are exclusively utilized. Thus, advanced end-to-end communication techniques such as source synchronized data transfer technique [83, 84, 85, 6, 86] can be utilized in SDM based circuit switched NoCs. With such techniques, data transfer can suffer less clock uncertainties and thus benefit from a higher clock frequency.

1With source synchronous data transfer technique, the sender sends a clock together with the data. Since the clock and data travel along the same path, they experience the
4.2. AN ALLOCATOR FOR CHANNEL ALLOCATION

The disadvantage of SDM NoCs is that, as the number of sub-links increases, the area spent on the crossbar of a router increases dramatically. Moreover, if dynamic connection setup method is applied, each router needs an allocator for sub-link allocation. However, both the area and the critical timing path of the allocator increases significantly, when the number of sub-links grows up.

4.2 An allocator for channel allocation

In this section, we present a new allocator designed by us. This allocator can make an allocation decision within one clock cycle. It also guarantees maximal matching quality and strong fairness.

4.2.1 Background and problem description

On-chip communication networks often use allocators for resource allocation purpose. For example, if we want to dynamically set up connections in a distributive way, we have to allocate output channels in each router individually according to the setup requests [87]. Besides, in virtual channel based packet switched NoCs, an allocator is also needed inside each router to allocate virtual channels for packets during run time [3].

We illustrate the allocation problem in Figure 4.2. Suppose two channel setup requests arrive at a router simultaneously and each of them demands an output channel from East output direction. If we define the setup requests as requesters and the output channels of as resources, an allocation problem is about how to allocate resources to their requesters.

Figure 4.2: Illustration of a channel allocation problem in NoC

similar delay and jitter. At the receiver side, data is re-sampled with the incoming clock.
Allocator inside a system is responsible for solving such allocation problems. It performs a matching between resources and the requesters. A matching is a distribution of resources to requesters satisfying the following three rules [88]:

- Only if the corresponding request exists, a resource can be granted to the requester.

- Each resource is at most granted to one requester.

- Each requester is at most granted once.

Two criteria [89, 88] are often used to evaluate an allocator. One is matching quality, and the other is fairness.

Matching quality refers to how well resources can match the needs of the requesters. It can be classified into Maximum matching and Maximal matching (refer to Paper D and [88] for detailed definitions). Maximum matching is often too costly to be realized in hardware. However, maximal matching is achievable. Maximal matching refers to that the resources are distributed in such a way that no additional requests can be served without removing existing grants.

Fairness [89] can be classified into strong fairness and weak fairness [90, 91]. Intuitively, fairness is about how requesters are served in proportion to their relative request rates. In practice, strong fairness often means that persistently active requesters are served in a periodic sequence equally often, while weak fairness only guarantees that each active requester is eventually granted, without any guarantee on the service behavior and the service rate.

Allocators designed for NoC confront many constraints. Firstly, since they are hardware based, the allocation algorithms have to compromise for hardware cost. Secondly, the allocator is desirable to work out an allocation decision within one clock cycle [89, 3]. This is due to the performance of a NoC is very sensitive to the delay in each router.

Because of these constraints, allocators used in NoC often suffer from drawbacks in either matching quality or fairness. For example, on one hand, the separable-input-first (SIF) and separable-output-first (SOF) [3, 92, 89] allocators utilize two-stage round-robin arbitrations to ensure strong fairness. However, they cannot guarantee the maximal matching quality. On the other hand, the wave-front (WVF) [93, 94, 95, 92], rectilinear-propagation-arbiter
(RPA) and diagonal-propagation-arbiter (DPA) [96] allocators provide maximal matching quality. However, they have no or only weak fairness guarantees.

Although other kinds of allocators, such as SPAA [97], iSlip [98], D2DDR [99], provide both maximal matching quality and strong fairness, they take too many clock cycles to work out an allocation, since these allocation algorithms need several iterations to optimize their allocation decisions. As a result, it is infeasible to use them in NoC environment.

For general matching problems, no existing solution is known to meet the constraints on timing while overcoming the shortcomings on matching quality and fairness.

4.2.2 A fair and maximal allocator for HRA

By taking a close look at the allocation problems in NoCs, frequently we encounter a special kind of allocation problems, which is named as homogeneous resource allocation (HRA) problem. For this kind of problems, we propose a single-cycle allocator to guarantee both maximal matching and strong fairness.

HRA problems in NoCs

The term homogeneous resources in our definition refers to a class of resources that have the same functionality and can be used interchangeably. homogeneous resource allocation problem obeys two more rules besides the three rules aforementioned in section 4.2.1:

- For each requester, all desired resources should belong to the same class;
- Any resource in a class can be granted to the requester which asserts requests on this class.

As Figure 4.3 suggests, HRA problems commonly exist in the systems which can be modeled by a multi-queue and multi-server model. In such a system, the allocation of servers to serve input queues constitutes an HRA problem if all servers are identical and can be used interchangeably. In this model, the matching quality of the allocator describes how well idle servers can discover and serve non-empty queues efficiently and without conflicts.
CHAPTER 4. SPATIAL DIVISION MULTIPLEXING NOC

Homogeneous allocation

Figure 4.3: Illustration for homogeneous resource allocation problem

The fairness of the allocator affects the service sequence and service frequency on input queues. This example suggests that the HRA problem revealed by us exists in different kinds of NoCs as well as in other on-chip applications, as long as they can obey the multi-queue and multi-server model. In NoC design practices, HRA may exist in circuit-switched NoC, for example, SDM based channel allocation; or in packet switched NoC, such as virtual channel allocation.

We can use a matrix with rows representing requesters and columns representing resources to express a homogeneous resource allocation problem. For example, as illustrated in Figure 4.4, input channels 0, 1, 3 (marked as ch0, ch1, ch3 in the figure) each requests a channel from output direction 1. Input channel 2 (ch2) requests a channel from output direction 2. It is an HRA problem since 1) each input channel (requester) just has one desirable output direction (resource class), and 2) any channel (resource) of a requested output direction can satisfy the need of a requester.

An HRA problem has the properties that 1) the request matrix can be split into sub-matrices based on resource classes, and 2) each of the sub-matrices can be merged into a single column, as described in Figure 4.4. As a result, we can assign an allocator to each reduced sub-matrix to solve the allocation individually. In such a way, the complexity of the allocator design can be reduced.
4.2. AN ALLOCATOR FOR CHANNEL ALLOCATION

4.2.3 Single-cycle fair and maximal allocator

We proposed a single cycle fair and maximal allocator to solve each reduced sub-matrix. The allocator used to solve each reduced matrix contains two parts: the resource allocation logic part and the priority updating logic part.

Overview of the resource allocation logic

The resource allocation logic makes resource allocation decisions. As depicted in Figure 4.5a, a matrix-based structure is applied for making resource allocation decisions. We name it as "waterfall" (WTF) because it
finds the matching with the help of consecutive rows of arbitration cells. For an n-requester allocator, it requires n rows.

We use an example to illustrate how an allocation decision is made by using the WTF allocation logic. Suppose the allocation logic is used to solve the reduced request matrix at the output direction 1 in Figure 4.4. Since it has 2 resources and 4 requesters, the allocator is also composed of 2 columns and 4 rows of arbitration cells. 3 of the 4 requesters are active, marked by an "1" of the reduced matrix. The two small dots in Figure 4.5a denote the availability and grant decision of the two channels of output direction 1, respectively. In Figure 4.5a, the allocation starts from the top to the bottom, so that requester $r_0$ get channel 1, then requester $r_1$ get channel 2.

However, in order to guarantee the fairness, we have to roll the allocation start and the end row at each round. Thus, the structure of the allocation logic is modified into Figure 4.5b. The allocation start row i can be selected by assert signal $p_i$. If row i is selected as the start row, due to the looped structure in Figure 4.5b, the end row becomes $(i + n - 1) \mod n$, where n is the total number of rows.

From Figure 4.5b, it seems that the allocation logic needs a looped structure to roll the start row and end row. However, we proposed a technique to make it loop-free, as illustrated in the left part of Figure 4.6. For an n-row allocator, the loop-free structure needs $2n - 1$ rows. The general idea is that
4.2. **AN ALLOCATOR FOR CHANNEL ALLOCATION**

using the bottom \( n - 1 \) rows replicates the top \( n - 1 \) rows, from row 0 to row \( n - 2 \). In this way, rolling of the start and end row is equivalent to selecting an active area. For example, if an allocation starts in a looped structure starts from row 1 and ends at row 0, it is equivalent to activate the area in the loop-free structure from row 1 to row 5, since row 5 replicates row 0, as illustrated in the left part of Figure 4.6. The right part of Figure 4.6 depicts the detailed circuits of the loop-free structure. It is implemented entirely by combinational logic gates, which means that the entire allocation behavior takes no more than one clock cycle.

![Diagram](image_url)

**Figure 4.6:** The loop-free structure of the WTF allocator

**Massive round-robin fairness policy**

To guarantee the strong fairness, we proposed a Massive Round-Robin (MRR) policy. Briefly speaking, our MRR policy states that the last granted requester of the current round will become the end row (has the lowest priority) in the next round. The start row is acquired by incrementing the end row by 1. If no request is granted in the current round, the start row and the end row remain unchanged in the next round.
The traditional round-robin policy \[100, 101, 102\] operates on the principle that a request that was granted in the current round should have the lowest priority in the next round. However, such a fairness policy only works in the situation that no more than one requester granted in a round. In comparison, our MRR policy extends the round-robin policy and makes it suitable in the situation that multiple requesters can be granted in a round.

Suppose there are \(m\) available resources and \(n\) requesters (numbered from 0 to \(n-1\)). Given current grants \(g_i, 0 \leq i \leq n - 1\) for each requester \((g_i = 1\) means granted\), and the start row of round \(t\) is row \(k\). Then the detailed algorithm to find the start row \(k_{t+1}\) of round \(t+1\) is depicted in Algorithm 1.

**Algorithm 1** Priority updating of MRR policy in \(m \rightarrow n\) allocation  

**Require:**  
Exist unique \(k\), that \(k \in \{p_k = 1, k \in [0, n)\}\) in round \(t\)

**Ensure:**  
Find unique \(k_{t}, k \in [0, n)\), that \(p_{k_t} = 1\) in round \(t + 1\)  

for \(b = n - 1; b \geq 0; b = b - 1\) do  
    if \(g_{(b+k \mod n)} = 1\) then  
        \(k_{t} = (b + k) \mod n\)  
        return \((k_{t} + 1) \mod n\)  
    return \(k_{t} = k\)

We have implemented the MRR policy on hardware with a loop-free architecture to control the selection of the start row and the end row of each allocation round.

For more detailed description of the allocator’s architecture, the proof and evaluation of the fairness mechanism and the performance, please refer to our Paper D.

### 4.3 Sub-networks and sub-channels

#### 4.3.1 Problem description

We need to think about three questions when splitting a link into multiple physical sub-links. (For convenience, in the following discussions, we use the term *sub-link* and *channel* interchangeably.)
4.3. SUB-NETWORKS AND SUB-CHANNELS

1. What would be an optimal number of sub-links: as many as possible, or there are certain limitations?

2. What is the optimal way of organizing the physical sub-links (channels) of a router? One possible way is to use a crossbar to interconnect all the channels. This method is called spatial division multiplexing (SDM). Another possible way is assigning the channels to different sub-networks.

3. How to dynamically set up a connection which needs multiple sub-links (channels) to satisfy its bandwidth requirement.

4.3.2 Circuit switched NoC with multiple channels and multiple networks

Overview

We propose a multi-channel and multi-network circuit switched NoC (MultiCS). It combines spatial division multiplexing [78, 79], which we call sub-channels, with sub-networks, as described in Figure 4.7. Sub-channels divide the wires between two nodes that then can be allocated separately and independently. Sub-networks are independent parallel physical networks that connect to the same nodes of a network [103, 104]. A connection between two nodes can utilize one or several sub-channels, and span one or several sub-networks.

In Figure 4.7, we use switch block diagrams [105] to reveal the differences between sub-channel organization and sub-network organization. With sub-channel organization, data from an input channel of a router can be switched to any channel of the desired output direction. However, with the sub-network organization, data from an input channel can only be switched to the output channels of the sub-network that the input channel belongs to. We can observe that organizing the same number of channels into sub-channels offers better switching flexibility [105] than organizing them into sub-networks, at the cost of a more complex crossbar and channel allocator.

Generally speaking, with the sub-channel organization, the critical timing path increases as the number of channels grows up. More specifically, the critical timing path for the control path scales up with $O(n)$, where $n$ is the number of channels. The critical timing path of the data path scales up with
CHAPTER 4. SPATIAL DIVISION MULTIPLEXING NOC

\[ O(\log(n)) \]. Besides, the area of both control path and data path scales with \( O(n^2) \), due to the scale up of the allocator [106] and the crossbar [82, 78].

With sub-network organization, the critical timing path is irrelevant with the number of channels, while the area increases linearly. Thus, compared with the sub-channel organization, although the sub-network organization is inferior in switching flexibility, it is superior in clock frequency and area cost.

**Dynamic connection Setup**

We introduce the parallel probe based method Paper A for connection setup in MultiCS. As a result, the dynamic connection setup no longer relies on
an auxiliary packet switched NoC which was commonly used in [81, 87, 46, 72, 80].

However, the parallel probing algorithm proposed in Paper A utilizes priority based arbitration method to do channel allocation. This allocation method is not suitable for a router with multiple channels per direction, since the circuit logic used for priority comparison becomes slow and cumbersome as the number of channels increases. Thus, we utilize the allocator introduced in Paper D for channel allocation purpose. Besides, for live-lock avoidance purpose, since the priority preemption policy in Paper A can no longer be used, we adopt a method which randomizes the retry interval for failed probes as a replacement.

Set up connections with multiple channels

We proposed two schemes for setting up connections that have a width requirement of more than one channel width. For example, suppose a connection has the width requirement of 4 bytes, while the width of each channel is 2 bytes, then the connection should consist of 2 one-channel connections.

The first scheme is called Deterministic Channel Allocation (DCA). DCA imposes a mandatory requirement on the connection width. For example, if a connection has a width requirement of 4 bytes and the width of each channel is 2 bytes, then the connections must consist of 2 one-channel connections; if the width of each channel is 1 byte, the it should be composed of 4 one-channel connections; any allocation below or above this figure is unacceptable.

The second scheme is called Adaptive Channel Allocation (ACA): ACA scheme has no hard connection width requirement. During the setup phase, a setup request tries to build as many one-connections as possible from the source to the destination as possible. However, the final connection width is determined by the number of successfully established one-channel connections, which depends on the run-time congestion situation of a network.

4.4 Circuit switching versus packet switching

4.4.1 Problem description

Traditionally, circuit switched NoC is considered as a method for offering Quality of Service (QoS) guarantee for communication, e.g. guaranteed
throughput or guaranteed delay. However, circuit switched NoC can also achieve high communication performance. Circuit switched NoC has, when compared to packet switched NoC, a longer setup time, but lower HW complexity and higher clock frequency [107, 108, 109, 110]. Thus, depending on packet size and throughput requirements, it may exhibit better or worse average performances.

4.4.2 Analysis and comparison of packet switched NoC and circuit switched NoC

Intuitions on packet switched NoC

Large packets are detrimental to packet switched NoC. It will incur unbalanced usage of links. As illustrated by Figure 4.8, intuitively, when each packet just contains one flit, the span of idle periods are just made up of a few cycles and equally distributed between output flow A and flow B. However, when packet size increases, the average span of idle periods grows wider and wider. Within a short period and with large packets, we may observe that output flow A is quite busy, while output flow B is almost idle. Such unbalanced usage of links is undesirable, since it is not good for the exploitation of the bandwidths of a network.

If the packet size is small, we can deal with the burstiness by increasing the buffer depth of a Virtual Channel (VC), and re-balance the traffic flows by adding more VCs [111]. With the same injection rate, large packets are detrimental to packet switched NoC since they increase the needs on both sides. The longer the packets, the more VCs and buffers are required to compensate the performance loss. Unfortunately, both VC and buffer are expensive, especially that adding virtual channels can lower the clock frequency. Thus, we can imagine that, if packets are long enough, they are almost impossible to be well handled by packet switched NoC with acceptable cost.

Intuitions to circuit switched NoC

On the other hand, circuit switched NoC prefers large packets. As we know, in order to deliver data with a circuit switched NoC, we need to have two phases.
4.4. CIRCUIT SWITCHING VERSUS PACKET SWITCHING

Figure 4.8: Unbalanced traffic caused by large packets in packeted switched NoC using round-robin arbitration

- Connection setup phase. A connection setup attempt is not guaranteed to succeed. To successfully set up a connection, it needs to retry failed setup attempts.

- Data transfer phase. Data transfer can be launched only when the connection has been established.

We can use a Markov model to describe the connection setup and data transfer phases of a circuit switched NoC, as shown in Figure 4.9. Suppose for each setup attempt, the failure probability is $\alpha$, and the average time spent on each setup attempt is $t_1$. Suppose further that the average data
transfer takes time $t_0$. Then, we have normalized throughput as

$$TH = \frac{t_0}{t_0 + \frac{t_1}{1-a}}$$

As the packet size grows, data transfer time $t_0$ increases accordingly. Therefore, we can observe that the normalized throughput of circuit switched NoC goes up as the packet size increases.

Thus, we may conjecture that, as the packet size increases, the performance curve of circuit switched NoC and performance curve of packet switched NoC may have a cross point, since one rises and the other falls.

**Architectural analysis of packet switched NoC and circuit switched NoC**

We compare the structure of packet switched NoC with circuit switched NoC. For a packet switched NoC, the critical timing path length $t_p$ depends on the virtual channel (VC) allocator. Even with the most advanced allocator design [112, 113], the latency of an allocator still scales up with $O(\log(mq))$, where $m$ is the number of output ports and $q$ is the number of virtual channels per port.

For a circuit switched NoC, assuming data path and control path can have different clocks (applying preferable source synchronized data transfer). For control path, its critical timing path length $t_c$ is made up of an m-port arbiter (suppose no SDM channel sharing), plus some additional combination logic circuit delay. For the critical timing path of data path $t_d$, it is mainly an m-port crossbar delay, which is about $O(\log(m))$.

Therefore, the control path length $t_c$ of a circuit switched NoC can be close to, or shorter than the critical timing path length $t_p$ of a packet switched NoC, while the data path $t_d$ of a circuit switched NoC is much smaller than $t_p$.

**Conclusions about the comparison**

We study the performances of circuit switched NoC and packet switched NoC by assigning practical values to $t_p, t_c, t_d$ getting from example designs. For packet switched NoC, we use a classical input-buffering virtual channel (VC) wormhole router architecture with a SIF VC allocator. For circuit
4.5. FUTURE WORK

In this chapter, we have introduced our works related to multi-channel circuit switched NoCs. The future work could be focused on the following aspects:

- In-depth optimization of the architecture of the water-fall allocator. Currently, the critical timing path of the water-fall allocator scales up with $O(n)$, where $n$ is the number of requesters. We will try to explore techniques which can reduce it to $O(\log(n))$.

- Research on the techniques that enhance the success probability of multi-channel connection setup. We may apply a certain control mechanism based on feedback information to avoid the collision between setup requests and thus enhance the success probability.

- Implementation and evaluation of mixed packet and circuit switched NoCs. We have evaluated the pros and cons of circuit switched NoC and packet switched NoC, respectively. We think that in real applications, combining them together may achieve an even better performance. It will be an interesting topic on how to efficiently design and utilize packet switched and circuit switched NoCs together.
Chapter 5

Summary

This chapter summarizes the thesis and points out the future directions.

5.1 Thesis summary

Compared with packet switched NoC, circuit switched NoC has advantages in QoS guarantee, separation of control path and data path, smaller cost on buffers, and sometimes a high operating clock frequency. However, in the past decade, circuit switched NoC is not studied as intensively as packet switched NoC. In this thesis, we introduce our researches on circuit switched NoCs. We have focused on dynamic connection setup, time division multiplexing, and spatial division multiplexing.

- Traditionally, distributed dynamic connection setup in circuit switched NoC tries to use routing algorithms designed for packet switched NoC to search paths for connections. However, this results in low setup success probability, since the path diversity is failed to be explored. We propose a parallel probing algorithm to explore the path diversity and enhance the connection setup success rate. The merits of our algorithm are that, on one hand, it tries to search as many as possible paths in parallel. One the other hand, it tries to reduce the overhead of a connection setup attempt by removing the redundant paths as soon as possible. We have efficiently implemented this algorithm in hardware.
• TDM technique has been widely used in circuit switched NoCs for link sharing. However, the usage of TDM technique makes the establishment of connection difficult and the link bandwidth utilization low. We propose a double time-wheel technique to enable the utilization of probe based setup method in dynamic TDM connection setup. With this technique, the difficulties on TDM connection setup can be overcome and the hardware cost can also be reduced. Besides, we proposed a highway technique to enhance the link utilization of TDM NoCs.

• SDM is another popular link sharing method for circuit switched NoC. To better support the dynamic connection setup in SDM NoCs, we propose a single cycle allocator that guarantees both maximal matching quality and strong fairness. We design a MultiCS NoC which supports both SDM and sub-network to organize channels. A parallel probe based connection setup method is implemented in this NoC. Based on this NoC, we have investigated which is the optimal way of organizing multiple channels. Moreover, we have made a comparison between circuit switched NoC and packet switched NoC.

5.2 Future directions

Based on our current research results and according to our current considerations, we can continue on the following directions.

• Extending the parallel probing algorithm. Currently, our parallel probing path search algorithm is designed for mesh topology. However, this algorithm can be utilized for other topologies as well. As long as multiple minimal paths exist between two nodes on a certain topology, our parallel probing algorithm benefits the path search and connection setup.

• Exploring fault tolerance techniques in circuit switched NoC. First, we can extend the parallel probing algorithm for fault tolerance purpose. The parallel probing algorithm can tolerate a certain magnitude of path failures, since it searches the entire possible minimal paths between a source and a destination. It can find a fault-free path as long as the path exists. Second, the double time-wheel technique can also be utilized in error correction. For example, if a transient fault happens on
a connection and causes a flit error, when the destination node detects the error, it can send a message through the backward path to notify the source node about the error. Then, the source node can resend the flit for error correction purpose.

- Providing multicast support in circuit switched NoC. Multicast communication is required by many practical applications, e.g., shared cache invalidation in a many-core system. In current stage, our work mainly focuses on the setup of unicast connections. However, it is possible to build up multicast connections by extending our current work. For example, with the parallel probing algorithm, a tree based multicast structure can be readily established with a single setup endeavour. However, in order to implement this feature, we need to solve tricky issues like contentions of setup requests, flow control on the multicast tree and so on.

- Exploring TDM techniques for asynchronous routers. In practice, quite often the NoCs are mesochronous, or even asynchronous. In this situation, keeping all the routers having a unanimous agreement on the slot time is not an easy task. There are tricky issues in slot counter resetting and updating [8, 114, 115, 116]. However, we might be able to relax the requirement on slot time synchronization. For example, instead of requiring a unanimous agreement on the global notion of time, we might just demand all the neighboring routers updating their slot time together. In such a way, we might be able to reduce the effort on slot time synchronization.

- Enhancing the connection setup success probability. Intuitively, if we spawn many setup requests at the same time or within a small period, the success probability of each setup request decreases because of the massive contentions between the setup requests. Therefore, if every source node can be aware of the contention situation of the network and schedule the setup requests to reduce contentions, the success probability of each setup attempt can be enhanced, setup delay can be reduced and throughput can be increased due to the reduction in the time spent on retrying failed setup requests.

- Application of the proposed techniques. In this thesis, we have showed our novel techniques such as parallel probing, double time-wheel, TDM-
highway, WTF allocator and so on. In the future, we will study which kind of real applications can be benefited from our techniques, and how to deploy our techniques in real applications.
Bibliography


