SPATIAL PARALLELISM IN THE ROUTERS OF ASYNCHRONOUS ON-CHIP NETWORKS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2011

By
Wei Song
School of Computer Science

Contents

List of Tables

List of Figures

List of Abbreviations

AD	asynchronous dispatching, 125
ANOC	asynchronous network-on-chip, 72
ASPIN	asynchronous scalable programmable interconnection network, 70
ATM	asynchronous transfer mode, 118
BE	best-effort, 62
CD	completion detection, 77
ChSlice	channel slicing, 89
CM	central module, 120
CMD	central module dispatcher, 132
CMP	chip multi-processor, 22
CMRICB	CM request forwarding crossbar in IM, 132
CMSCH	central module scheduler, 132
CRRD	concurrent round-robin dispatching, 123
DI	delay-insensitive, 30
DOR	dimensional order routing, 63
DVFS	dynamic voltage and frequency scaling, 68
EOF	end of a frame, 85
FF	filp-flop, 66
FIFO	first-in-first-out, 66
GALS	globally asynchronous and locally synchronous, 30
GS	guaranteed service, 61
HOL	head-of-line, 58
IM	input module, 120
IMD	input module dispatcher, 132
IMSCH	input module scheduler, 132
IP	input port, 120
IP	intellectual property, 51

IP	internet protocol, 118
IRG	input request generator, 132
LH	lookahead pipeline style, 89
LI	link from IM to CM, 120
LO	link from CM to OM, 120
MANGO	message-passing asynchronous network-on-chip providing guaranteed service through OCP interfaces, 73
MPEG	moving picture experts group — the name of a family of audio/video coding standards, 157
MPSoC	multi-processor system-on-chip, 22
MTBF	mean time between failures, 68
MUTEX	mutual exclusive, 43
NoC	network-on-chip, 22
OCP	open core protocol, 73
OM	output module, 120
OMRCCB	OM request forwarding crossbar in CM, 132
OMRICB	OM request forwarding crossbar in IM, 132
OMSCH	output module scheduler, 132
OP	output port, 120
PE	processing element, 51
PIM	parallel iterative matching, 123
QDI	quasi-delay-insensitive, 30
QNoC	quality-of-service NoC, 74
QoS	quality of service, 61
RNB	rearrangeable non-blocking, 121
RS latch	set and reset latch, 45
SDF	standard delay format, 89
SDM	spatial division multiplexing, 25
SI	speed-independent, 31
SNB	strict-non-blocking, 121
SoC	system-on-chip, 22
SPA	static priority arbiter, 46
SpiNNaker	universal spiking neural network architecture, 69
STG	signal transition graph, 31
TDM	time division multiplexing, 24

VC	virtual channel, 25
VLSI	very large scale integration, 23
XY	A DOR algorithm used in mesh or torus networks, first X then Y, 63

Abstract

A thesis submitted for the degree of Doctor of Philosophy
Title: Spatial Parallelism in the Routers of Asynchronous On-Chip Networks
By Wei Song, The University of Manchester, 6th July 2011

State-of-the-art multi-processor systems-on-chip use on-chip networks as their communication fabric. Although most on-chip networks are implemented synchronously, asynchronous on-chip networks have several advantages over their synchronous counterparts. Timing division multiplexing (TDM) flow control methods have been utilized in asynchronous on-chip networks extensively. The synchronization required by TDM leads to significant speed penalties. Compared with using TDM methods, spatial parallelism methods, such as the spatial division multiplexing (SDM) flow control method, achieve better network throughput with less area overhead.

This thesis proposes several techniques to increase spatial parallelism in the routers of asynchronous on-chip networks.

Channel slicing is a new pipeline structure that alleviates the speed penalty by removing the synchronization among bit-level data pipelines. It is also found out that the lookahead pipeline using early evaluated acknowledgement can be used in routers to further improve speed.

SDM is a new flow control method proposed for asynchronous on-chip networks. It improves network throughput without introducing synchronization among buffers of different frames, which is required by TDM methods. It is also found that the area overhead of SDM is smaller than the virtual channel (VC) flow control method – the most used TDM method. The major design problem of SDM is the area consuming crossbars. A novel 2-stage Clos switch structure is proposed to replace the crossbar in SDM routers, which significantly reduces the area overhead. This Clos switch is dynamically reconfigured by a new asynchronous Clos scheduler.

Several asynchronous SDM routers are implemented using these new techniques. An asynchronous VC router is also reproduced for comparison. Performance analyses show that the SDM routers outperform the VC router in throughput, area overhead and energy efficiency.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

The four years spent in Manchester is one of my most joyful, peaceful and successful periods in life. I must express my appreciation to all the people who have helped me. I shall and will remember all the kindness and do the same in the future.

I feel deeply fortunate to be supervised by Dr. Doug Edwards. He continuously encourages me to explore all my naive ideas and always expresses concerns in the most delicate way. The whole research starts from those ideas and it is his careful guidance that leads the research to the right direction.

I have to express my gratitude to Dr. José Nuñez-Yañez whose simulation model has ignited my research in on-chip networks. I also thank all the members in the joined project, especially to Dr. Sohini Dasgupta for her help on Peri Net and STG, and Atukem Nabina and Dr. Mohammad Hosseinabady for their kind treatment in Bristol.

It is impossible to finish this PhD without the help from the members in the APT group. Thanks to Dr. Luis Tarazona, Dr. Andrew Bardsley and Dr. Will Toms for their help on Balsa. Special gratitude to Dr. Charles Brej for his support on computer issues. Thanks to Jeffrey Pepper, Eustace Painkras, Dr. Luis Plana, Dr. Steve Temple and Dr. Simon Davidson for their maintenance of EDA tools and their kindness of allowing me to use the tools even during the tape out of SpiNNaker chips. Thanks to Dr. Lilian Janin for his continuous maintenance of Stella. Thanks Dr. Zhenyu Liu for his contribution to the synchronous Clos scheduler. Thanks to Hongguang Ren for proofreading parts of the thesis.

Many friends have helped me pass the time joyfully. Many thanks to Dr. Yebin Shi, Dr. Jian Wu, Dr. Shufan Yang, Zheng Xie, and Dr. Xin Jin who directly or indirectly contribute to the progress of this research.

Finally, I will never forget Prof. Suiming Fang who had opened my interest into hardware design. I am lucky to be the son of Prof. Caifa Song and Guixia Xia who are my mentors for life and academia.

The Author

Wei Song received his B.S.EE. from the College of Electronic Information and Control Engineering at the Beijing University of Technology, Beijing, P.R.China in 2005. In the same year, he was admitted through recommendation by the same college to pursue his M.S.EE. and obtained it in 2008.

From 2004 to 2006, Wei Song was also a research assistant in the Beijing Embedded System Key Lab (BESKL). He participated in the design of demodulators for several wireless communication systems including WLAN 802.11a/g, DVB-T and ATSC. He also implemented the FPGA verification platforms for most digital designs in BESKL. After leaving BESKL, he went back to his studying college and designed a real-time non-preemptive thread scheduler for a central communication controller in the hybrid electric vehicle control system. The communication controller was later patented in 2008.

Wei Song was offered a full scholarship by the EPSRC doctorate training program and began his Ph.D. study in the School of Computer Science at the University of Manchester in 2007 when he was also writing up his master dissertation. His work in Manchester is designing the asynchronous routers in an energy efficient network-on-chip for dynamically reconfigurable computing platforms, supported by EPSRC.

Part I

Introduction and Background

Chapter 1

Introduction

1.1 Motivation

The continuously shrinking transistor geometry makes network-on-chip (NoC) [9] the practical communication fabric for state-of-the-art multi-processor system-on-chip (MPSoC) designs. Following Moore's Law, the capacity and complexity of a chip has been boosted significantly in recent decades. The function of a board level system in the last decade can be integrated into one chip in modern system-on-chip (SoC) designs. On the other hand, SoCs are no longer built from scratch because the complexity is beyond control. The fast and reliable integration of numerous reusable intellectual property (IP) blocks becomes crucial to meet the time to market requirement. As a replacement for traditional hierarchical bus systems and point to point connections, the on-chip network infrastructure provides a unified interface for new IP blocks to be easily plugged into a system. A modern MPSoC is a communication-centric system [57] lying on an on-chip network communication fabric.

Most NoCs are synchronous networks where network components are driven by the same or several global clocks. Thanks to the timing assumptions allowed by the global clock and mature electronic design automation (EDA) tools, these synchronous NoCs are fast and area efficient. However, there are several design challenges in synchronous NoCs that are difficult to resolve:

- *Support for heterogeneous networks.* Unlike chip multi-processor (CMP) systems where every network node is a homogeneous processor element, an MPSoC is a heterogeneous system where network nodes are IP blocks with different

functions and hardware structures. These IP blocks are provided and tested with different clock frequencies, area sizes and even working voltages. These differences complicate the network topology, compromise the latency performance of synchronous networks and make chip timing closure difficult.

- *Low power consumption.* It is crucial to reduce the power consumption of an SoC as it determines the maximum standby time of a handset device. The clock tree of synchronous on-chip networks consumes a significant amount of energy [79] and it is getting worse along with the shrinking transistor geometry.

- *Tolerance to variation.* Process, temperature and voltage variations affect future sub-micron VLSI designs significantly [72, 74]. According to the international technology roadmap for semiconductors, the delay uncertainty caused by variations in the sign-off timing closure will reach 32% in 2024 [60]. Traditional static timing analysis is going to be replaced by statistical timing analysis methods [14] to cope with dropping yield rate and over-conservative timing estimation. Synchronous on-chip networks alleviate this effect by considering variations in their task mapping procedure [74]. However, this works only in homogeneous networks and the routers are still working at the worst speed estimated.

Instead of using synchronous on-chip networks, asynchronous on-chip networks are a promising solution to the above challenges. The communication components in an asynchronous on-chip network are built with clockless asynchronous circuits. Data are transmitted according to certain handshake protocols which can be insensitive to delay [117]. Because of this delay insensitivity, the interface between all IP blocks to the global asynchronous on-chip network is unified by the same synchronous to/from asynchronous interface. The fact that all synchronous blocks are isolated by the asynchronous network simplifies chip-level timing closure. Also, thanks to the delay insensitivity, an asynchronous on-chip network is naturally tolerant to all variations as the delay uncertainty caused by these variations cannot affect the function of those handshake protocols. Finally, since no clock is needed in asynchronous circuits, an asynchronous on-chip network consumes zero dynamic power when no data is in transmission.

However, most asynchronous networks [3, 45, 12, 7, 38] are slower than the synchronous on-chip networks with similar structures and resources [79]. Although the global clock in synchronous circuits is power consuming, it is a speed and area efficient approach to synchronize combinational operations. Asynchronous circuits rely

on handshake protocols to control data transmission. Combinational operations are explicitly detected and guarded to ensure the insensitivity to delay. The circuits used in detecting combinational operations introduce area and speed overhead. Delay insensitive asynchronous circuits are intrinsically slow.

Another issue is that, the state-of-the-art way of designing asynchronous on-chip networks is to asynchronously reproduce the structures of synchronous on-chip networks. As synchronous on-chip networks synchronize data with no speed penalty, timing division multiplexing (TDM) techniques [30] are extensively utilized. Simply reproducing such TDM structures in asynchronous on-chip networks introduces extra completion detection circuits and causes speed penalties.

Although the speed penalty of completion detection is unavoidable, as the promising advantages of asynchronous circuits are derived from those delay insensitive handshake protocols, the scale of the synchronization in asynchronous circuits can be limited to small transmission units, such as a single pipeline. The speed penalty is therefore alleviated. The following question is how to build asynchronous networks with such limited synchronization.

The solution presented in this thesis is *spatial parallelism*. TDM is not a good approach in asynchronous circuits because it brings extra synchronization and compromises speed. If the synchronization is constricted to a small scale such as a single pipeline, these pipelines are controlled distributedly. In other words, communication resources are spatially divided into unsynchronized low-level components and the speed penalty of synchronization is minimized.

1.2 Research objectives

The overall objective of this research is to explore the spatial parallelism in asynchronous on-chip routers. It is expected that 49% of the global signals will be driven by handshake protocols by 2024 and the latency of asynchronous signalling will be improved through 2014 [60]. Routers are the key components of an on-chip network. Improving the speed of asynchronous routers using spatial division techniques provides a feasible way of meeting the speed requirement for future chip designs and hopefully the techniques can be utilized in general asynchronous circuits beyond asynchronous on-chip networks.

Spatial parallelism will be explored in different layers. Although there is no consensus on the definition of layers in on-chip networks, the lower communication structure can be generally separated into three layers: *routing layer*, *switching layer* and *physical layer* [41]. The data transmitted in a network are divided accordingly as *frames*, *flits* and *phits*. The physical layer refers to the basic communication resources such as buffers and channels which deliver phits from one buffer to another. A flit comprises one or several phits. The *switching layer* dynamically allocates communication resources of the *physical layer* to different flits. The hardware structure and algorithm used in this allocation process is normally named as the *control flow* method. A frame is the smallest data unit which is self-explainable to a network node. It contains one or several flits. The *routing layer* determines the route through which a frame travels in a network. This research concentrates on exploring the spatial parallelism in the lowest two layers: the *physical layer* and the *switching layer*.

In the *physical layer*, the state-of-the-art routers use synchronized multi-bit pipelines as buffer stages, which are similar to the latches on buses in synchronous circuits. This pipeline style simplifies the control logic but introduces significant speed overhead. The effect of the speed degradation caused by synchronization will be analysed. Some techniques will be proposed to alleviate this degradation and will be compared with the synchronized pipeline style for speed, area and power performance.

In the *switching layer*, most asynchronous on-chip networks use timing division flow control methods such as the virtual channel (VC) flow control method. The new spatial division multiplexing (SDM) flow control method proposed in this research will be compared with the virtual channel flow control method in various router implementations. Their speed performance, area consumption and power dissipation will be analysed within different working environments.

1.3 Research contributions

The following contributions have been made upon this research:

- In the *physical layer*
  - Analysis of the speed and area overhead of synchronizing multiple low-level pipelines
  - *Channel slicing*, a technique that removes the synchronization among pipelines

  - A method of utilizing the *lookahead* pipeline style in normal asynchronous pipelines

- In the *switching layer*
  - Overhead analysis of the virtual channel (VC) flow control method
  - Overhead analysis of the *spatial division multiplexing (SDM)* flow control method
  - Utilization of SDM in asynchronous routers
  - Reducing the area overhead of SDM using Clos switches
  - A new *2-stage Clos switch* for on-chip routers
  - An *asynchronous dispatching* algorithm and its implementation to dynamically reconfigure Clos switches
- Overall
  - A novel asynchronous SDM router
  - Performance comparison among various wormhole, SDM and VC routers

1.4 Thesis organization

The thesis is divided into three parts: Part I provides a brief background introduction of this thesis. Part II proposes several new techniques to increase spatial parallelism in asynchronous routers. Finally a router is implemented in Part III utilizing all the techniques introduced in Part II.

In the rest of Part I, Chapter 2 presents an overview of asynchronous circuits including their different delay assumptions, handshake protocols, data encoding methods and arbitration components. Chapter 3 introduces the concepts related to on-chip networks and reviews previously published asynchronous router designs.

Part II proposes several new techniques in different layers. Chapter 4 concentrates on the *physical layer*. Channel slicing is utilized to remove the synchronization among low-level pipelines and the lookahead pipeline style is used to further reduce the period. Instead of using timing division flow control methods, Chapter 5 proposes the spatial division multiplexing (SDM) flow control method and examines its advantage over the virtual channel (VC) flow control method by behavioural level simulations. The

major implementation overhead of SDM is the enlarged crossbar. Chapter 6 provides a solution to the large area overhead — replacing the crossbar with a Clos switch. However, dynamically reconfiguring a multi-stage Clos switch is complicated and has not yet been implemented asynchronously. The first asynchronous Clos scheduler is designed and implemented also in Chapter 6.

Part III combines all the techniques in Part II into one router design. Chapter 7 briefly describes the final asynchronous SDM router. The performance analyses of several router implementations are provided in Chapter 8. The thesis is finally concluded in Chapter 9.

1.5 Publications

The following papers have been produced during the research of this work. The chapters that are closely related to these papers are identified respectively.

1. Wei Song, Doug Edwards, Zhenyu Liu and Sohini Dasgupta. Routing of asynchronous Clos networks. In submission to *IET Computers & Digital Techniques*, 2011.
   The hardware implementation and the performance evaluation of an asynchronous Clos scheduler in Chapter 6 come from this paper.

2. Wei Song and Doug Edwards. Asynchronous spatial division multiplexing router. *Microprocessors and Microsystems*, Vol. 35. No. 2, pp. 85–97, 2011 [115].
   The SDM router implementation in Chapter 5 originated from this paper.

3. Wei Song and Doug Edwards. Improving the throughput of asynchronous on-chip networks with SDM. In *Proc. of UK Electronics Forum*, pages 47 – 56, June 2010.

4. Wei Song and Doug Edwards. An asynchronous routing algorithm for Clos networks. In *Proc. of International Conference on Application of Concurrency to System Design*, pages 67-76, June 2010 [113].
   The asynchronous dispatching algorithm in Chapter 6 was first published in this paper.

5. Wei Song and Doug Edwards. A low latency wormhole router for asynchronous on-chip networks. In *Proc. of Asia and South Pacific Design Automation Conference*, pages 437443, January 2010 [114].

The area and speed performance of using channel slicing and lookahead pipelines in Chapter 4 was published in this paper.

6. Wei Song and Doug Edwards. Channel Slicing: a way to build fast routers for asynchronous NoCs. In *Proc. of UK Asynchronous Forum*, September 2009.

7. Wei Song and Doug Edwards. Building asynchronous routers with independent sub-channels. In *Proc. of International Symposium on System-on-Chip*, pages 48-51, October 2009 [112].
   The channel slicing technique introduced in Chapter 4 was first proposed in this paper.

8. Wei Song, Doug Edwards, Jose Nunez-Yanez, and Sohini Dasgupta. Adaptive stochastic routing in fault-tolerant on-chip networks. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 32-37, May 2009 [116].

9. Wei Song and Doug Edwards. A dynamic link allocation router. In *Proc. of UK Asynchronous Forum*, September 2008.

Chapter 2

Asynchronous Circuits

The asynchronous circuits in this thesis refer to the circuits where sequential components (registers and latches) are driven by handshakes rather than global clocks. The circuits using global clocks are, on the other hand, synchronous circuits. The necessary background knowledge for asynchronous circuits in asynchronous on-chip networks is introduced.

2.1 General description

Although asynchronous circuits have a long history of over 50 years [81], most VLSI circuits are synchronous due to the mature EDA support. Since registers and latches in synchronous circuits are synchronized by the global clock, they are the natural timing boundaries by which a circuit can be divided into paths. All these paths are driven by the same clock and operate concurrently and independently. EDA tools, especially synthesis tools, are therefore able to improve speed by optimizing these paths individually. On the other hand, the latches in asynchronous circuits are driven by handshake protocols (circuits). The operation of one latch is normally triggered by events generated from other latches. It is difficult to optimize the speed of asynchronous circuits due to the lack of clear timing boundaries to break large circuits into small analysable pieces as in synchronous circuits. Some asynchronous synthesis tools have been proposed recently, such as Petrify [29] and Balsa [43], to translate behavioural hardware descriptions into low level netlists. However, high speed asynchronous circuits are manually designed [110, 93, 105].

Shrinking transistor geometry brings opportunities for asynchronous circuits. As the number of transistors in a single die increases corresponding to the prediction of

Moore's Law, the area and power overhead of synchronizing the whole chip with one global clock is unacceptable and beyond the control of current EDA tools. Future MPSoCs should be globally asynchronous and locally synchronous (GALS) designs where synchronous IP blocks talk with each other using an asynchronous communication infrastructure. 49% of the global signals will be driven by asynchronous circuits by the year 2024 [60]. Variation is another problem. The decreasing transistor size increases power density which leads to temperature and power variation [59]. Process variation worsens the situation with non-deterministic cell latencies. The worst case timing analysis in synchronous circuits generates over-pessimistic speed estimation [14]. Asynchronous circuits are tolerant to variations and provide average speed performance.

2.2 Delay assumptions

Delay assumptions are the assumptions made for estimating the latency of circuit components. They are used by designers to analyse, simplify and implement asynchronous circuits. Different delay assumptions lead to circuits with different speed, area and robustness.

2.2.1 Delay-insensitive

Delay-insensitive (DI) assumes all the gates and wires in an asynchronous circuit have positive, undetermined and unbounded delays. With this assumption, every operation is forced to indicate its completion to allow the following operation to be processed. DI circuits are the most robust because logic function is independent to delay. However, the assumption itself constricts its usage in practical implementations. Nearly all basic gates in synchronous circuits, such as AND, OR, XOR, etc., are not delay-insensitive. Only C-elements (see Section 2.4.1 and Appendix A) and inverters can be used in delay-insensitive circuits [76].

2.2.2 Quasi-delay-insensitive

Quasi-delay-insensitive (QDI) circuits relax the delay assumption of DI by allowing isochronic forks [117], the sinks of which have the same delay from their common driver. This timing assumption allows a signal to be safely sent to multiple sub-circuits

and indicated by a common acknowledge signal. QDI can be used to implement practical designs.

2.2.3 Speed-independent

Speed-independent (SI) assumes all wires in a circuit have zero delay while all gates have positive, undetermined and unbounded delays. This may look unrealistic as wires have positive delays, but the delay of a wire can be counted in the delay of the gate driving it. In other words, SI assumes all forks in a circuit are isochronic. Since QDI assumes some, but not all, forks are isochronic, SI is a relaxed assumption from QDI to circuit designers. As presented in [25], it is possible to synthesize behavioural circuit models described in signal transition graphs (STGs) into SI circuits.

It is not necessary in most situations for a circuit designer to differentiate the non-isochronic forks from all wires. QDI is more robust than SI but they are usually discussed without clear differentiation. In this thesis, "QDI" is used for both QDI and SI circuits.

2.2.4 Relaxed QDI

DI, QDI and SI circuits assume gate delays are unbounded. As operations explicitly indicate their completion, some completion detection circuits are introduced when multiple operations are synchronized. These completion detection circuits lead to extra speed and area overhead. In addition, the delays of gates in practical circuits are bounded. Some delay relations between certain paths can be utilized to reduce the area and speed overhead introduced by synchronization. In these relations, the gate delay is still unbounded but some paths are assumed longer than other paths [131, 121, 110, 111]. When multiple paths are synchronized, only the operations through the longer paths are indicated leaving the shorter paths undetected. As long as the assumed delay relations are not violated, these circuits are still tolerant to variations as DI, QDI and SI. Currently there is no common name for these circuit styles and it is called "relaxed QDI" in this thesis.

2.2.5 Self-timed

Although self-timed circuits are the superset of all asynchronous circuits, they are usually referred by their narrow meaning of circuits that assume gates and wires have

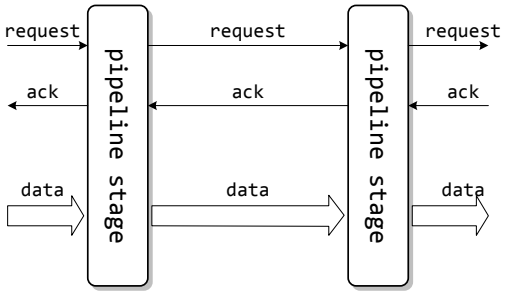


Figure 2.1: Two adjacent pipeline stages

bounded delays. Worst delay analysis and matched delay lines are used in self-timed circuits to avoid completion detection. This circuit style is area and speed efficient. However, the bounded delay assumption compromises the tolerance to delay variations; therefore, self-timed circuits require careful timing analysis and implementation to ensure that the estimated bounded delay assumption is not violated.

2.3 Handshake protocols

Figure 2.1 shows the abstract view of asynchronous pipeline stages. Every pipeline stage is a storage component. The data from the previous pipeline stage are ready when a valid *request* is received. When incoming data are safely captured, the previous pipeline stage is notified through the *ack* line and the old data stored in the previous stage can be released. A handshake protocol controls the transitions between two adjacent pipeline stages. Specifically, it determines the waveform and timing on *request* and *ack* lines. Two protocols are available in asynchronous circuits: 4-phase and 2-phase.

2.3.1 4-phase

The 4-phase protocol (also called return-to-zero signalling or level signalling) is the most utilized handshake protocol in asynchronous circuits as all gates are level triggered. Standard cells in synchronous circuits are also level triggered. Asynchronous circuits complying with the 4-phase protocol can be implemented using synchronous standard cell libraries.

Figure 2.2 illustrates the waveform of a pipeline stage using the 4-phase protocol. Positive *request* denotes the readiness of data and positive *ack* denotes the incoming data are captured. Data are subjected to change when *request* is low. The pipeline stage is ready for new data when *ack* is low. As shown in Figure 2.2, data must remain

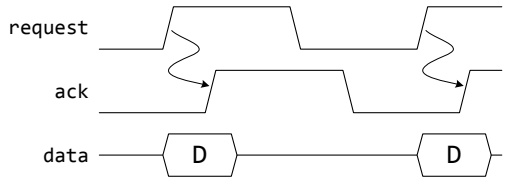


Figure 2.2: 4-phase handshake protocol

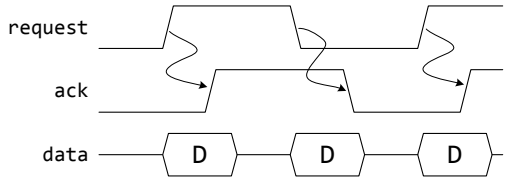


Figure 2.3: 2-phase handshake protocol

stable during the period between the positive edge of *request* and the positive edge of *ack*. In some implementations, data are required to remain stable until the negative edge of *request* according to different types of storage circuits.

2.3.2 2-phase

As described by the name, the 2-phase protocol (also called non-return-to-zero signalling or transition signalling) contains only two phases: a transition on *request* and a transition on *ack*. Figure 2.3 depicts the waveform of a 2-phase pipeline. When there is a transition on *request*, no matter positive or negative, new data are ready. In the same way, the capture of data is indicated by a transition on *ack*. Data must remain stable after the transition of *request* until the transition of *ack*. As *request* can transit immediately after *ack*, pipeline stages are always ready for new data.

Theoretically 2-phase is better than 4-phase because a set of data is delivered in two phases instead of four phases. The 2-phase protocol is fast and energy efficient. However, it has some implementation problems: The storage components in 2-phase pipeline stages are normally transistor level designs [122] which cannot be replaced with synchronous standard cells. If the combinational circuits in a 2-phase pipeline comply with the QDI delay assumption, they need to be re-designed in the same way as the storage components, which is complex and area consuming. Fault tolerance is another problem. In 4-phase pipelines, storage components are active and vulnerable to transient faults 50% of time in the worst case. On the other hand, the storage components of 2-phase pipelines are always ready for new data leading to 100% time

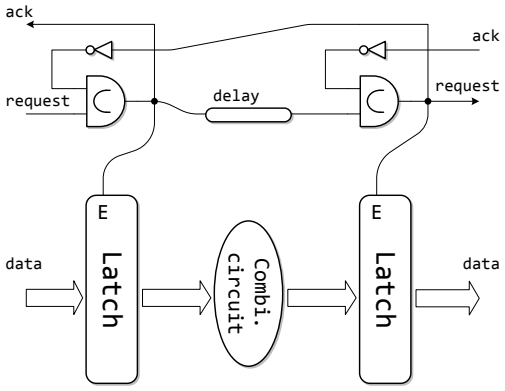


Figure 2.4: Bundled-data pipline

vulnerability. 4-phase pipelines are safer than 2-phase pipeline in the aspect of tolerance to transient faults [103].

2.4 Data encoding

Data are encoded in different formats, such as binary code and one-hot code, in synchronous circuits. Similarly, asynchronous circuits have various data encoding methods. Data encoding is related to different timing assumptions. It is also an important determinant to the area, speed and energy efficiency of asynchronous pipelines.

2.4.1 Bundled-data

Bundled-data pipelines encode data in binary code. Figure 2.4 shows a possible implementation of bundled-data pipelines.

For an N-bit bundled-data pipeline, the wire count of *data* is N. The number of available symbols is 2^N and every symbol is valid. Thus the *data* bus cannot differentiate stable data from changing data. A separate control pipeline is added to identify the availability of data. In Figure 2.4, data latches are triggered by a Muller pipline. Because the control pipeline is always one bit wide, a bundled-data pipeline is also named a single-rail pipeline in some articles.

The Muller C-element is one of the most utilized primitives in asynchronous circuits. The symbol and the truth table of a 2-input C-element is shown in Figure 2.5 and Table 2.1 respectively. Its function is a combination of logical AND and latch. The output q updates when both input pins have the same value, otherwise the output value remains.

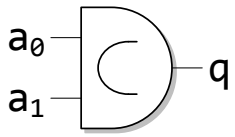


Figure 2.5: 2-input C-element

Table 2.1: Transition table of a 2-input C-element

a_0	a_1	q pre	q next
0	0	–	0
1	1	–	1
0	1	q	q
1	0	q	q

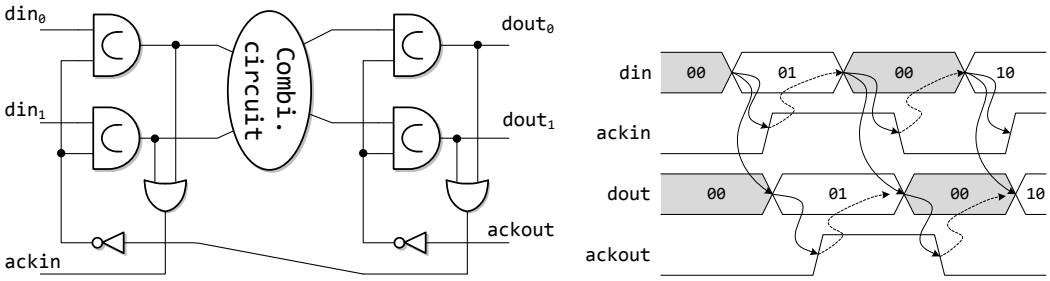
The storage components in Figure 2.4 are level triggered latches complying with the 4-phase handshake protocol. The C-elements in the control pipeline trigger data latches only when the data from the previous pipeline stage is valid (*request*+) and the next pipeline stage is ready for new data (*ack*−). The waveform of this bundled-data pipeline is already shown in Figure 2.2.

Since the control pipeline triggers latches without detecting the data bus, some delay gates are added on the *request* path to ensure that latch triggers always reach latches after data are stable. These delay gates are one of the major problems of implementing a bundled-data pipeline. To match the delay of data, the actual delay of the combinational circuit between latches is estimated and delay gates with equal delay are inserted. In other words, bundled-data pipelines assume delays are bounded, which violates the QDI delay assumption. In addition, the delay gates cannot be inserted automatically by EDA tools but are prone to be removed by them. Delay insertion introduces a heavy design burden.

It is possible to implement 2-phase bundled-data pipelines, namely a micropipeline which Ivan Sutherland first introduced [122]. A micropipeline can utilize the Muller pipeline as the trigger control logic but the data latches are double-edge triggered. As described in Section 2.3.2, these double-edge triggered latches are transistor level designs which are not available in pure standard cell design flows.

2.4.2 Multi-rail

Multi-rail pipelines represent a pipeline family including several data formats. They usually comply with the QDI delay assumption.



(a) Schematic (b) Waveform

Figure 2.6: 4-phase dual-rail pipeline

A bundled-data pipeline is not QDI for two reasons: One is that data buses cannot indicate the data availability as all symbols are valid. The other one is that the control pipeline triggers latches without detecting data. To meet with the QDI delay assumption, multi-rail pipelines use expanded symbol spaces where only a portion of symbols are valid. As long as the transform of any two valid symbols is connected by invalid symbols, data can indicate its availability and no control pipeline is needed.

Dual-rail

A dual-rail pipeline is the simplest multi-rail pipeline. Every dual-rail pipeline has two data wires: d_0 and d_1. Although two digits $\{d_1 d_0\}$ provide four available symbols, only $\{01\}$ and $\{10\}$ are used to represent data "0" and "1" respectively. The symbol $\{00\}$ represents idle or bubble denoting invalid data and the symbol $\{11\}$ is illegal. In this way, the data wires denote valid data and request at the same time.

Figure 2.6 depicts a 4-phase dual-rail pipeline and its waveform. In every pipeline stage, two C-elements are used as latches triggered by *ackout*. Initially all data wires are low indicating data are not valid and *ackin* is low indicating pipeline stages are ready for new data. When a valid symbol arrives at *din*, C-elements capture this symbol and deliver it to the next pipeline stage immediately through *dout*. An OR gate is always monitoring *dout*. When the data symbol is captured, it is detected by the OR gate which sets *ackin* to high.

4-phase is also called return-to-zero because the request line is reset before another valid request. In a 4-phase dual-rail pipeline, this means every two adjacent valid symbols are intersected by an idle symbol. As shown in the waveform, when the next pipeline stage has captured the data, *ackout* (the *ackin* driving by the next stage) is driven high indicating that data can be released. When the previous pipeline stage receives an acknowledge through *ackin*, *din* is soon driven to the idle symbol $\{00\}$.

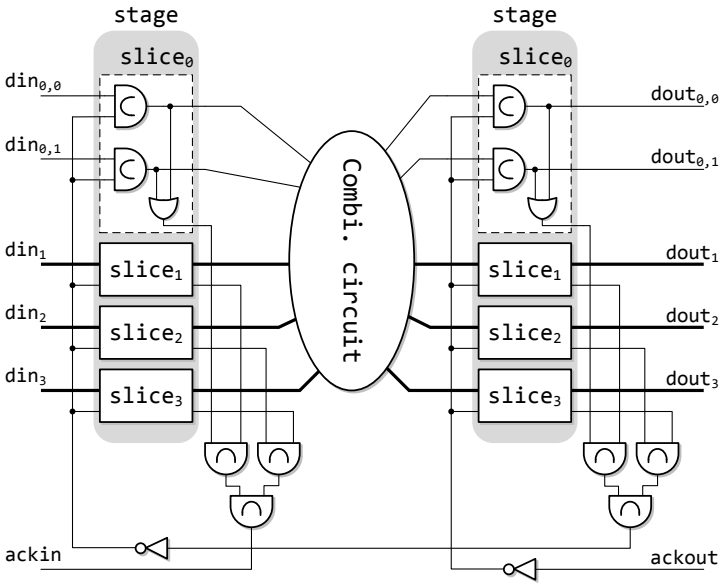


Figure 2.7: 4-bit 4-phase dual-rail pipeline

Consequently the data output *dout* of the current stage is released. The OR gate detects the idle symbol and resets *ackin* allowing the previous pipeline stage to capture new data. The idle symbol further releases the next pipeline stage which in turn resets *ackout*. A new data cycle thus starts.

The basic dual-rail pipeline shown in Figure 2.6a delivers only one bit of data in one cycle. To form a practical pipeline that can deliver multiple data bits simultaneously, multiple dual-rail pipelines are combined and synchronized into a wide one.

Figure 2.7 demonstrates a 4-bit dual-rail pipeline built by four 1-bit dual-rail pipelines. Data on this pipeline are divided into four slices and allocated to the four bit-level pipelines in bitwise order. The four bit-level pipelines (or slices) are synchronized by the C-element tree generating a common acknowledge signal (*ackin* and *ackout*). Therefore, an acknowledge is sent only when all the four bit-level pipelines have captured valid symbols and is, vice versa, reset only when all bit-level pipelines have released their data. The C-element tree along with the OR gates in every pipeline is the completion detection circuit in wide pipelines.

1-of-n

1-of-n pipelines encode data in one-hot code. A 1-of-n ($n > 1$) pipeline has n data wires providing 2^n available symbols but only n symbols are valid. In every valid symbol, one and only one data wire is high. Table 2.2 illustrates the translation between

Table 2.2: 1-of-4 code

	binary	1-of-4
0	00	0001
1	01	0010
2	10	0100
3	11	1000

1-of-4 symbols and binary symbols [3]. Because a 1-of-4 pipeline has four data wires, it provides four different data symbols and delivers two data bits in one cycle. In general, a 1-of-n pipeline delivers $log_2 n$ binary bits per cycle. The dual-rail pipeline is the smallest 1-of-n pipeline where $n = 2$. Similar to the 4-bit dual-rail pipeline in Figure 2.7, multiple 1-of-n pipelines can be combined into a wide pipeline.

Since the data on a 1-of-n pipeline are one-hot coded, only one data wire transits during one data cycle and the power consumption is nearly the same for different configurations of n. Increasing the number of data wires in a 1-of-n pipeline improves energy efficiency. However, a 1-of-n code is not an area efficient encoding method. Only n of 2^n symbols are utilized. The total area increases proportional to n, which can be unaffordable when n is large. The number of data wires, n, in practical asynchronous on-chip networks is usually fewer than five [3, 99].

m-of-n

An m-of-n pipeline represents a valid data symbol by driving m of the total n wires to high. The number of valid data symbol is $\binom{n}{m}$. A 1-of-n pipeline can be classified as a special m-of-n pipeline where $m = 1$ but normally m-of-n pipelines indicate $m > 1$. For a pipeline with n data wires, $\binom{n}{m}$ is larger than $\binom{n}{1}$ when $n \geq 4$ and $n-1 > m > 1$. Therefore, m-of-n pipelines can deliver more data than 1-of-n pipelines with the same wire count and they are more area efficient than 1-of-n pipelines with the same data width.

Table 2.3 shows an example of a 2-of-7 code [4]. The pipeline using this 2-of-7 code delivers four bits of data per cycle, which is equivalent to a wide pipeline built from four dual-rail pipelines or two 1-of-4 pipelines. The number of data wires used by dual-rail pipelines or 1-of-4 pipelines is eight while only seven wires are needed by the 2-of-7 pipeline. The 2-of-7 code is also promising in the aspect of power consumption. Four data wires transit per cycle in a wide dual-rail pipeline while only two data wires transit in a 2-of-7 pipeline. The total power is reduced by half.

Table 2.3: 2-of-7 code

	binary	2-of-7		binary	2-of-7
0	0000	000-0101	**8**	1000	010-0001
1	0001	000-0110	**9**	1001	010-0010
2	0010	000-1001	**10**	1010	010-0100
3	0011	000-1010	**11**	1011	010-1000
4	0100	001-0001	**12**	1100	100-0001
5	0101	001-0010	**13**	1101	100-0010
6	0110	001-0100	**14**	1110	100-0100
7	0111	001-1000	**15**	1111	100-1000

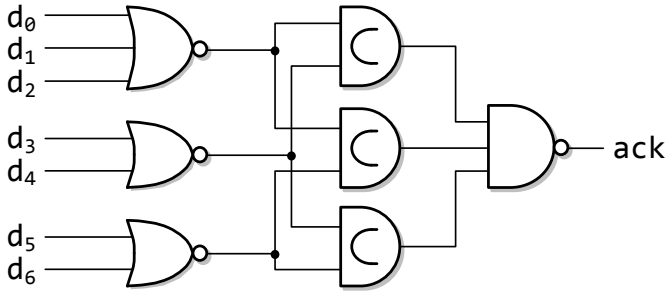


Figure 2.8: Completion detection circuit of a 2-of-7 pipeline stage

Although m-of-n pipelines are area and energy efficient, they are usually utilized to connect modules built by 1-of-n pipelines instead of to build combinational circuits directly [108]. One of the problems is the large area overhead of the completion detection circuits for m-of-n pipelines. Figure 2.8 depicts the completion detection circuit for the pipeline using the 2-of-7 code in Table 2.3. The area of this circuit is larger than that in a wide 1-of-4 pipeline with the same data width. Actually, all m-of-n codes need larger completion detection circuits than 1-of-n codes. The other problem is the area overhead of the code translation circuit. Conventional combinational circuits, especially those synchronous ones, use binary code. They can be directly transformed into circuits complying with dual-rail pipelines. Using the 1-of-4 code introduces moderate area overhead compared with dual-rail. However, it is difficult to do calculation in 2-of-7 or general m-of-n codes. Translating m-of-n to dual-rail leads to extra encoding and decoding circuits. Since the area reduction of using m-of-n codes in complex combinational circuits is normally smaller than the area overhead introduced by completion detection and code translation, m-of-n pipelines are used only in long distance communications in most circumstances.

2.5 Performance comparison of pipelines

In the previous sections, pipelines using different delay assumptions, handshake protocols and data encoding methods have been described. This section tries to provide a general performance comparison of all pipeline styles.

Pipeline delay

Pipeline delay is the forward latency of delivering a valid data symbol from one end of an empty pipeline to the other end. It represents the delay performance of the pipelines stages. Empty multi-rail pipeline stages capture data whenever they are ready. Bundled-data pipelines insert matched delay lines in the control pipeline to ensure that latch triggers are slower than data. As a result, bundled-data pipelines normally demonstrate longer pipeline delay than multi-rail pipelines. However, it is not true when complicated combinational circuits are implemented. The multi-rail pipelines, especially the m-of-n pipelines, introduce slow combinational circuits. When the extra delay caused by these slow combinational circuits is longer than the matched delay line, bundled-data pipelines show better speed performance.

Period / throughput

Sometimes the pipelines are over-fed. Incoming data cannot be delivered as pipeline stages are busy processing old data. In this situation, the speed performance of a pipeline is determined by the period instead of the pipeline delay. The reverse of the period is the maximum throughput. It is obvious that pipeline stages may have different periods. The stage with the maximum period determines the maximum throughput because other pipeline stages are always waiting for the slowest one when the pipeline is heavily loaded. The slowest loop path is the *critical cycle* of the pipeline.

For long distance serial pipelines, 2-phase pipelines are normally faster than 4-phase pipelines because no reset on request lines is needed. Bundled-data pipelines may suffer from the large data skew in long distance transmission.

For wide pipelines without complicated combinational circuits, bundled-data pipelines outperform QDI multi-rail pipelines. Although the matched delay line introduces extra latency, this overhead can be reduced if data wires are carefully routed. On the other hand, the completion detection circuits used to synchronize multiple bit-level QDI pipelines leads to significant delay.

If complicated combinational circuits are implemented between pipeline stages, the delay overhead of using a multi-rail pipeline can be large. Similar to pipeline delay, bundled-data pipelines normally show better throughput thanks to their fast combinational circuits.

Area consumption

Multi-rail pipelines introduce significant area overhead in building data pipelines and combinational circuits due to their inefficient data encoding method. Bundled-data pipelines utilize binary coding and the combinational circuits are the same as those in synchronous circuits. Multi-rail pipelines, on the other hand, need extra data latches in pipeline stages due to the expanded symbol space, require encoders and decoders when m-of-n codes are used, and lead to large combinational circuits.

For long distance serial pipelines, m-of-n pipelines consume less area than dual-rail or 1-of-n pipelines. To achieve a reasonable latency, buffers and inverters are inserted on the long wires to increase driving strength and reduce transition time. These buffers are the major area overhead. m-of-n codes reduce the wire count as well as the area. However, when complex functions are needed, the use m-of-n codes should be generally avoided because of their unaffordable area overhead in combinational circuits.

Power consumption

No clock is needed in asynchronous circuit styles therefore no dynamic power is consumed when the circuit is idle. The leakage power is proportional to the area. The key issue is the dynamic power consumed in active circuits.

Dynamic power is related to toggle rate — the number of transitions during in a certain period of time. Table 2.4 shows a simple estimation of the toggle rate of different asynchronous pipelines. In the table, D denotes the equivalent data width. The toggle rates of 4-phase QDI pipelines are two times that of 2-phase pipelines because all wires return to zero every cycle. Bundled-data pipelines represent data in binary code. On average 50% of data wires transit very cycle. Multi-rail pipelines represent data in 1-of-n or m-of-n codes. In each 4-phase multi-rail pipeline, one wire (dual-rail or 1-of-n) or m wires (m-of-n) must transit twice every cycle. All pipelines have some extra toggles introduced by the *ack* line and the *request* line (only in bundled-data pipelines).

Table 2.4: Average toggle rate

	4-phase	2-phase
bundled-data	$\frac{D}{2}+4$	$\frac{D}{2}+2$
dual-rail	$2D+2$	$D+1$
1-of-n	$\frac{2D}{\lfloor \log_2 n \rfloor}+2$	$\frac{D}{\lfloor \log_2 n \rfloor}+1$
m-of-n	$\frac{2mD}{\left\lfloor \log_2 \binom{n}{m} \right\rfloor}+2$	$\frac{mD}{\left\lfloor \log_2 \binom{n}{m} \right\rfloor}+1$

Based on the toggle rates in Table 2.4, we can draw some simple conclusions for asynchronous pipelines without complex combinational circuits:

- 2-phase pipelines consume less dynamic power than 4-phase pipelines.
- Bundled-data pipelines consume less power than dual-rail pipelines.
- 1-of-n pipelines consume less power than dual-rail pipelines.

The power consumption comparison between m-of-n pipelines and 1-of-n pipelines is hard to evaluate. In theory, the toggle rate of m-of-n codes is smaller than 1-of-n codes. On the other hand, m-of-n pipelines require complicated completion detection circuits which cause extra power consumption.

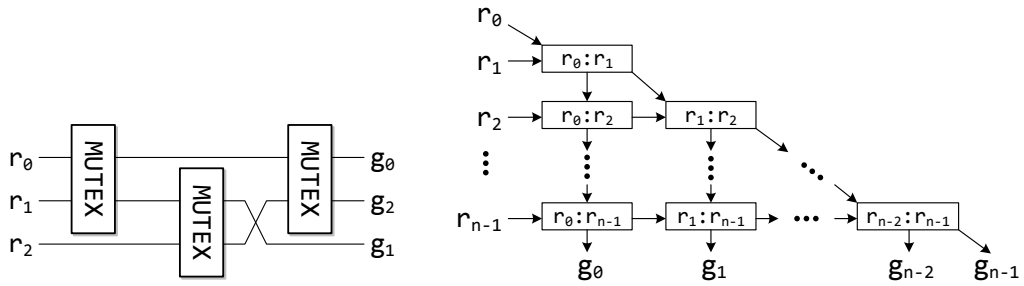
2.6 Arbiter

Arbiters are an important class of combinational components in asynchronous on-chip networks where arbitration occurs frequently. This section introduces the arbiters which have been used in state-of-the-art asynchronous on-chip networks.

2.6.1 Multi-way MUTEX arbiter

The multi-way MUTEX arbiter is fast and area efficient with a small number of requests. It is utilized extensively in asynchronous on-chip networks [38, 114]. it is also a basic component of complicated asynchronous allocators (Section 2.7).

Figure 2.9a depicts a 3-way MUTEX arbiter which grants one resource to one of the three requests. The 2-input MUTEX gate (Appendix A) is the basic arbitration logic arbitrating two requests. When one request is granted, the other one is blocked until



(a) 3-way MUTEX arbiter (b) N-way expansion

Figure 2.9: Multi-way MUTEX arbiter

the granted request is withdrawn. A MUTEX gate is also a 2-way MUTEX arbiter. In the 3-way MUTEX arbiter, every pair of requests is checked by a MUTEX gate. As a result, when one request is granted, the arbiter ensures that all other requests are blocked.

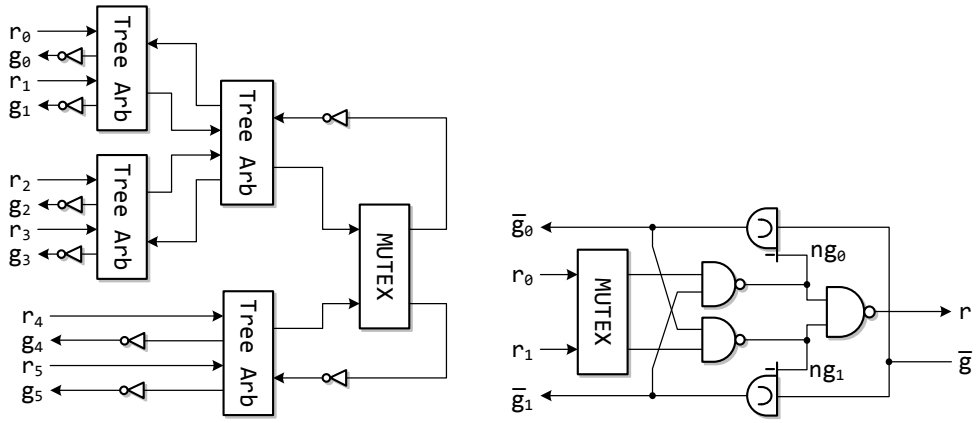
The 3-way MUTEX arbiter can be expanded into an arbitrary N-way MUTEX arbiter shown in Figure 2.9b [66]. The total number of MUTEX gates in an N-way MUTEX arbiter is $\binom{N}{2}$ where N is the number of requests.

Multi-way MUTEX arbiters have several advantages over other arbiter structures. Since the number of MUTEXes on every request is equal, requests have the same minimum arbitration delay. Multi-way MUTEX arbiters are strictly fair. They are fast and area efficient when the number of requests is small. However, multi-way MUTEX arbiters are no longer area efficient when the number of requests increases to a certain amount, and other arbitration structures should be selected.

2.6.2 Tree arbiter

The tree arbiter is an expandable arbiter structure and it is area efficient, able to handle a large number of requests. Figure 2.10a shows the diagram of a tree arbiter with six requests. It is expanded from the basic MUTEX gate. The six requests are first divided into two groups and their combined requests are guarded by the MUTEX. Then the tree arbiter grows into a binary tree. Each arbiter block, as shown in Figure 2.10b [61], divides its requests into two sub-groups until there is only one request in each sub-group.

In Figure 2.10b, the negative grant signals $\bar{g}_0$ and $\bar{g}_1$ are driven by two asymmetric C-elements (Appendix A). These two asymmetric C-elements guarantee that the output grant signal $\bar{g}_0$ or $\bar{g}_1$ is low (active) only when this arbiter block is granted (low on $\bar{g}$)



(a) Diagram (b) Arbiter block

Figure 2.10: Tree arbiter

and the internal MUTEX has chosen the corresponding request through signals ng_0 or ng_1. When the granted request is released, the combined request r drops and later the positive voltage on $\bar{g}$ (inactive) drives both $\bar{g}_0$ and $\bar{g}_1$ to high.[1]

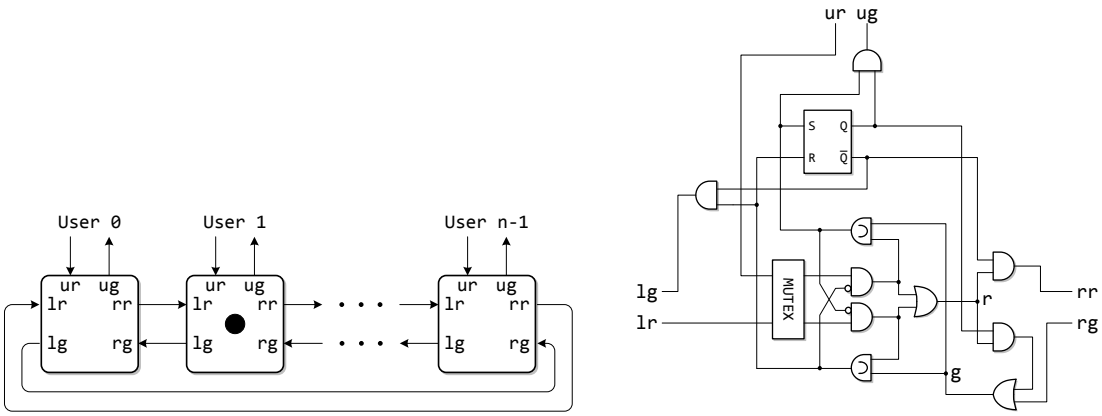
Tree arbiters are strictly fair arbiters when the binary tree is balanced; otherwise, the leaf requests near to the root MUTEX gate are more likely to be granted than others. The area of a tree arbiter is linear with the number of requests as every new request leads to a new arbiter block. A request is granted when all the arbiter blocks on its route to the root MUTEX gate select it. Thus the minimum arbitration delay is proportional to the depth of each request. If the tree arbiter is balanced, the depth of the tree is $\lceil log_2 N \rceil$ where N is the number of requests. Therefore, the delay of tree arbiters increases logarithmically.

2.6.3 Ring arbiter

First introduced in [75], the ring arbiter is the asynchronous "round-robin" arbiter. Figure 2.11 [105] shows an implementation of ring arbiter developed from [75].

In the ring arbiter, each request from a user is connected to an individual arbiter block shown in Figure 2.11b. The whole ring arbiter has only one token. A request must secure the token in order to be granted. If the token is not stored inside the arbiter

[1] In the original design [61], symmetric C-elements are used instead of asymmetric C-elements. Because ng_0 and ng_1 are directly connected to C-elements and they are released before $\bar{g}$ returns to high, it is not necessary to check the values on ng_0 and ng_1 in practical circuits. Using asymmetric C-element reduces delay and area.



(a) Diagram (b) Arbiter block

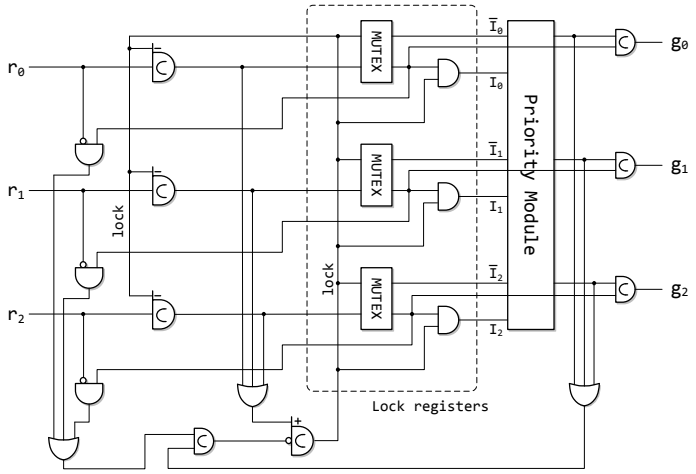
Figure 2.11: Ring arbiter

block connected to the active request, the arbiter block forwards this request to its neighbour. In this implementation, the forward process goes clockwise.

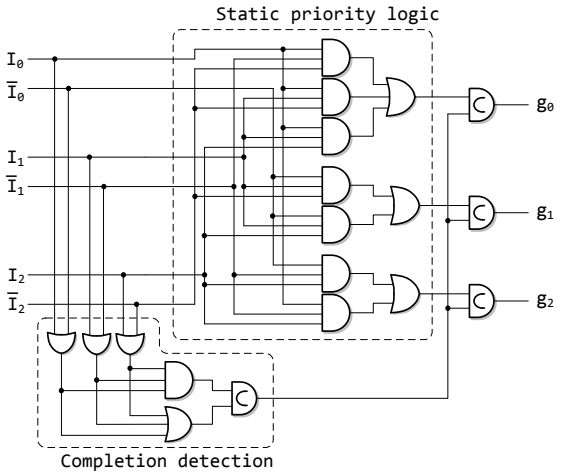
The arbiter block has a similar internal structure to the tree arbiter block. Both of them have two request inputs and one combined request output, which is the forward request to its neighbour in a ring arbiter. These two input requests, one from the local user and the other one from its anti-clockwise neighbour, are selected by a MUTEX gate. The winner first checks the local token storage, which is the set-reset (RS) latch in Figure 2.11b. A grant is immediately returned if there is a token stored locally, otherwise the combined request is forwarded to the neighbour.

Ring arbiters have a special advantage over all other asynchronous arbiters: it remembers the last granted request and select users in a round-robin fashion. The minimum arbitration delay for a request depends on the location of the token. On average, the distance between the active request and the token is $N/2$ where N is the number of requests. Thus delay is proportional to the number of requests. Obviously, the area of a ring arbiter is also proportional to the number of requests.

Another way to build a ring arbiter is to let the token circle the ring. A request is granted when the token arrives at its arbiter block [73]. It is shown that using the circling token scheme reduces arbitration latency by half but increases the power consumption significantly when requests are infrequent.



(a) Diagram



(b) Priority module

Figure 2.12: Static priority arbiter

2.6.4 Static priority arbiter

Figure 2.12 shows a 3-way static priority arbiter (SPA) [17]. It has two parts: request capture logic and a priority module.

The capture of requests is controlled by *lock* which is generated by the asymmetric C-element at the bottom. Initially *lock* is low and the column of asymmetric C-elements output low. Incoming requests go through the column of C-elements and MUTEXes. At the same time, *lock* is driven high, blocking other MUTEXes without an incoming request. Since multiple requests can arrive simultaneously and be captured by multiple MUTEXes, the priority module grants the request with the highest priority. After the request is served, the withdrawal of the corresponding *r* triggers the

release of *lock*, which consequently resets the corresponding asymmetric C-elements in the left column. Later the corresponding g is released. If new requests have been locked by those MUTEXes, *lock* returns to high immediately.

The priority module shown in Figure 2.12b is configured to grant all requests with approximately the same probability. Other priority configurations can be easily achieved by modifying the AND and OR matrix. It is also possible to reconfigure the priority dynamically [17].

Static priority arbiters are fast and area efficient. The arbitration delay depends on the request capture delay and the delay of the priority module. As the positive transition on *lock* and the locking requests in MUTEXes occur concurrently, the request capture time is short and independent on the number of requests. The delay of the priority module can be long if the number of requests is large.

2.7 Allocator

Although arbiters are capable of allocating one resource to multiple requests, they cannot allocate multiple resources concurrently, which is commonly required in asynchronous on-chip networks. In this case, asynchronous allocators are utilized instead of arbiters. In this section, several state-of-the-art asynchronous allocators are introduced. In the description, a resource is named a "client" to avoid the same initial letter with requests.

2.7.1 Virtual channel admission control

The virtual channel admission control (VCAC) module used in the QNoC project [38] allocates multiple virtual channels (clients) to several requests; it is an allocator design. Figure 2.13 shows the schematic of a VCAC allocator that can match m clients to n concurrent requests in a sequential way. It contains an n-way MUTEX arbiter and an m-way SPA configured with linear priority.

The allocation runs in three steps: Firstly, one incoming request is selected by the n-way MUTEX arbiter. Secondly, if there are available clients, the m-way SPA is enabled and an available client is chosen for the selected request. Finally, the request and the client are matched inside the match matrix which then generates a configuration (or grant).

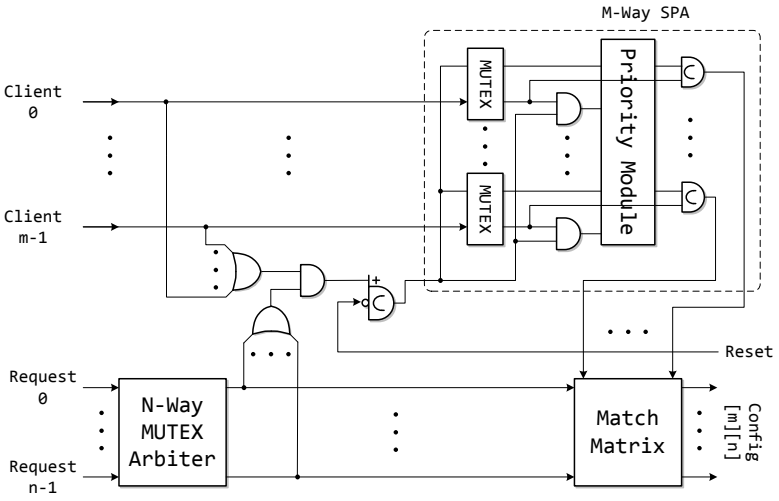


Figure 2.13: Virtual channel admission control

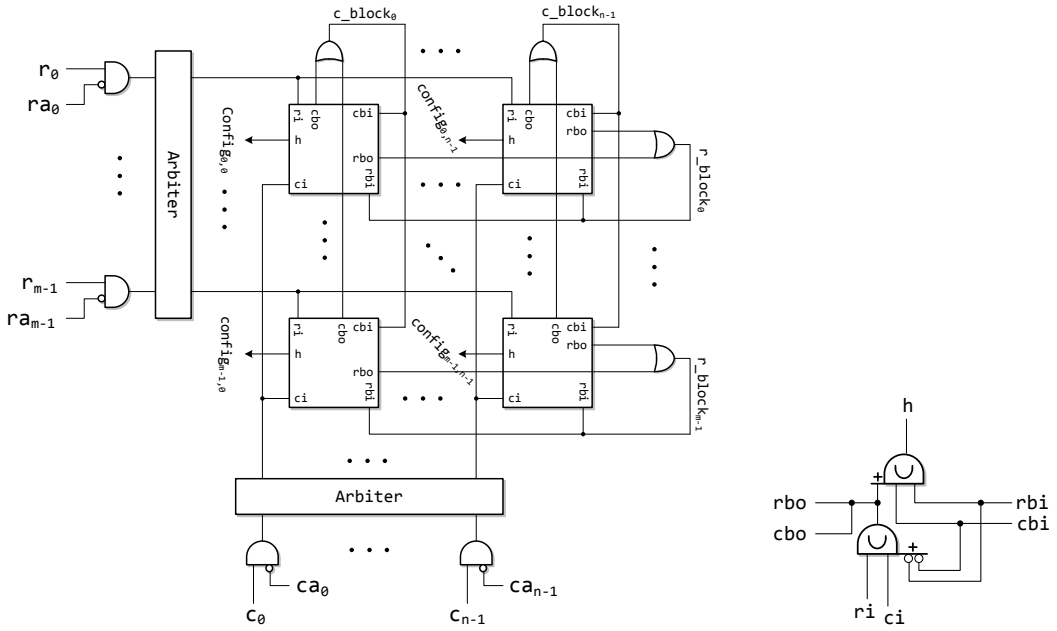
Because the VCAC allocator is capable of matching only one pair of request and client at one time, the generated configuration must be latched outside the allocator. The corresponding request and client inputs are also withdrawn once the configuration is captured. A pulse on *reset* is also generated to allow the SPA to choose another client.

The VCAC allocator is area efficient and fast but has some timing issues. The release of requests and clients is not guarded; therefore, VCAC is not strictly QDI. Only one pair of request and client can be matched at one time, which can be a bottleneck in a busy system. The VCAC allocator is not prioritized. It is believed that SPA is used in VCAC due to its reset capability which makes its peripheral circuit easy to design. However, the area overhead of a SPA is larger than a fair (non-prioritized) arbiter such as a multi-way MUTEX arbiter or a tree arbiter.

2.7.2 Multi-resource arbiter

A multi-resource arbiter is QDI [52, 53, 104]. As shown in Figure 2.14, multi-resource arbiters and VCAC have similar structures. Both have two arbiters and a match matrix, but the match matrix of a multi-resource arbiter is more complicated than that of VCAC. Every tile in the match matrix of a multi-resource arbiter is a speed independent circuit which ensures the corresponding request and client can rise and drop independently.

Like VCAC, a multi-resource arbiter matches clients to requests in a sequential way. At first, the arbiters on requests and clients select a winner pair of request (*ri*) and



(a) Diagram (b) Tile

Figure 2.14: Multi-resource arbiter

client (*ci*) independently. Then the asymmetric C-element in the corresponding tile (Figure 2.14b) drives *rbo* and *cbo* to high. These two signals trigger the corresponding blockage signals *r_block* and *c_block* in Figure 2.14a, which block all the tiles in the same row and column of the chosen request and client. Once the tile receives the blockage signals meaning that other tiles are safely blocked, the configuration *h* is generated.

The allocated pair of request and client must be withdrawn as soon as possible to allow other requests to be served. The configuration is stored outside the multi-resource arbiter. The peripheral circuit has to raise the corresponding acknowledge signals *ra* and *ca* to withdraw the allocated request and client. In the match matrix, the blockage signals *rbi* and *cbi* guarantees that the lower asymmetric C-element in Figure 2.14b is disabled until the previous allocation is safely withdrawn.

The major advantage of the multi-resource arbiter over VCAC is its robustness. Multi-resource arbiters are QDI and the timing relation between requests and clients are fully decoupled. As a result, the arbiters on requests and clients run independently. However, the tile based match matrix is area consuming. When the number of requests or clients is large, the blockage signals in the match matrix may fail to release all tiles soon enough when false configurations are produced. The reason is related to the QDI delay assumption. The blockage signals *r_block* and *c_block* no longer fit the

isochronic fork assumption in a large design. A blockage ring can resolve this problem but it introduces extra area overhead [104].

2.8 Summary

This chapter has introduced some fundamental concepts of asynchronous circuits. Asynchronous circuits are clockless circuits driven by handshake protocols. Data in asynchronous circuits can be delivered by bundled-data pipelines using the self-timed delay assumption or by multi-rail pipelines using the QDI delay assumption. The arbitration circuits, including arbiters and allocators, have also been described. Unless stated otherwise, the circuit designs in this thesis are relaxed-QDI circuits using the 4-phase multi-rail pipelines.

Chapter 3

Network-on-Chip

The number of transistors on a single die has doubled periodically as predicted by Moore's law for several decades but the design complexity coupled with abundant resources is becoming unmanageable. Traditional SoC systems were hierarchically built from processors, functional units, intellectual property (IP) blocks and bus-based communication infrastructures. When tens to hundreds of SoC systems are packed into one die, the hierarchical bus is not scalable and becomes the major throughput bottleneck. To overcome this problem, researchers have proposed the idea of network-on-a-chip (NoC) over the last decade [34, 9, 57, 89]. In such a NoC system, SoC sub-systems use buses as their local communication method but the data travelling across the boundary of a sub-system are encapsulated into packages and delivered to the target sub-system by a scalable network-based communication infrastructure [34]. In this chapter, the background of NoCs and the recent developments of asynchronous on-chip networks are introduced.

3.1 Concepts of on-chip networks

Network-on-a-chip or on-chip network is the chip level communication infrastructure that connects tens to hundreds of sub-systems. As shown in Figure 3.1, a sub-system has its own bus, processor, memory/cache and other function blocks. Such a sub-system, namely a processing element (PE), communicates with other PEs through the on-chip network consisting of routers. The network interface in each PE, acting as a bridge, converts and delivers data between the on-chip network and the local sub-system.

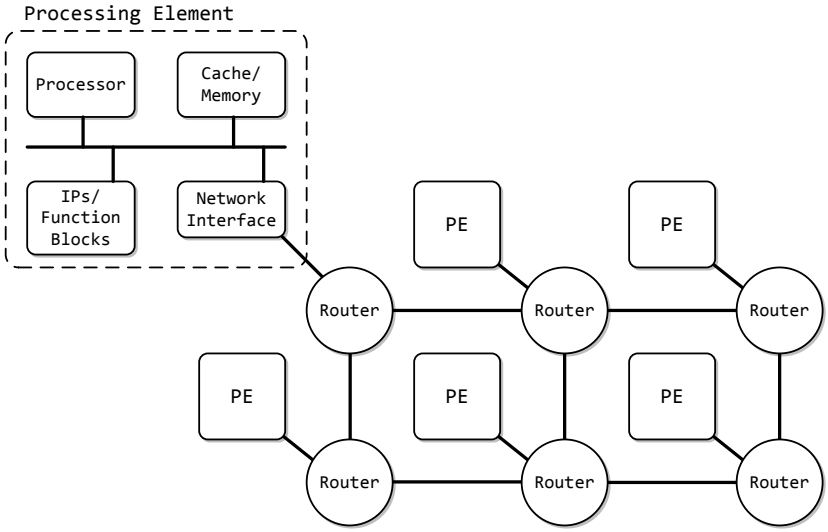
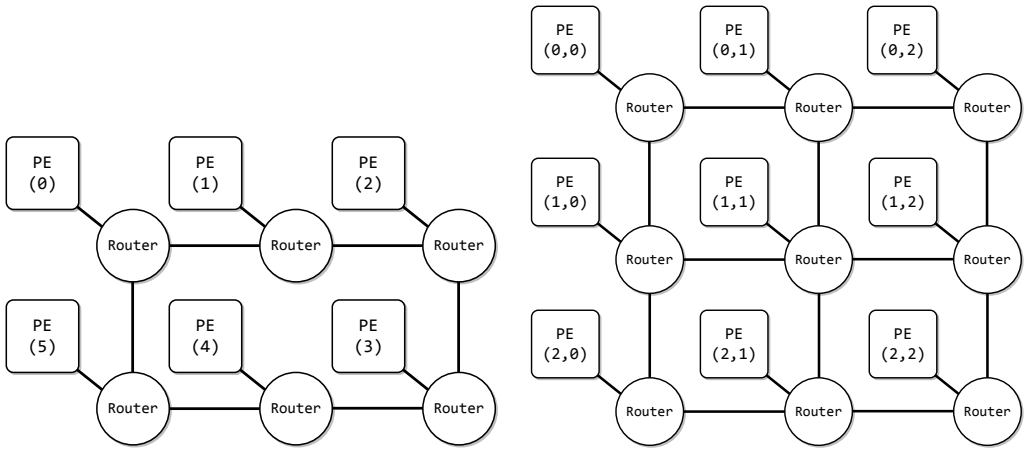


Figure 3.1: Architecture of NoC

Depending on the structure of PEs, a NoC can be classified into two categories: homogeneous NoCs and heterogeneous NoCs. All PEs in a homogeneous NoC use the same internal structure; therefore, they have the same size and the chip is symmetric, whereas the PEs in a heterogeneous NoC may use different structures. The size of a PE is therefore variable and the chip floorplan is hardly symmetric. Chip multi-processors are normally homogeneous NoCs as every PE is a processor [127]. On the other hand, MPSoCs are generally heterogeneous NoCs because PEs have different functions [26].

It is possible to describe the network using layer models such as the open system interconnection model used in macro networks. However, there is currently no consensus on the division of layers in NoCs. According to one well-accepted layer division [41], the lower network structure can be divided as *physical layer*, *switching layer* and *routing layer*. The physical layer determines the way that PEs are connected to each other. The switching layer describes how data packages are delivered and how resources, such as buffers and links, are allocated to data packages. Finally the routing layer controls the path over which a data package traverses.

The following sections will describe the major design issues in different layers: network topologies (physical layer), flow control methods (switching layer) and routing algorithms (routing layer).



(a) Ring (b) Mesh

Figure 3.2: Examples of direct networks

3.2 Topology

"Network topology refers to the static arrangement of channels and nodes in an interconnection network — the roads over which packets travel."[35] The topology of a NoC determines the global layout and fundamentally affects the strategies and performance of higher layers. Selecting a topology is the first step of building a NoC.

A node in a NoC is either a switch node which is basically a router delivering packets from input ports to output ports, or a terminal node which contains a PE generating and receiving packets. A network is a direct network when every router is connected to one or more PEs, otherwise, it is an indirect network where some routers are connected only with other routers.

The ring network shown in Figure 3.2a is a direct network. PEs in this network can be addressed by a vector. Routing algorithms are simple as every pair of PEs has only two possible paths: clockwise and anti-clockwise. The maximum distance between two PEs is $\lfloor N/2 \rfloor$ where N is the number of PEs. It is beneficial to use the ring topology with a small N thanks to its simple structure and low hardware overhead but the distance between two PEs can be intolerably long when N is large.

Mesh is one of the most utilized topologies in direct on-chip networks. Figure 3.2b shows a 3×3 mesh network where every PE is addressed by a two-dimensional index (x, y). Compared with ring networks, mesh networks have two major advantages: One is the reduced communication distance. For two PEs located at (x_1, y_1) and (x_2, y_2),

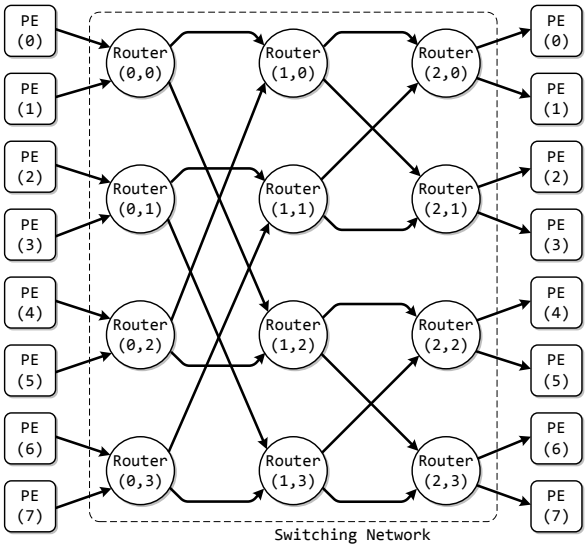


Figure 3.3: A butterfly network

the minimal distance is the taxicab distance ($|x_1 - x_2| + |y_1 - y_2|$). Thus the maximum distance is $2(\sqrt{N} - 1)$ where N is the number of PEs. When N is larger than 12 (a 4×3 mesh), mesh networks have shorter distances than ring networks. The other is the scalable throughput performance. The paths between a pair of PEs are not deterministic in mesh networks. The number of channels per PE in a mesh network is $\lim_{N \to +\infty} \frac{2\sqrt{N}(\sqrt{N}-1)}{N} = 2$ while it is only one in a ring network. Mesh networks doubles the amount of communication resource. The network throughput should increase consequently if data packets are directed to all channels in a balanced way.

Most MPSoC designs use direct networks [26]. The design of a direct network is modularized. Every router is connected to a PE. As long as the interface between PEs and routers is unified, all routers are functionally equivalent and PEs can be designed independently. This is essential to meet the short time-to-market requirement for current MPSoC designs.

Unlike MPSoC systems, many CMP systems prefer indirect networks. Figure 3.3 depicts a butterfly network [35]. In the figure, the PEs in the left column are in fact the same PEs in the right column. Duplicating them in the figure is a simplified way of showing the butterfly network between PEs. In a practical implementation, the butterfly network is folded [102].

Butterfly networks are one class of indirect networks. All pairs of PEs are connected by deterministic paths with equal communication delays. The butterfly network between PEs can be replaced with other switching networks, such as crossbars [94],

Clos networks [102], Beneš networks, etc. (Chapter 6). All of these switching networks provide fixed communication delay between all pairs of PEs. In some switching networks, such as the butterfly networks and crossbars, only one path exists between any pair of PEs. In other switching networks, such as Clos networks and Beneš networks, multiple paths are available. Compared with direct networks, these indirect networks provide equal minimal communication delay between all PEs but the network implementation is not modularized. The routers and channels in an indirect network must be carefully calibrated to provide equal delays and fair resource allocation.

As most MPSoC systems use direct networks, all router designs in this thesis assume that the mesh topology is selected.

3.3 Flow control

Network topology provides communications with different paths. Communications are launched concurrently and paths are selected individually. Contention occurs when more than one communication chooses the same resource at the same time. To resolve such contention, the resource is allocated to one communication forcing the others to wait until the allocated one is served. The algorithms utilized in this process are called flow control methods. They dynamically allocate resources inside routers and extensively influence the router structure.

Before describing the numerous flow control methods utilized in on-chip networks, the data packets delivered in on-chip networks need to be clarified. Normally a self-explainable data packet generated by a PE is a *frame*. Each frame has a header denoting the target PE and a payload containing the data to be delivered. A frame is usually large and cannot be delivered in a single piece. It is divided into *flits*, each of which is able to be transmitted through a channel or a switch at one time.

3.3.1 Circuit switched and packet switched

Flow control methods can be classified into two categories: circuit switched and packet switched. The flow control methods used in circuit switched networks allocate a path between the transmitting PE and the receiving PE for each frame before delivering the frame payload. On the contrary, the flow control methods used in packet switched networks deliver frames immediately after they are generated. The path for each frame

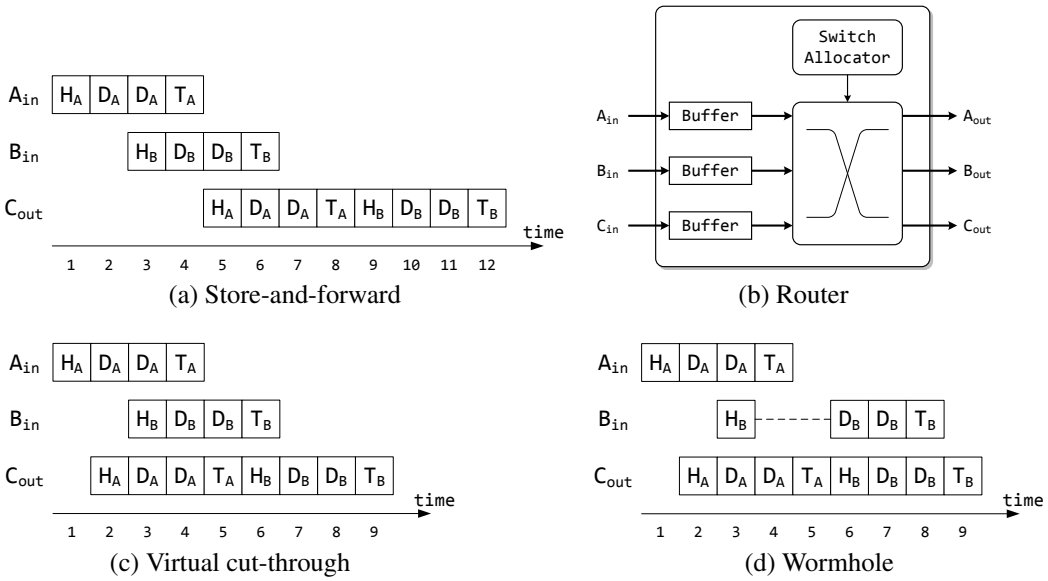
is dynamically allocated on the fly. The flow control methods in packet switched networks do not guarantee that enough resources are available before delivering payload.

The path allocation in a circuit switched network can be static: the path configuration is prefabricated or statically configured during the setup stage [129, 71, 119]. It is also possible to allocate path dynamically. One way is using handshake protocols [128, 12, 88]. The initiating PE sends a request to the target PE. This request frame reserves all channels which it has traversed. When it reaches the target PE, an acknowledge returns through the same path denoting that a path is reserved. The other way is by reconfiguring the switches by instructions from the connected PEs [125].

Circuit switched networks are area and energy efficient compared with packet switched networks. Paths are statically configured for a long time or even the whole life time. As a result, routers do not analyse data most of the time and switch configuration is simple. Little or no energy is consumed by reconfiguring switches. No buffer is needed because two communications cannot share any resource at the same time. However, a significant portion of the available throughput is wasted when a communication pauses temporarily or some resources are reserved by an ongoing path request. When the network is heavily loaded, circuit switched networks suffer from long communication latency and significant power overhead in the path reservation process [128].

Rather than allocating a path to a communication before any data transmission, packet switched networks dynamically allocate resources to frames. Since resources are not pre-allocated, it is unavoidable that multiple frames may compete for the same resource, particularly when a network is heavily loaded. The flow control method temporarily allocates the resources to one frame and reserves some buffers for the blocked frame. Depending on the buffer space reserved in each allocation procedure, a packet switched network can be classified into three types: a store-and-forward network where buffers are reserved in units of frames, a wormhole network where buffers are reserved in units of flits, and a virtual cut-through [63] network where buffers are reserved in units of flits but enough buffer space is ensured to store a whole frame.

Figure 3.4 illustrates an example of different packet switched networks. The router in Figure 3.4b has only three ports: A, B and C. Input ports and output ports are dynamically connected through the crossbar, which is reconfigured by the switch allocator. A buffer is inserted between each input port and the crossbar. If the router is used in store-and-forward or virtual cut-through networks, the buffer is enough for a whole frame, otherwise it is only able to store one flit (in wormhole networks). The



(a) Store-and-forward (b) Router

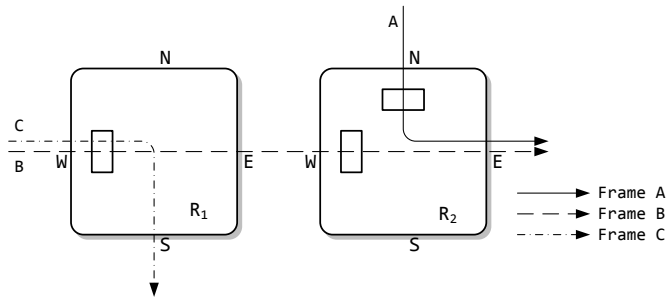
(c) Virtual cut-through (d) Wormhole

Figure 3.4: Packet switched flow control

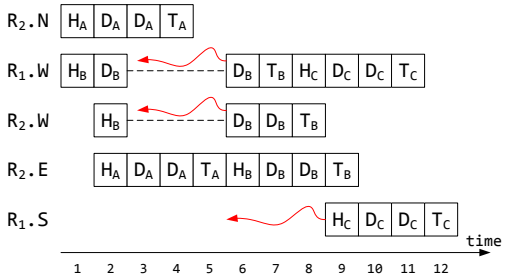
frames in this example have fixed length of four flits: one header flit denoting the target PE, two data flits carrying the data payload and a tail flit indicating the end of a frame. Assuming that networks are implemented in synchronous circuits, each channel can transmit one flit in every cycle. Two frames *A* and *B* arrive at the router from input ports A_{in} and B_{in} at the first and the third clock cycle respectively. Both of them ask for the output port C_{out}. Contention is thus generated.

As shown in Figure 3.4a, the router in a store-and-forward network starts to transmit a frame only when all its flits are received. Since frame *A* arrives before frame *B*, the output port C_{out} is first allocated to *A* but it is only utilized until the fifth clock cycle when all parts of frame *A* are received. Consequently, frame *B* is temporarily stored in buffers and is transmitted when frame *A* is served. The idea of store-and-forward is straightforward but using it leads to long frame transmission latency and waste of resources. In this example, the output port C_{out} is allocated but not utilized for three clock cycles.

Virtual cut-through reduces the long latency by allowing routers to transmit a frame whenever a flit is received. As shown in Figure 3.4c, the output port C_{out} starts to transmit frame *A* immediately after the header flit is received. Although virtual cut-through reduces frame transmission delay, the buffer size is not reduced. Virtual cut-through requires enough buffer space for every frame. Thus a frame is stored in only one router when it is blocked. In some networks, a frame can be extremely long and



(a) Router connection



(b) Data flow

Figure 3.5: Head-of-line

the area overhead of this requirement will be intolerable.

The wormhole flow control method relaxes the size requirement to one flit in the minimal case. Figure 3.4d demonstrates the outcome of limiting the size of buffers to one flit. Frame *A* is served before frame *B* but now only the header flit of frame *B* is stored in the router. Other parts of frame *B* are blocked in previous routers. In this example, both virtual cut-through and wormhole can deliver two frames in nine clock cycles.

3.3.2 Virtual channel

The wormhole flow control flow control method reduces buffers significantly. However, it has head-of-line (HOL) issues introducing long frame transmission delay when networks are heavily loaded.

Figure 3.5a shows an example of how HOL prolongs the frames transmission latency. Three communications (*A*, *B* and *C*) are initiated concurrently. Frames *A* and B compete for the east output port of router R_2. The other frame, frame *C*, has contention with frame *B* and will be transmitted through the west input port of router R_1 immediately after frame *B*. All frames have the same length of four flits. Initially both frames *A* and *B* arrives at ports $R_2.N$ and $R_1.W$ respectively at the first clock cycle.

The result data flow is depicted in Figure 3.5b. In router R_2, frame A is one clock cycle earlier than frame B and is thus served first. As a result, frame B is blocked waiting for the output port $R_2.E$ to be available again. In the meanwhile, the first two flits of frame B, H_B and D_B, are stored in the buffers connected to ports $R_2.W$ and $R_1.W$ respectively. As described, frame C is waiting for frame B. As frame B is blocked, frame C is also blocked by the contention in router $R2$, although frame C does not go through R_2 at all. This is the HOL problem introduced by the wormhole flow control method.

If the buffer space is large enough for a whole frame, the virtual cut-through flow control method can be used and frame C would be delivered three clock cycles ahead (indicated by the red arrows in Figure 3.5b). In this case, the whole frame B is temporarily stored in the buffers in router R_2 while frame C is in transmission.

The nature of HOL is: When a frame is blocked in one router, the router does not have enough buffer space to store the blocked frame. As a result, parts of the blocked frame are stored in other routers and corresponding switches and channels are occupied. These occupied resources then block other frames who are active and asking for these resources at the same time

Virtual channel (VC) is an extensively utilized flow control method which significantly alleviates the HOL problem [30]. The basic idea of VC is to share channels among multiple frames. When one frame is blocked, the frame must not occupy the channels which are otherwise reserved in wormhole networks; therefore, other frames can utilize these channels at the same time. However, the flits belonging to the blocked frame cannot be thrown away. Extra buffers are inserted and the buffers connecting to each input port are divided into groups, each of which is called a virtual channel and stores flits of an individual frame.

A VC router is shown in Figure 3.6. It has four bidirectional ports: south, west, north and east. Two VCs are inserted between each input port and the crossbar. Channels and the crossbar is shared by both VCs. The allocation of VCs is controlled by a VC allocator.

When two frames compete for the same channel, they are allocated different VCs. Both frames are transmitted through the same channel using different VCs. Obviously only one flit is delivered in every cycle. Thus the shared channel is occupied by both VCs in a time divided manner. When one frame is blocked wherever in the network, the other frame can fully utilize the channel.

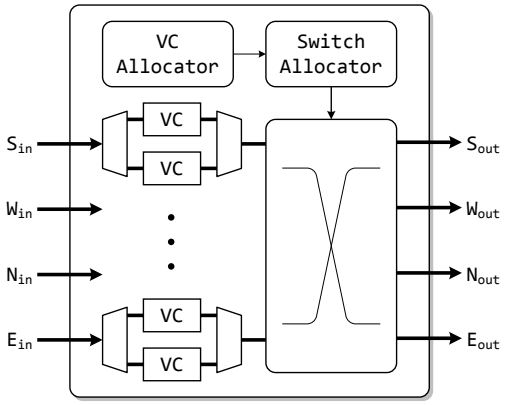
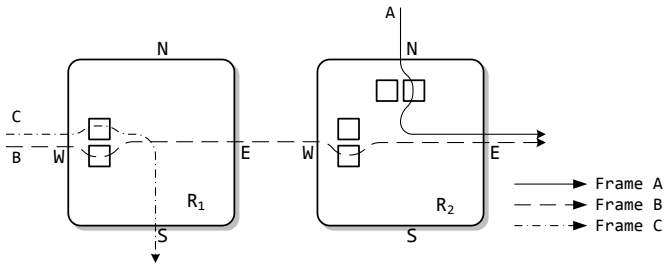
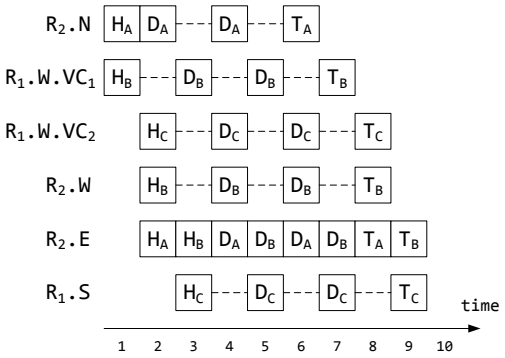


Figure 3.6: VC router



(a) Router connection



(b) Data flow

Figure 3.7: Resolving head-of-line with VCs

To further illustrate the impact of VCs, the same example in Figure 3.5 is implemented again using VC routers as demonstrated in Figure 3.7. All VCs and crossbars are allocated by fair round-robin arbiters. Frames *A* and *B* share the channel connected to the east output port of router R_2, which is therefore alternately used by both frames. The same sharing procedure occurs on the channel connected to $R_1.W$. Frames *B* and *C* occupy different VCs and are delivered alternately. Finally, all frames are transmitted in nine cycles, which is three cycles shorter than using wormhole routers.

3.3.3 Other flow control methods

VC is the most popular flow control method in NoCs [45, 12, 84, 69, 38, 26] but it is not the only flow control method.

Time division multiplexing (TDM) is another extensively utilized flow control method[1] [56]. It divides time into time slots and multiple frames share the same channel by occupying some slots. The number of slots allocated to a frame defines the maximum throughput available to the frame. The transmission delay of every frame is predictable. However, the network throughput may be wasted when some slots are not allocated or the allocated slots are not used.

Rather than sharing a channel among multiple frames in a time divided manner, spatial division multiplexing (SDM) divides every channel into several virtual circuits and allocates virtual circuits to frames [71, 54] (Chapter 5). In this way, multiple frames traverse the same channel concurrently using different virtual circuits. However, frames are effectively serialized as every virtual circuit provides only a portion of the total bandwidth of a channel. Although using SDM prolongs the transmission delay of a single frame, the overall throughput increases because no contention occurs until all virtual circuits in a channel are allocated.

3.3.4 Quality of service

Quality of service (QoS) is an important concept in NoCs. The communications in a network have different requirements, such as delay or throughput. For the whole system to work properly, a network needs to provide certain guaranteed services (GS) to meet the requirements of these communications.

Some communications require reaching their targets before certain deadlines, such as the interrupt signals in some real-time systems. The networks supporting this service are called delay-guaranteed networks. Circuit switched or TDM flow control methods can be used to build hard delay-guaranteed networks where communications always reach their target before deadlines. Other flow control methods, such as VC and SDM, provide only soft delay-guaranteed services where the probability that communications reach their target before deadlines is guaranteed.

Many communications, such as multimedia data streams, can tolerate some delay variance but are sensitive to throughput. The networks that satisfy this requirement are

[1] Strictly speaking, VC is also a TDM method because channels are shared by multiple VCs in a time divided manner. In literatures, TDM usually refers to its narrow meaning of sharing channels according to explicitly defined time slots.

called throughput-guaranteed networks. Nearly all flow control methods can provide through-guaranteed services by assigning high priorities to the data with the throughput requirement over others.

Some communications, or even all communications when a network is fast enough, do not have requirements. It is beneficial to deliver them as soon as possible but failing to do so will not break down the whole system. These communications are called best-effort (BE) traffic. Wormhole, VC and SDM flow control methods are good at providing such service.

3.4 Routing algorithm

The routing algorithm determines the path via which a frame traverses a network using certain topology. It is an important research topic in NoCs as a poor routing algorithm can congest the network by overloading some communication resources while leaving others unused. However, the problem is so complicated that no routing algorithm can perfectly balance all communications in all circumstances. A routing algorithm that fits one traffic pattern may congest the same network if the pattern changes. However, the traffic pattern is not always pre-known at design time and can vary dramatically at run-time. Furthermore, if on-chip faults are considered, the network topology is altered temporarily or permanently when a fault occurs. This change of topology can convert a deadlock-free routing algorithm into a deadlocking one. Obviously the network throughput is significantly affected.

The major research goal of this thesis is to explore better trade-offs among speed, area and power in asynchronous on-chip networks. The research concentrates on the hardware structure of channels and flow control methods. As a result, this thesis will not discuss the impact of using different routing algorithms on certain router implementation in detail. This section will go over the classification of routing algorithms, introduce those routing algorithms extensively utilized in NoCs and analyse their performance.

3.4.1 Deterministic and non-deterministic

Routing algorithms can be classified into two categories: deterministic routing algorithms and non-deterministic routing algorithms. The frames of any pair of PEs in a network using a deterministic routing algorithm always go through the same path

while they may go through different paths if a non-deterministic routing algorithm is used. The advantage of deterministic routing algorithms is their simplicity. They do not require complicated routing calculation circuits in routers. They are also capable of routing communications in non-regular networks where some non-deterministic routing algorithms may not work. However, they lack flexibility in their paths. A network can easily be congested when multiple communications simultaneously request the same resource. They can also generate deadlocks when the network topology is altered by faults [31]. In these situations, using non-deterministic routing algorithm is a better choice.

Most deterministic routing algorithms use source routing. The frame sender knows where the target is and specifies a route there. The usual way to implement source routing is to record the chosen path in the header flit. Every router on the path analyses the header flit, finds out the recorded output port, removes the turn in the record, and forwards the modified header flit to the next router. Source routing has been utilized in many NoC designs [10, 12, 7, 127] due to its advantages: It can be used in any topology and it introduces extremely low hardware overhead. However, the path recorded in the header flit increases frame size. If the path is extremely long, multiple header flits are required to record the path introducing extra delay and energy.

If the topology is regular (ring, mesh or torus [32]), a class of source routing algorithms, namely dimensional order routing (DOR) [33], can be used without recording the whole path within the header flit. Instead, only the address of the target is recorded. Frames are delivered to targets through the edge of the minimal line (ring) or rectangle (mesh or torus) that can be drawn between the source and the target. As an example, if a frame is sent from (x_1,y_1) to (x_2,y_2) in a mesh, the selected path will be (x_1,y_1) $\rightarrow$ (x_2,y_1) $\rightarrow$ (x_2,y_2). To obtain the correct output port, routers simply compare the target address in the header flit with their own addresses. When DOR algorithms are used in a mesh or torus network, they are also called XY routing algorithms.

DOR algorithms have many advantages over other routing algorithms: Since DOR algorithms are source routing algorithms, they introduce extremely low area overhead. Meanwhile, DOR algorithms are also distributed routing algorithms as the routing decisions are made by routers. It is not necessary to pre-define the whole path and store it within the header flit as other source routing algorithms. The selected paths by DOR algorithms are minimal paths leading to the shortest frame delay when networks are not busy. Due to these advantages, DOR algorithms have been extensively utilized in many NoC designs [80, 47, 83, 79, 105, 38].

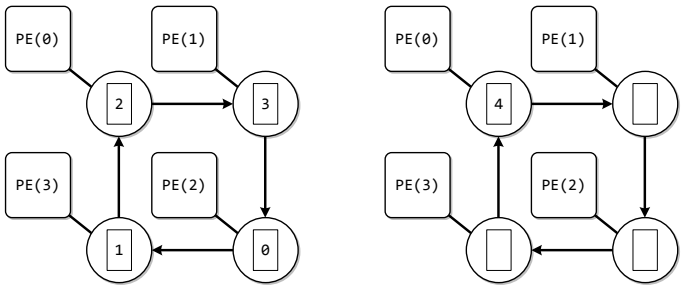
A major problem of deterministic routing algorithms is their lack of path flexibility, which leads to congestion and prolonged frame delay when network traffic is not uniform. Non-deterministic routing algorithms, on the other hand, can divert frames away from their original paths and thus distribute traffic to all resources. Depending on the methods of generating diversions, non-deterministic routing algorithms can be classified into two categories: oblivious routing algorithms and adaptive routing algorithms.

Oblivious routing algorithms route frames without regarding the states of the network [34].[2] Valiant's randomized routing algorithm [126] can balance the load of any traffic pattern in nearly all topologies. For each frame, an intermediate node is randomly selected, and the frame is first delivered to the intermediate node and then to the target node. As the randomly selected node can be anywhere inside the network, the diversion is hardly a minimal path. ROMM algorithm [86] restricts the intermediate node to be one inside the minimal rectangle between the sender and the receiver, thus the diversion is also minimal. ROMM can outperform DOR algorithms in some unbalanced traffic patterns [109].

The stochastic routing algorithm is another well-known oblivious routing algorithm [42, 15]. A frame is duplicated and forwarded to multiple output ports in every router at which it has arrived. The result is that the frame is flooded from the sender to the whole network. This algorithm has strong tolerance to on-chip faults and it guarantees that a frame is delivered to the target as long as the target is still reachable. However, the flood wastes resources and throughput is extremely low. It is possible to constrain the flood to improve throughput while reserving the tolerance to faults [97, 88, 116].

Adaptive routing algorithms route frames according to the states of the network. Different algorithms divert frames for different reasons. Some algorithms try to balance the load and improve throughput [23, 58]. Some others provide tolerance to on-chip faults by diverting or retransmitting frames when a channel or a router is broken [55, 132, 100]. After all, the price of using them is normally expensive. The states of the network, such as congested buffers and channels, or broken channels and switches, need to be collected before they can be used to generate diversions. Extra area, energy and delay overhead is introduced consequently.

[2]Some literatures believe deterministic routing algorithms are also oblivious routing algorithms. Literally it is right but in this thesis oblivious routing algorithms refer to non-deterministic routing algorithms only.



(a) Deadlock (b) Livelock

Figure 3.8: Deadlock and livelock

3.4.2 Deadlock and livelock

Deadlock and livelock are harmful by-products of some routing algorithms. Their occurrences paralyse a part or even the whole network. A practical NoC design must either prove that its routing algorithm is safe from deadlock or livelock, or demonstrate that it can recover when deadlock or livelock occurs. Several issues must be considered: the routing algorithm should be deadlock-free. If it is not deadlock-free, the network must be able to recover from deadlock. The traffic pattern itself is assumed to be free of message-dependent deadlock[3] [28, 5]. If on-chip faults produce permanent damage, routing algorithms must be able to adapt to the altered topology otherwise the whole system must allow a temporary halt.

An example of deadlock and livelock is shown in Figure 3.8, which is a simple unidirectional ring network containing four PEs. Deadlock occurs when the channel dependency graph of a network is cyclic [33] — a group of concurrent frames are individually holding some resources which are mutually waited for by others in the same group. Since no frame releases its resources before obtaining the necessary resources occupied by others, all frames in the group stop. In Figure 3.8a, four frames targeting different PEs have consumed all buffers in all routers. Since every frame is waiting for the buffer in its clockwise neighbour to become free, no frame can be transmitted.

Livelock denotes the situation when one or more frames are continuously traversing the network without reaching its or their target(s). Figure 3.8b demonstrates an example of how livelock can occur in a ring network. A frame continuously circles the ring clockwise targeting the nonexistent PE(4). This may look ridiculous but it is

[3]Due to the limited buffer size in practical network interfaces, a deadlock occurs when a network interface fails to receive enough data in order to produce a response. This deadlock is introduced by the dependence between frames, which cannot be eliminated by deadlock-free routing algorithms alone.

possible when a transient fault attacks the header flit or a permanent fault disconnects PE(4) from the ring network.

Adaptive routing algorithms are prone to generate deadlock because they usually have cycles in the extended channel dependency graph [40]. These cycles can be removed by restricting the turn model [51, 23, 58] or by adding extra VCs [33]. However, a deadlock-free adaptive or deterministic routing algorithm may still produce deadlocks when permanent faults alter the network topology. Adaptive routing algorithms can cope with faults by changing their routing strategies [55, 16, 132] but expensive fault detection circuits are required. Oblivious routing algorithms are naturally tolerant to faults but they also need complicated strategies to avoid deadlocks [42, 97, 88, 116].

Instead of deadlock-avoidance, a network can recover from deadlock. However, deadlock-recovery mechanisms degrade throughput significantly [70].

3.5 Globally asynchronous and locally synchronous

An MPSoC system using an asynchronous on-chip network is a globally asynchronous and locally synchronous (GALS) network [68]. The structure of a GALS network is shown in Figure 3.9 where asynchronous circuits are depicted in grey. The on-chip network is built from asynchronous routers and asynchronous channels. A PE is a synchronous SoC sub-system connected to the asynchronous network by the network interface. Thus the network interface is the bridge between a local synchronous subsystem and the global asynchronous network. There are three different ways to implement an interface between synchronous and asynchronous circuits: pausible clocks, synchronizers and asynchronous first-in-first-out (FIFO) buffers [68].

The pausible clock scheme is simple to understand. The major issue of data transmission between synchronous and asynchronous circuits is the metastability problem [18]. Suppose a group of synchronous flip-flops (FFs) capture data from asynchronous circuits. The data may arrive at any time as asynchronous circuits are not synchronized. If the data arrive exactly during the positive transition of the global clock, FFs may go into metastability state during which the output values are not stable and wrong data may be generated. The pausible clock scheme resolves this problem by deferring the clock when metastability may occur. As shown in Figure 3.10a, the synchronous circuit is driven by a local clock generator. When asynchronous data are ready, they make a request to the local clock generator. If the data are going to clash with the

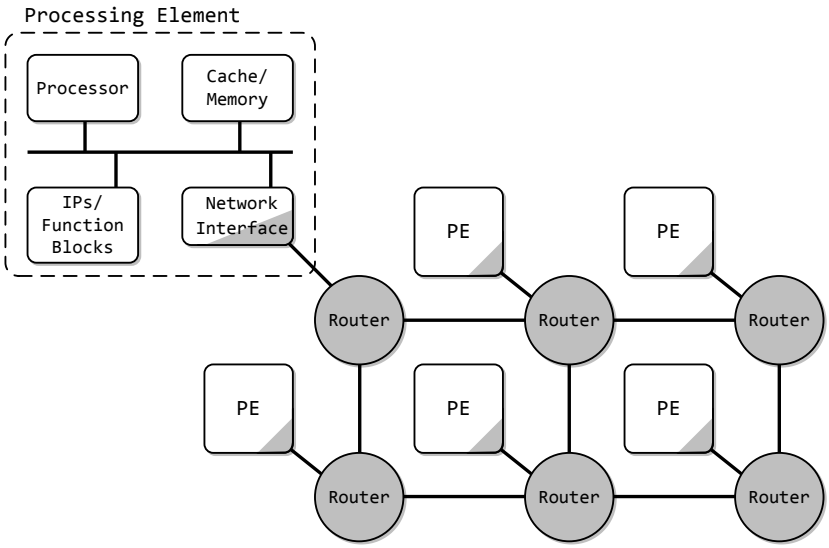


Figure 3.9: GALS network

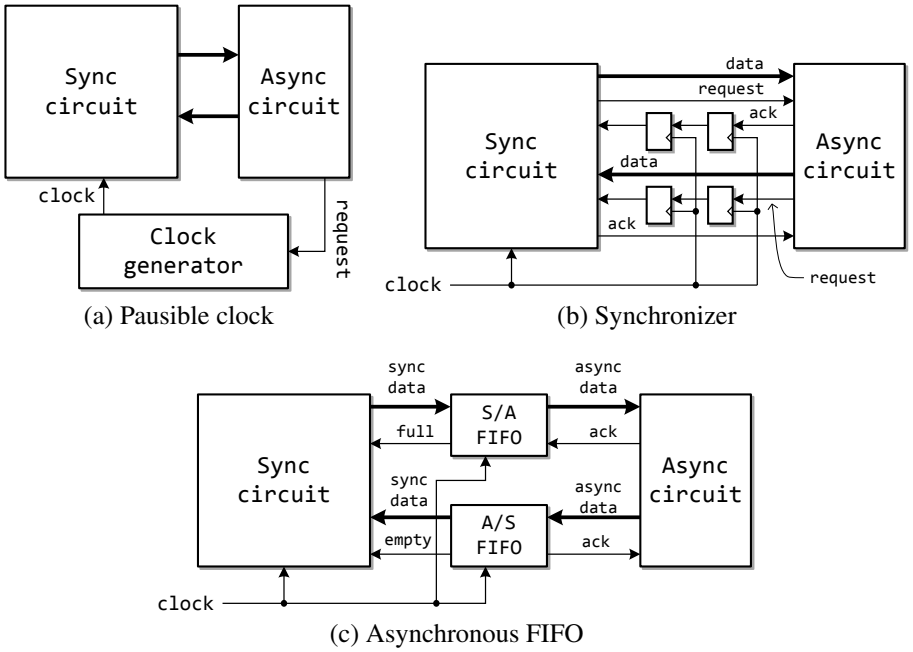


Figure 3.10: Synchronous and asynchronous interface

clock, the clock will be postponed until the asynchronous data can be safely captured. Hence the metastability problem is avoided [82, 124].

The pausible clock scheme has some advantages over other interfacing methods. It is the only scheme that can fully eliminate metastability. The data latency between asynchronous and synchronous circuits is extremely small as asynchronous data are

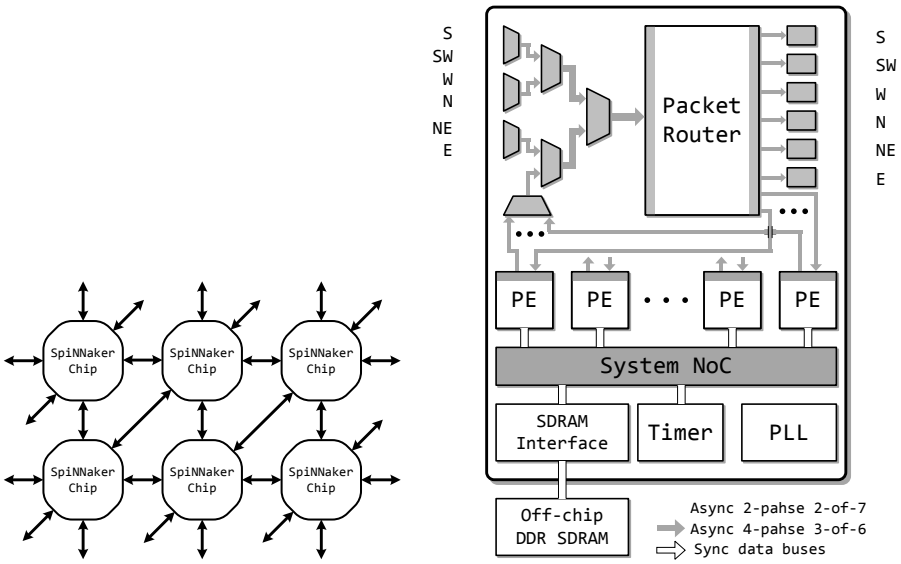
usually captured in the same cycle. The local clock generator also provides the possibility to implement dynamic voltage and frequency scaling (DVFS) [6], which can significantly reduce the power consumption of the synchronous sub-system. However, the pausible clock scheme restricts the highest clock frequency because the clock period is required to be longer than the clock tree delay [36]. It is also not practically viable for fabless designs to carefully calibrate the local clock generator [68].

Synchronizers are extensively utilized in synchronous circuits [50]. The basic two-flop synchronizers shown in Figure 3.10b are usually inserted on the control signals crossing clock boundaries to increase the mean time between failures (MTBF), although they cannot eliminate the metastability problem. The same method can be used on the asynchronous signals heading to synchronous circuits. For signals from synchronous circuits to asynchronous circuits, no synchronizers are needed as asynchronous circuits are event-triggered.

The sync/async interface built from synchronizers introduces the minimum area overhead of all interface schemes but the throughput is the lowest. Using the two flop synchronizer shown in Figure 3.10b, the sync/async interface introduces two extra clock cycles in every data transmission. Several attempts have been made to reduce the synchronization delay by new synchronizer structures [36, 37] or by duplicating the interface to compensate the throughput loss [98]. Some asynchronous on-chip networks use synchronizer based interfaces [11, 98].

If long latency can be tolerated and high throughput is the major design concern, the sync/async FIFO scheme is the best choice. As shown in Figure 3.10c, a synchronous to asynchronous FIFO (S/A FIFO) and an asynchronous to synchronous FIFO (A/S FIFO) are inserted on forward and backward paths respectively. Unlike synchronizers, sync/async FIFOs have no throughput degradation, and the buffers inside the FIFOs balance the speed difference between synchronous and asynchronous sides. However, FIFOs introduce area overhead and prolong the synchronization latency. This scheme has been utilized in several asynchronous on-chip networks [106, 124].

No matter which interface scheme is selected, the channels between routers and network interfaces are pure asynchronous circuits. As a result, the routers in a mesh network are modular designs independent from their local PEs. Hence this thesis concentrates on the asynchronous architecture of routers without discussing further the interface problem.



(a) Interconnect network (b) SpiNNaker chip

Figure 3.11: SpiNNaker system

3.6 Previous GALS NoCs

Several GALS on-chip networks have been proposed and implemented in recent years. This section will review some of them. Descriptions of the router structures in these GALS NoCs are provided along with some analyses of their advantages and limitations.

3.6.1 SpiNNaker

The SpiNNaker system is a universal spiking neural network architecture aiming to simulate the human brain activities in real time [99, 85, 118]. As shown in Figure 3.11a, ten to thousands of SpiNNaker chips interconnect with each other in a mesh network. Every SpiNNaker chip is linked with its south (S), southwest (SE), west (W), north (N), northeast (NE) and east (E) neighbours. The channels between chips are 2-phase asynchronous pipelines using the 2-of-7 data encoding method [108].

The internal structure of a SpiNNaker chip is demonstrated in Figure 3.11b. 18 synchronous processors are connected by two on-chip networks: a communication NoC and a system NoC. As shown in the upper half of Figure 3.11b, the communication NoC is a part of the system-level interconnection network. The synchronous packet router in each chip receives spike signals from its six neighbours and the local 18 PEs

[130]. These signals are delivered by 4-phase 3-of-6 pipelines and are merged by a tree of bandwidth aggregators [98]. Running at $200\ \mathrm{MHz}$, the packet router is capable of processing all spike signals and forwarding them to their destinations. The system NoC is shared by the 18 local PEs inside a chip. Implemented using an asynchronous CHAIN network [3, 48], it allows PEs to access the off-chip memory concurrently.

The SpiNNaker system demonstrates an example of how asynchronous communication systems can be smoothly integrated with traditional synchronous bus systems. The inter-chip 2-phase 2-of-7 asynchronous channels reduce power consumption and communication delay between chips. On-chip communications are delivered by 4-phase 3-of-6 asynchronous channels leading to simple control circuits. Complicated computations are handled by synchronous circuits. Neurons and spike signals are simulated and generated by synchronous processors. The routing and broadcasting of spikes is calculated in the synchronous packet router.

However, in the aspect of networks, only the inter-chip interconnection network is fully scalable. Both on-chip networks use central communication structures whose throughput is limited by the central communication devices, such as the packet router in the communication NoC and the SDRAM interface in the system NoC. The on-chip networks would be saturated when the accumulative throughput requirement of all PEs exceeds the throughput capability of the central communication devices. Such central communication fabric is less scalable to support a large number of PEs than the on-chip networks using mesh topology.

3.6.2 ASPIN

The asynchronous scalable programmable interconnection network (ASPIN) is an asynchronous packet-switched on-chip network supporting clustered and shared memory MPSoCs [1, 107]. It is a direct mesh network. An ASPIN router is depicted in Figure 3.12 [105]. It uses the basic wormhole flow control method. Its switches have been simplified by removing disabled turns defined in the XY routing. The router comprises ten hard macros: an input module and an output module in each direction. Data between hard macros are delivered by 4-phase dual-rail pipelines. An input module includes a FIFO and an intermediate pipeline stage (IPS). The FIFO contains several bundled-data pipeline stages controlled by the fully decoupled control circuit [49]. Extra dual-rail to bundled-data and bundled-data to dual-rail converters are added before and after the FIFO respectively. IPS is a simple dual-rail pipeline stage inserted to cope with the long wire delay between input and output modules. An output module is a

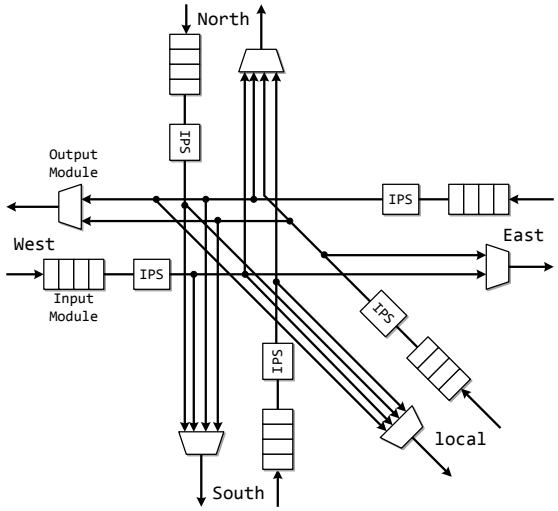


Figure 3.12: ASPIN router

multiplexer connecting input modules to the output port. The multiplexer is controlled by a ring arbiter [75].

ASPIN is a fast and area efficient NoC system. Long wire delays between input and output modules have been compensated by inserting intermediate pipeline stages. Area consuming buffers are replaced by area efficient bundled-data pipelines. The fully decoupled control circuit further doubles the storage efficiency. Complicated dual-rail control circuits have been manually designed in the fully customized hard macros. The ten macros can be carefully placed to balance long wires and improve throughput. The estimated period of such NoC using a 90 nm technology was reported 0.88 ns, which is equivalent to 1.131 GHz [105]. It is, so far, the fastest asynchronous NoC reported.

However, the fully customized hard macros and special designed asynchronous cells are not usually affordable in standard cell based design flows. Ten hard macros per router also implies a heavy burden in chip floorplanning. The reported 0.88 ns period was extracted from a claimed latency accurate simulation, which uses transport delay models where gate delays are scaled down to 90 nm and are fixed without considering the output load and input transition time of individual gates. The actual period could be much longer due to the scaling process and the inaccurate gate delay estimation.

3.6.3 QoS NoC

The QoS NoC [47, 45] is the only published asynchronous VC router that strictly follows the structure of synchronous VC routers shown in Figure 3.6. All data channels

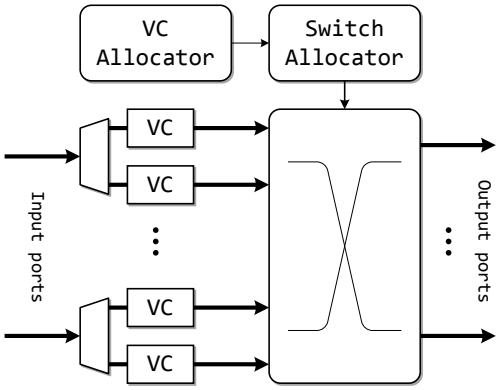


Figure 3.13: ANOC router

are QDI 4-phase 1-of-4 pipelines. Every input buffer has four VC buffers assigned with different priorities. The SPA arbiters inside output ports [46] always allocate the output port to the VC with the highest priority in all concurrent requests. This router is reproduced to support best-effort traffic in Appendix B.

QoS NoC is an early effort to provide QoS in asynchronous on-chip networks. The full QDI implementation consumes low power and provides tolerance to variations. However, the period of a 8-bit QoS NoC router in 0.18 $\mu\mathrm{m}$ is around 4 ns [45], which is much slower than ASPIN. It will be shown in Chapter 5 that VC compromises throughput significantly.

3.6.4 ANOC

The asynchronous network-on-chip (ANOC) [7] is also a QDI VC router providing QoS support. The ANOC router shown in Figure 3.13 and the router in the QoS NoC have similar structures except for the central switch. In the QoS NoC router, VCs in one input buffer are arbitrated and multiplexed; therefore, the central switch is a 5×5 crossbar. On the other hand, the ANOC router connects all VCs to the central switch. As every direction is equipped with two VCs, the central switch is a 10×5 crossbar. VCs in one direction are also assigned with different priorities and output ports arbitrate these VCs using static priorities, which is similar to the QoS NoC.

The extended switch in ANOC improves throughput. The ANOC router has been implemented using a 0.13 $\mu\mathrm{m}$ standard cell library with an augmented asynchronous cell library [78]. The router has been packaged into a hard IP core and utilized in the MAGALI and the FAUST chips [26]. The period reported in [7] is around 4 ns.

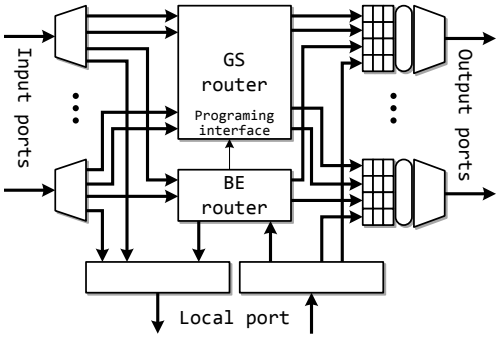


Figure 3.14: MANGO router

It has the same throughput problem as the QoS NoC. The augmented asynchronous cell library is another design problem. Building asynchronous circuits using only synchronous standard cell libraries is more adaptable although using special asynchronous cells significantly improves power, area and speed [78].

3.6.5 MANGO

The message-passing asynchronous NoC providing guaranteed service through OCP interfaces (MANGO) [11, 12, 13] is the first asynchronous NoC providing circuit-switched guaranteed services while also supporting packet-switched best-effort traffic. The router structure is shown in Figure 3.14. Every router has a local port and four other ports. The local port supports five services levels: four different GS service levels and one BE service. Other ports use VC buffers for different services and eight VCs are implemented (seven GS VCs and one BE VC). Unlike other input-buffered asynchronous routers, MANGO routers are output-buffered. VC buffers are thus implemented inside output ports. The central switch is expanded like the one in ANOC but is divided into two separated switches: a GS router for circuit-switched GS services and a BE router for packet-switched BE traffic. The GS router is dynamically configured by packets from the BE router. The eight VCs in each output ports are dynamically scheduled by an asynchronous latency guarantee algorithm, which provides latency and throughput guarantees for all GS VCs. All circuits use self-timed bundled-data protocols. Implemented using a $0.12\ \mu\mathrm{m}$ standard cell library, the MANGO router runs at equivalent $795\ \mathrm{MHz}$ (around $1.26\ \mathrm{ns}$ period).

Compared with other asynchronous NoCs that provide QoS using VCs, MANGO guarantees a maximum frame delay for each service level. The asynchronous latency guarantee algorithm running in each output port ensures that VCs with higher priorities

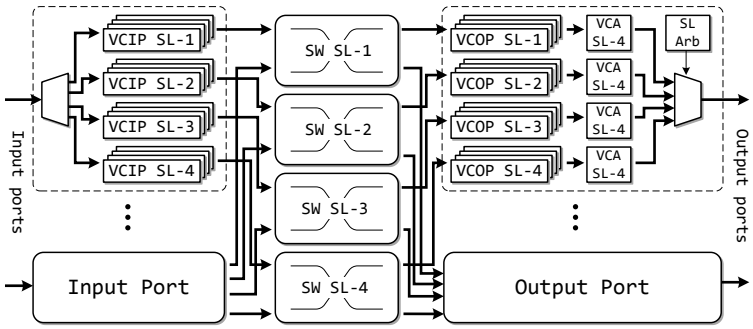


Figure 3.15: QNoC router

cannot starve other VCs with lower priorities [13]. On the contrary, the static priority arbiters utilized in QoS NoC and ANOC allow the highest QoS level to occupy a shared channel as long as it is busy.

However, the output-buffered switch introduces significant area overhead and is the most area consuming part in the router. The self-timed bundled-data implementation is less tolerant to variations than QDI implementations.

3.6.6 QNoC

The quality-of-service NoC (QNoC) [38] provides multiple VCs in each service level. The router structure is depicted in Figure 3.15. Its structure is similar to the ANOC router shown in Figure 3.13 but every service level (SL) has several equal-priority VCs. The central switch is expanded to allow the input VCs (VC-IP) in one service level to communicate directly with the output VCs (VC-OP) in the same service level. The VC arbiter (VCA in Figure 3.15) in each service level selects one VC-OP from all active VC-OPs every cycle. The SL arbiter in each output port selects one VC using a static priority arbiter. All buffers and channels use self-timed 4-phase bundled-data pipelines. The period of a QNoC router implemented using a $0.18\ \mu\mathrm{m}$ standard cell library is around $4.8\ \mathrm{ns}$.

QNoC demonstrates an example of providing multiple equal-priority VCs for every service level. In this way, multiple equal-priority communications can share the same router and traverse concurrently. However, it is not clear that such capability is worthwhile in MPSoC systems because VCs introduce large area overhead.

3.7 Summary

Network-on-chip is the state-of-the-art communication fabric in current MPSoC systems. Data delivered in a NoC are encapsulated into packets. According to the network topology, a packet travels from its sender to the receiver over a path selected by a routing algorithm. Multiple packets may compete for network resources during transmission. Various flow control methods are therefore implemented to resolve contention. The structure of a router is determined by the topology and the flow control method selected by the NoC. Asynchronous NoCs are GALS networks where synchronous PEs are connected by a global asynchronous on-chip network. The sync/async interface issues are handled by the network interface inside every PE. Several asynchronous on-chip networks have been proposed and implemented in recent years.

This thesis concentrates on improving throughput by exploring spatial parallelism in the router structure. Channels and flow control methods are the major research objectives. Because most MPSoC systems utilize mesh networks, all the router designs in this thesis assume that the mesh topology is used.

Part II

Levels of Parallelism

Chapter 4

Parallelism in the Physical Layer

Asynchronous channels and pipelines are the basic elements of asynchronous on-chip networks. The throughput of an asynchronous circuit is determined by the slowest pipeline stage. This chapter tries to introduce spatial and timing concurrency into the circuits of the physical layer. Two new techniques, channel slicing and lookahead pipeline style, are proposed to reduce the period of basic asynchronous pipelines.

4.1 Synchronization overhead

Many asynchronous pipeline styles have been utilized in asynchronous on-chip networks. Self-timed 4-phase bundled-data pipelines have been used in MANGO [12], ASPIN [105] and QNoC [38]. QDI 4-phase dual-rail pipelines have been used in ASPIN. QDI 4-phase 1-of-4 pipelines have been used in ANOC [7] and QoS NoC [47]. QDI 4-phase and 2-phase m-of-n pipelines have been used in SpiNNaker [99]. As described in Section 2.4, bundled-data pipelines need detailed timing analysis and careful delay insertion. Compared with 4-phase pipelines, 2-phase pipelines reduce period but introduce complicated combinational circuits (see Section 2.3). QDI 4-phase pipelines are promising candidates for asynchronous on-chip network due to their moderate area overhead and tolerance to variations.

In all asynchronous routers using QDI 4-phase pipelines, wide pipelines are built by synchronizing multiple bit-level pipelines as the 4-bit dual-rail pipeline shown in Figure 4.1. The synchronized pipelines are easy to control because data on all bit-level pipelines (slices) are synchronized. However, the completion detection (CD) circuit, normally a C-element tree, introduces extra delay and prolongs the period.

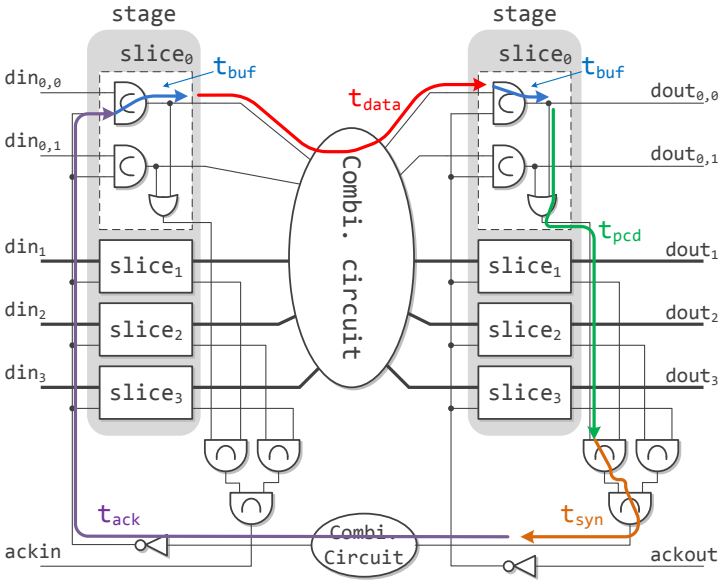


Figure 4.1: 4-bit 4-phase dual-rail pipeline

Assuming rising time and falling time are equal, the period T of the 4-phase dual-rail pipeline depicted in Figure 4.1 can be calculated as:

$$T = 4t_{\text{buf}} + 2(t_{\text{pcd}} + t_{\text{syn}}) + 2t_{\text{data}} + 2t_{\text{ack}} \tag{4.1}$$

In this equation, t_{buf}, t_{data} and t_{ack} are the delays of a buffer cell, the combinational circuit on data wires and the combinational circuit on ack wires respectively. t_{pcd} and t_{syn} are the delays of the completion detection circuits inside one slice and among slices (the C-element tree). The buffer cells in multi-rail pipelines are normally 2-input C-elements. Thus the buffer latency t_{buf} is equal with the latency of a 2-input C-element t_{C}. For a wide dual-rail pipeline of N bits, the latency of the C-element tree is:

$$t_{\text{syn}} = (\log_2 \left\lceil \frac{N}{2} \right\rceil) \cdot t_{\text{C}} \tag{4.2}$$

4-phase pipelines use return-to-zero signalling. The period T includes a positive cycle for data transmission and a negative cycle for data withdrawal. Assuming gate rising time and falling time are equal, T is obtained by doubling the loop path latency.

To evaluate the delay overhead of synchronizing bit-level pipelines (or slices), the period is divided into two parts: the period of a single dual-rail pipeline ($T_{\text{dual-rail}}$), and the extra delay caused by the synchronization (Δ). In Figure 4.1, Δ is the extra delay caused by the C-element tree ($2t_{\text{syn}}$). In this way, synchronization overhead can

be represented as:

$$\frac{\Delta}{T_{\mathrm{dual-rail}}} = \frac{\log_2 \lceil \frac{N}{2} \rceil \cdot t_{\mathrm{C}}}{2t_{\mathrm{C}} + t_{\mathrm{OR}} + t_{\mathrm{data}} + t_{\mathrm{ack}}} \tag{4.3}$$

where t_{OR} is the completion detection delay in a single dual-rail pipeline — the delay of a 2-input OR gate.

In Equation 4.3, the denominator $T_{\mathrm{dual-rail}}$ is the minimum period without synchronization, which is independent of N. The numerator Δ is the synchronization overhead, which increases logarithmically with N. Increasing the data width of a wide pipeline leads to slow pipelines. From another aspect, routers are not computation centric circuits. The combinational circuit between two pipeline stages are simple multiplexers and de-multiplexers. Their delays t_{data} and t_{ack} are small, as well as the minimum period $T_{\mathrm{dual-rail}}$. Synchronization overhead is significant in simple circuits like asynchronous routers.

Increasing the wire count of a single 1-of-n pipeline or m-of-n pipeline seems to increase the total data width without introducing synchronization overhead. However, a 1-of-n pipeline using more than five data wires is not area efficient. The completion detection circuit of a single m-of-n pipeline introduces a similar delay overhead.

4.2 Channel slicing

The pipelined completion process [77] is a QDI pipeline style which moves the completion detection circuit outside the loop path of pipeline stages. A simplified implementation of the pipelined completion process in a 2-bit dual-rail pipeline is presented in Figure 4.2. Compared with normal synchronized dual-rail pipelines (Figure 2.7), extra C-elements (coloured in yellow) are inserted between slices and the synchronization circuits. Each slice has its own ack line generated by these extra C-elements. Hence individual slices generate acknowledge signals in parallel with pipeline synchronization. To ensure the data transmitted on all slices are still synchronized, the extra C-elements are guarded by the result of the synchronization C-element tree. An extra synchronization C-element tree (coloured in grey) is also inserted in the sending pipeline stage to guarantee that data are not released before they are captured.

The pipelined completion process can be applied to all QDI pipelines even with complicated combinational circuits. Individual slices have their own ack lines and are semi-decoupled from each other. The data sent by different slices in the same stage are

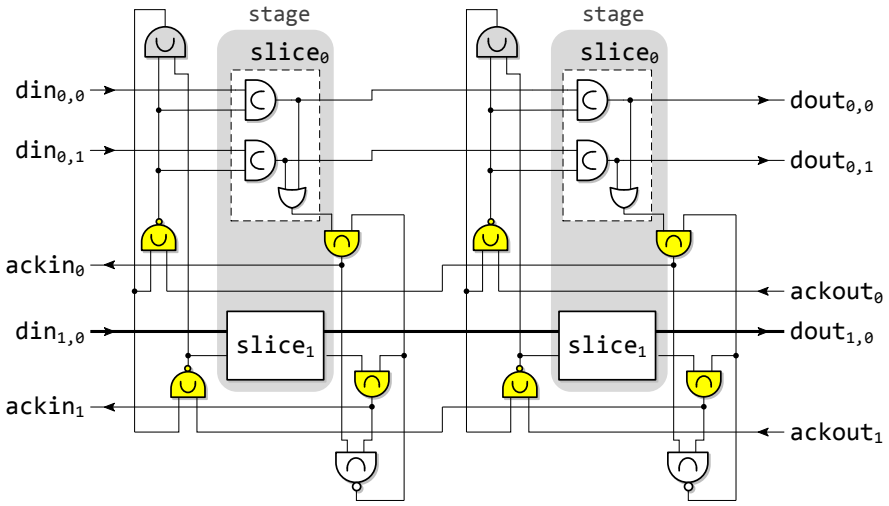


Figure 4.2: Pipelined completion process

still synchronized. Extra C-elements and an extra C-element tree are used to guarantee synchronization while decoupling pipelines.

Similar to the pipelined completion process, channel slicing is an aggressive technique that fully decouples slices whenever possible. An abstract implementation of channel slicing is shown in Figure 4.3. The vertically divided pipelines where data can be delivered independently, namely sub-channels, are fully decoupled. Every sub-channel has its own ack line. The synchronization C-element tree is removed in all stages and no extra C-elements are inserted. For the pipeline stages where complicated combinational circuit is implemented or control is data dependent, extra sub-channel control circuits are added on each slice in this stage to temporarily stop individual sub-channels if synchronization is required on certain occasions. The pipeline control circuit receives the data from all sub-channels along with the ack lines. Synchronization can be regenerated when necessary.

Compared with the pipelined completion process, channel slicing is expected to introduce much lower area overhead. Extra control circuits and synchronization circuits are inserted only in the stages requiring synchronization. These circuits re-establish the synchronization only when it is necessary. Thus the period of channel slicing is shorter than that of the pipelined completion process most of the time because the ack signals are directly generated by sub-channels. However, the circuit implementation of the pipeline control circuit tightly depends on the local logic function while the pipelined completion process generally fits all pipeline situations.

Channel slicing is suitable for asynchronous routers for two reasons: (1) Most

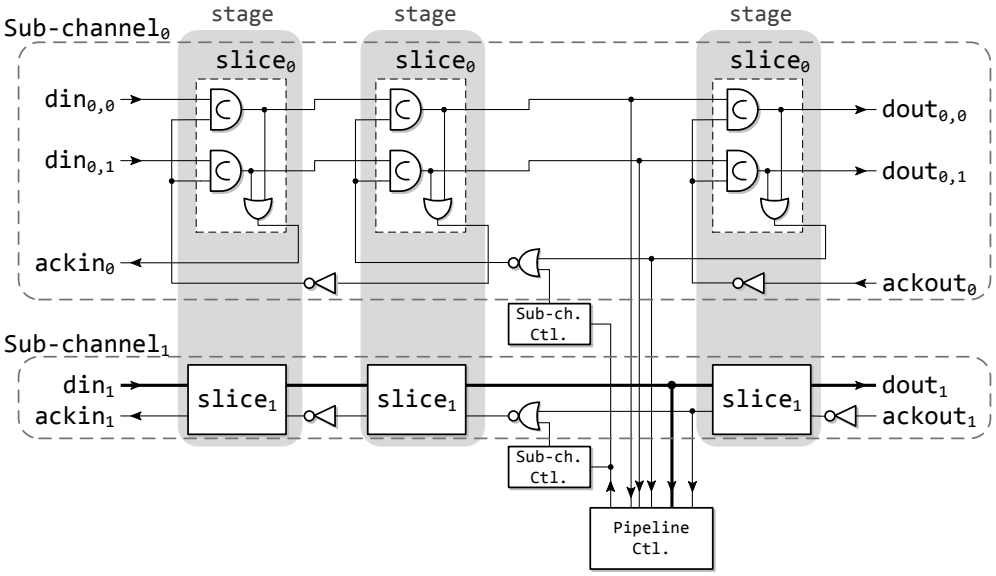


Figure 4.3: Channel slicing

pipeline stages in an asynchronous router are simple pipelines without complex combinational calculation or data related control logic. (2) For a frame, only the header flit and the tail flit need to be detected and analysed; data flits, which are the main body of a frame, are normally delivered without reading them.

The major design issue of using channel slicing in asynchronous routers is re-synchronization. Unlike the pipelined completion process, sub-channels in channel sliced pipelines are fully decoupled. Without re-synchronization, parts of a new flit can be transmitted by fast sub-channels while the old flit is under transmission by slow sub-channels as shown in Figure 4.4, where two frames are transmitted by a 4-bit channel sliced pipelines. Routers need to analyse the header flit for routing decisions. Thus sub-channels should be re-synchronized when the header flit arrives. After a path is reserved in the central switch, sub-channels deliver data independently at their fastest speeds. The detailed implementation of channel slicing in a wormhole router will be revealed in Section 4.4

4.3 Lookahead pipeline style

The data paths of asynchronous on-chip networks can be simplified into pipeline stages and switches as shown in Figure 4.5. Several pipeline stages are placed in every input buffer and output buffer. Input buffers and output buffers are connected by switches inside routers and long wires outside routers.

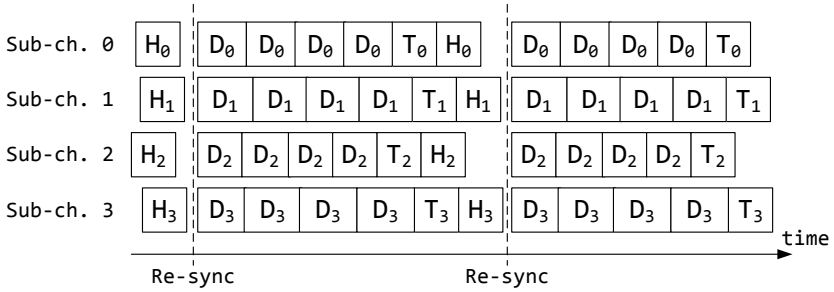


Figure 4.4: Data flow with channel slicing

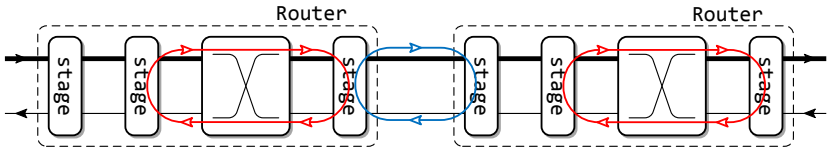
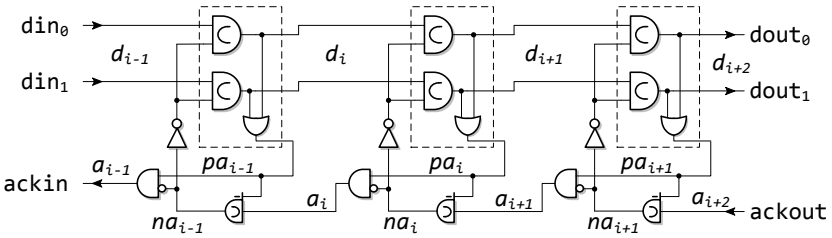


Figure 4.5: Critical cycles in asynchronous on-chip networks

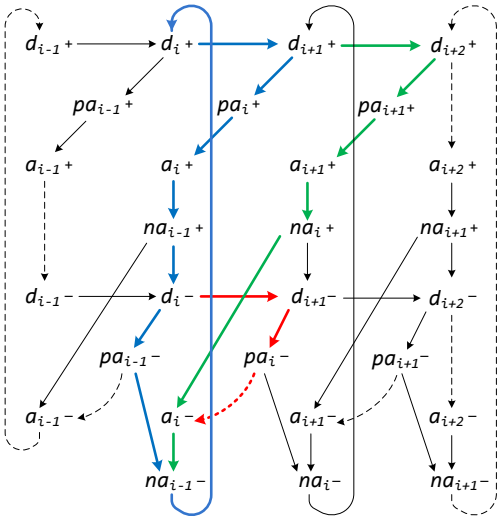
The throughput of synchronous circuits is determined by the slowest data path — the critical path. Likewise, the throughput of asynchronous circuits is determined by the slowest loop path between two adjacent pipeline stages — the critical cycle. Data paths of asynchronous on-chip networks normally have two types of critical cycles as shown in Figure 4.5: the loop path through the central switch inside a router (coloured in red) and the loop path between two adjacent routers (coloured in blue). The loop paths between routers are comparatively easy to handle if they are the throughput bottleneck. More pipeline stages can be inserted on long wires to reduce the period. It is also preferable to use other asynchronous pipeline styles if allowable, such as 2-phase pipelines [108], high-speed QDI pipelines [93] and wave-pipelined data links [39]. On the other hand, the loop path through the central switch is difficult to cope with. No intermediate pipeline stages can be inserted and using other pipeline styles complicates the switch control logic. This is the critical cycle that needs special treatment.

The lookahead pipeline [111] is a relaxed QDI pipeline style providing fast data rates by allowing the data wires and the ack line to be reset simultaneously. Since a router is normally shipped in the form of a hard IP core in large GALS projects, using relaxed QDI pipelines inside a router introduces no extra design burdens outside the router IP but improves throughput.

A dual-rail lookahead pipeline and its STG are shown in Figure 4.6. It has the basic structure of a QDI dual-rail pipeline but extra AND gates and C-elements are added on ack lines. The AND gate in pipeline stage i allows the ack signal a_i to be reset when pipeline stage $i + 2$ has safely captured the data transmitted in the current cycle;



(a) Dual-rail lookahead pipeline



(b) Signal transition graph

Figure 4.6: Lookahead pipeline

therefore, stage i can receive a new data while stage $i+1$ is releasing its data buffers. The C-element ensures that the ack signal is not reset too fast, otherwise stage $i+1$ fails to receive new data.

The critical cycle of this dual-rail lookahead pipeline is coloured in blue in the STG (Figure 4.6b) while part of the original critical cycle of a QDI dual-rail pipeline is coloured in red. Since the dependence between a_i- and pa_i- does not exist in lookahead pipelines, it is drawn as a dotted line. In the original QDI pipelines, ack line a_i is reset after the release of data d_{i+1} indicated by pa_i-. The lookahead pipelines allow d_{i+1} to be withdrawn concurrently with a_i-, which reduces period by 25% in the best case. However, this STG is not speed-independent. Transition a_i+ and transition a_i- are located in two parallel paths coloured in red and green. The STG itself cannot ensure that a_i- occurs after a_i+. Two timing assumptions must be satisfied for the correct operation [114, 111].

Ack setup time: Expressed in Equation 4.4, the first timing assumption ensures that

the positive pulse on a_i is long enough for the asymmetric C-element to capture it.

$$t_{d_{i+1}+\rightarrow a_i-} - t_{d_{i+1}+\rightarrow a_i+} > t_{setup} \tag{4.4}$$

where t_{setup} is the setup time for a C-element (an asymmetric C-element in this case). Assuming all C-elements incur the same cell delay, Equation 4.4 can be expressed in gate delays and simplified into:

$$2 \cdot t_C + t_{AND} > t_{setup} \tag{4.5}$$

where t_C and t_{AND} are the delays of a C-element (or an asymmetric C-element) and an AND gates respectively. Obviously Equation 4.5 is easy to satisfy in gate-level implementations.

Data override: Since the reset of a_i occurs concurrently with the release of d_{i+1}, the new data on d_i must not arrive before the old data on d_{i+1} are released. Assuming $t_{d_i+\rightarrow d_{i+1}+}$ is equal with $t_{d_i-\rightarrow d_{i+1}-}$, this timing requirement can be expressed as:

$$t_{d_i-\rightarrow na_i-} + t_{na_i-\rightarrow d_i+} > t_{setup} \tag{4.6}$$

which can be further described in gate delays:

$$t_{CD} + 2 \cdot t_C + t_{INV} > t_{setup} \tag{4.7}$$

where t_{CD} is the delay of the completion detection circuit in each pipeline stage (an OR gate in dual-rail pipelines). Since the left side of Equation 4.7 is half the period of the fastest dual-rail pipeline, it is longer than the setup time of a C-element. The data override assumption is already met by hardware.

4.4 A channel sliced wormhole router

In this section, a wormhole router using channel sliced pipelines and the lookahead pipeline style in its internal critical cycle is designed and implemented to demonstrate the throughput improvement along with design overhead [114].

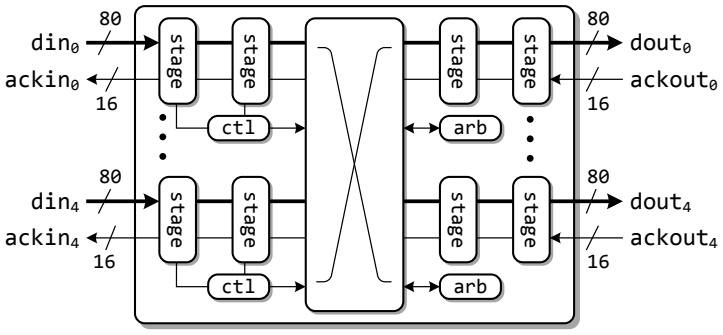


Figure 4.7: A channel sliced wormhole router

4.4.1 Router structure

The internal structure of a channel sliced wormhole router is shown in Figure 4.7. It has five bidirectional ports for four adjacent neighbours and the local PE. A 2-stage pipelined buffer is connected to each input and output port. Input buffers and output buffers are connected by a central switch controlled by the arbiters in the output ports. Besides buffers, every input port has a router analysing the target address of incoming frames using the XY routing algorithm and a controller sending routing requests to the target output arbiters. In this implementation, each port delivers 32 bits per cycle using 16 4-phase 1-of-4 pipelines. 1-of-4 pipelines are preferred as they consume less energy than dual-rail pipelines and less area than m-of-n pipelines (Section 2.5) [114]. Using the channel slicing technique, every 1-of-4 pipeline is a sub-channel with its own sub-channel controller.

According the wormhole flow control method, a frame is divided into flits. The flit format is explained in Table 4.1. The address of the target node is represented in eight binary digits which are enough to identify a 16×16 mesh network. The X and Y addresses are further translated into 1-of-4 codes and stored in the least significant bits of the header flit. The rest of the header flit is used for data. A variable number of data flits may follow the header. Following the data payload, the frame is ended by a tail flit indicating the end of a frame (EOF), in which the *eof* bit of every sub-channel is set high.

As described in Section 4.2, all pipeline stages are spatially divided into unsynchronized sub-channels. In the wormhole router implementation, only the second pipeline stages of each input buffer are synchronized when the tail flit of the preceding frame is transmitted and the header flit of the new frame is going to be analysed. A sub-channel controller is inserted in the second pipeline stage of each sub-channel and a

Table 4.1: Flit format

	sub-channels				
flit type	0	1	2	3	4 ∼15
header	tar_X[1:0]	tar_X[3:2]	tar_Y[1:0]	tar_Y[3:2]	byte[2:0]
data *i*	byte[$4i+7$:$4i+3$]				
tail	EOF	EOF	EOF	EOF	EOF

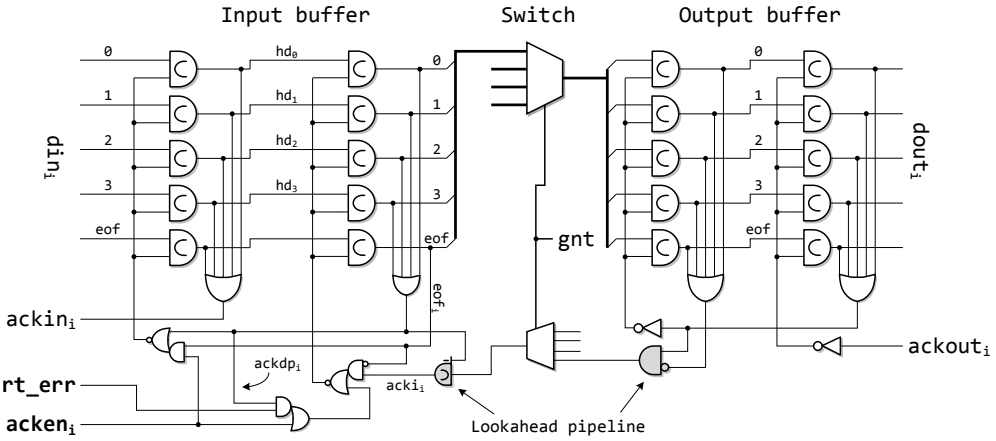
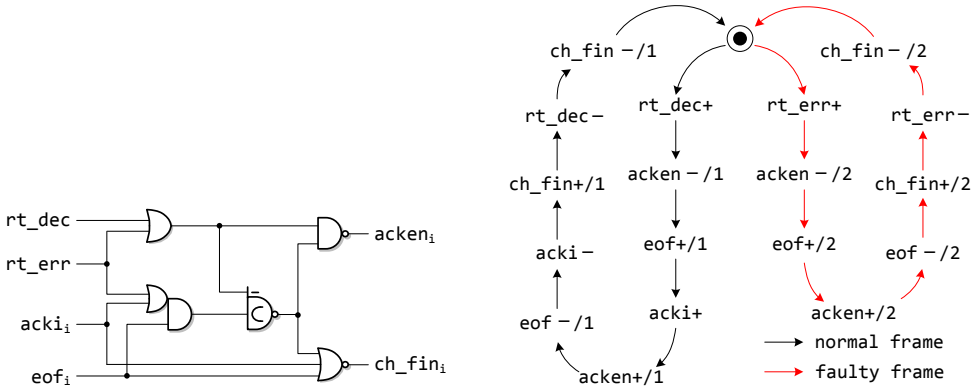


Figure 4.8: Data path of the ith sub-channel

router controller is inserted in each input buffer to control the XY router and establish synchronization.

The circuit structure of the ith single sub-channel in both input and output buffers is demonstrated in Figure 4.8. The second pipeline stage is controlled by two signals in the bottom left corner of Figure 4.8: *acken*$_i$ and *rt_err*. The active low signal *acken*$_i$ is driven by the sub-channel controller of each sub-channel. A negative *acken*$_i$ allows the second pipeline stage to capture new data. It is driven high to block the sub-channel when a header flit is expected or is being analysed. If a decoded routing request is not valid due to wrong addresses or on-chip faults, *rt_err* is driven high denoting the frame should be dropped. In this case, *acken*$_i$ is still driven low to activate the sub-channel but the ack line generated by the second pipeline stage (*ackdp*$_i$) is connected back to the second pipeline stage. The ack line from the output buffers (*acki*$_i$) is inactive during the process as no valid routing request is produced. The second pipeline stage is converted into a flit sink consuming all flits until the tail flit.

The tail flit of each frame is detected concurrently in all sub-channels, regardless of the validity of the routing request. Once a sub-channel has delivered its share of the



(a) Implementation (b) STG

Figure 4.9: Sub-channel controller

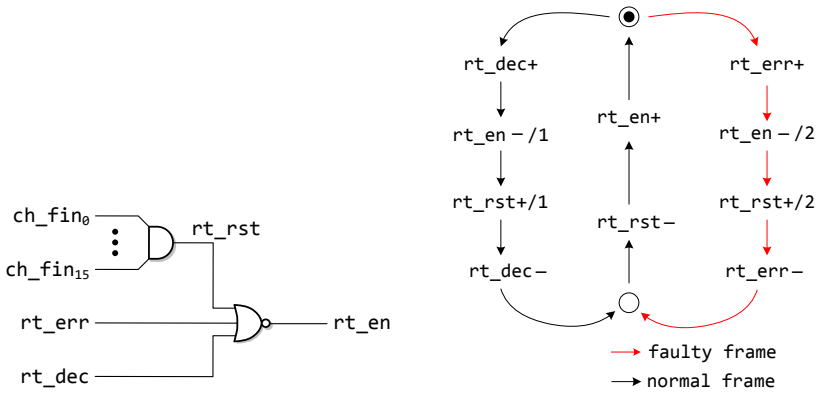
tail flit, indicated by the eof_i bit, its $acken_i$ is driven high. Consequently, the next flit is captured and blocked in the first pipeline stage forcing re-synchronization.

The lookahead pipeline style is utilized in the pipeline stages connected to the central switch — the critical cycle (see Figure 4.5). An AND gate is added in the first stage of the output buffer generating the early evaluated ack signal ($acki_i$). An asymmetric C-element is added on this ack line on the receiver side ensuring new data do not override the old data. This implementation exactly follows the dual-rail lookahead pipeline shown in Figure 4.6.

The $acken_i$ signal of an individual sub-channel is generated by the sub-channel controller shown in Figure 4.9. This control circuit reads four inputs: the correct routing decoding flag *rt_dec* and the faulty routing decoding flag *rt_err* from the XY router in each input port, the ack line $acki_i$ from the central switch, and the eof_i bit of the second pipeline stage. Two signals are generated: the active low ack enable signal $acken_i$ and the frame termination flag ch_fin_i which is high when the sub-channel has delivered its share of the tail flit. The implementation depicted in Figure 4.9a is synthesized from the STG shown in Figure 4.9b using Petrify [29].

As shown in the STG, a sub-channel is enabled after a decoded routing request, either right or wrong, and remains active until the tail flit is transmitted. The second pipeline stage releases the tail flit after it is blocked, making sure no further flits are transmitted. The frame termination flag ch_fin_i is set after the release of the tail flit and unset when the previous routing request is withdrawn.

Every input port has an XY router analysing the header flit of incoming frames. It is controlled by the router controller demonstrated in Figure 4.10. It generates an active



(a) Implementation (b) STG

Figure 4.10: Router controller

high router enable signal *rt_en* by reading the routing requests *rt_dec* and *rt_err*, and the frame termination flags ch_fin_i from all sub-channels. The circuit implementation is also synthesized from the STG illustrated in Figure 4.10b. The router is enabled when all sub-channels have delivered and released their shares of the tail flit, and is disabled once a routing request is decoded, either right or wrong. The decoded request is captured by some C-elements inside the router.

Figure 4.11 depicts the XY router in the south input port and its connection with the arbiter inside the east output port. The target addresses *tar_X* and *tar_Y* are read from $hd_0 \sim hd_3$ (Figure 4.8), which are the output of the first pipeline stage in the input buffer. Note that *rt_en* is only high when the second pipeline stage has released the tail flit and been disabled. Therefore, the data on $hd_0 \sim hd_3$ must be the header flit blocked in the first stage. The addresses are compared with the local addresses *loc_X* and *loc_Y* using two 1-of-4 comparators. They are implemented in one-hot circuits using a structure similar to the dual-rail comparators in ASPIN (Figure 18 in [105]). The comparison results are translated into routing requests and captured by the C-elements controlled by *rt_rst*, which resets the routing requests after releasing the tail flit. Valid routing requests are forwarded to corresponding arbiters in output ports while invalid requests trigger the *rt_err* signal and withdraw the frame. Signal *rt_dec* is set immediately after valid requests and remains high until corresponding arbiters in output ports are released.

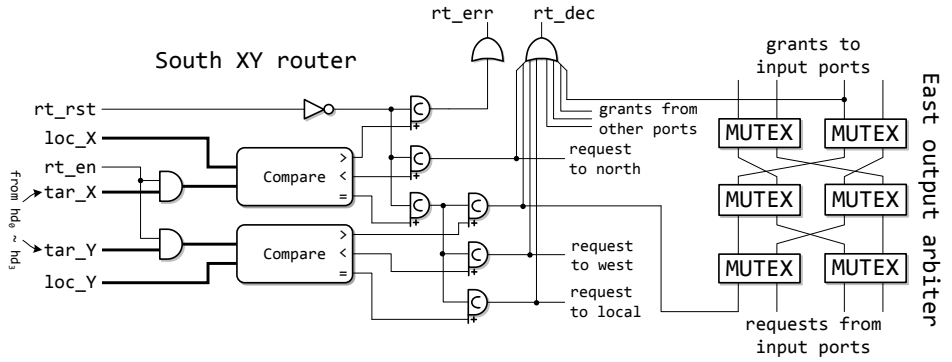


Figure 4.11: XY router and output arbiter

4.4.2 Performance

The channel sliced wormhole router has been implemented using the Faraday $0.13\ \mu\mathrm{m}$ standard cell library [44] based on the UMC $0.13\ \mu\mathrm{m}$ technology. All parts of the router are described in synthesizable Verilog HDL. The router controller and sub-channel controllers are synthesized from their STGs using Petrify [29] while other parts are manually written in gate-level netlists. The router has been mapped into the Faraday library cells using Synopsys Design Compiler. The netlist is further placed and routed using Synopsys IC Compiler. Synopsys Star-RCXT has been used to extract parasitic resistors and capacitors, which are used in Synopsys PrimeTime to generate a back-annotation standard delay format (SDF) file for post-layout simulation. All simulation results are collected under the typical corner ($25\ ^{\circ}\mathrm{C}$, $1.2\ \mathrm{V}$) with back-annotated delays.

The area after synthesis is around $14.3\ \mathrm{k}$ gates ($0.057\ \mathrm{mm}^2$). The period of data flits is $1.7\ \mathrm{ns}$ providing the maximum throughput of $18.8\ \mathrm{Gbit/Port/s}$. The average latency for a data flit traversing the router is $1.7\ \mathrm{ns}$. For the header flit, a minimum of $0.8\ \mathrm{ns}$ is required to analyse the target address and request to the arbiter in the corresponding output port.

Besides the channel sliced (ChSlice) wormhole router using lookahead (LH) pipelines, a traditional wormhole router using synchronized pipelines and a wormhole router using channel slicing only have been implemented. The area of these routers is presented in Table 4.2.

Channel slicing introduces 23% area overhead compared with the traditional router using synchronized pipelines. The area of the central switch increases due to the enlarged wire count. The synchronized 1-of-4 pipeline needs 66 wires (64 data bits, one EOF bit and one ack line). Channel slicing uses independent sub-channels with

Table 4.2: Router area

	input buffer	output buffer	switch	overall
No ChSlice or LH	4.3 k	4.4 k	2.4 k	11.3 k
ChSlice only	5.8 k	4.5 k	3.2 k	13.9 k
ChSlice and LH	6.2 k	4.5 k	3.3 k	14.5 k

(unit: equivalent gate)

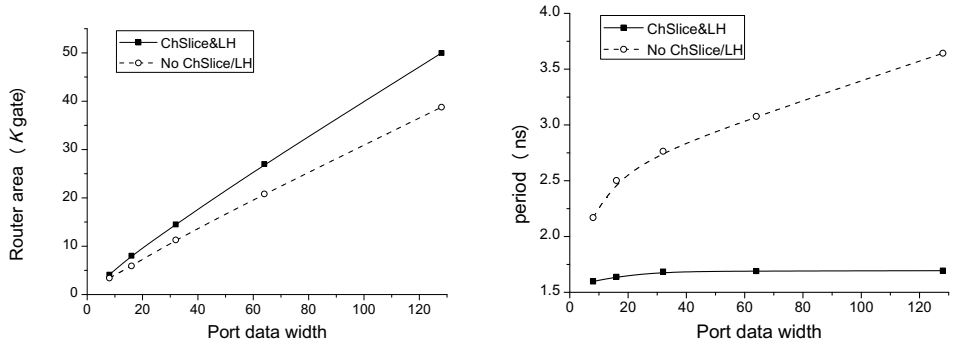
Table 4.3: Speed performance

	period	equ. frequency	router latency	routing
No ChSlice or LH	2.9 ns	345 MHz	2.8 ns	0.8 ns
ChSlice	2.2 ns	455 MHz	2.1 ns	0.8 ns
ChSlice and LH	1.7 ns	588 MHz	1.7 ns	0.8 ns

individual EOF and ack lines. The total wire count is increased to 96 (64 data bits, 16 EOF bits and 16 ack lines). Consequently the switch is enlarged. The total number of C-elements in channel sliced pipelines and synchronized pipelines are equal. Although an extra C-element is added in each individual sub-channel for the EOF bit, this area overhead is compensated by the removal of the synchronization circuit. The area of output buffers does not vary in all routers. The area overhead of input buffers is mainly due to the sub-channel controllers. Since sub-channel controllers are added only in the last pipeline stage in the input buffers, increasing the length of buffers introduces no further area overhead.

The lookahead pipeline style brings no significant area overhead. Only the area of input buffers is slightly increased. It is believed that the asymmetric C-element added on *acki* in Figure 4.8 is the major cause. This C-element is located on the critical cycle and its driving strength has been optimized for speed performance.

As shown in Table 4.3, channel slicing and lookahead pipelines improve throughput significantly. These speed results are produced by averaging the period and router delays of sending a frame from the south input to the east output, which has the worst router delay. Channel slicing and lookahead pipelines reduce the period by 24.1% and 17.2% respectively. The router using both channel slicing and lookahead pipelines delivers data flits at equivalent 588 MHz, which is 1.7 times of the equivalent frequency of the traditional router using synchronized pipelines.



(a) Router area (b) Period

Figure 4.12: Area and speed with various data widths

A major disadvantage of using synchronized pipelines is the deteriorated throughput with increasing data width. The sub-channels in channel sliced pipelines are unsynchronized most of the time. Unlike synchronized pipelines, increasing data width leads to no significant throughput degradation. Figure 4.12 shows the router area and speed with various data widths. The router area is linear with data width as expected. The period of the channel sliced router remains around 1.7 ns while it increases to more than 3.5 ns in the router using synchronized pipelines. Channel slicing is a desirable technique in wide QDI pipelines.

4.5 Summary

In this chapter, two novel techniques have been proposed to improve the pipeline throughput of asynchronous routers: channel slicing and the lookahead pipeline style. Channel slicing removes the synchronization among the sub-channels in one wide pipeline. Hence every individual sub-channel delivers data at its fastest speed without waiting on other sub-channels. When synchronization is required in some particular pipeline stages, sub-channel controllers are added in these stages to re-establish the synchronization whenever necessary. The lookahead pipeline style is a relaxed QDI pipeline implementation using early evaluated acknowledge signals. It allows a pipeline stage to capture new data concurrently with the release of old data in the succeeding pipeline stage. The timing assumptions required for correct operation are shown to be satisfied in gate-level implementations. The introduced concurrency reduces the period by a maximum of 25%.

A wormhole router has been implemented in layout using the channel slicing technique and the lookahead pipelines. Compared with the router using synchronized pipelines, the new router achieves 70% throughput improvement with 28% area overhead. The implementation results with various data widths demonstrate that increasing the data width in channel sliced routers leads to no throughput degradation but the degradation is significant in routers using synchronized pipelines.

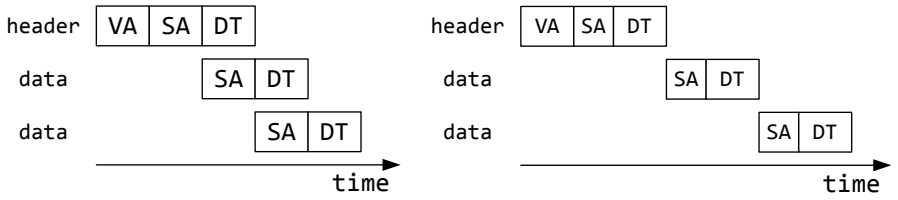
Chapter 5

Parallelism in the Switching Layer

The preceding chapter introduced two techniques which are ready for use in the physical layer to improve the throughput of asynchronous pipelines. This chapter proposes a new flow control method in the switching layer — spatial division multiplexing (SDM). The performance advantages and design overhead of SDM will be compared against the virtual channel (VC) flow control method, which is extensively adopted in most asynchronous on-chip networks. Behavioural level SystemC models have been built for routers using wormhole, SDM and VC. The simulation results show that SDM outperforms VC in throughput.

To avoid ambiguity, the terminologies used in this and following chapters are defined as follows:

Port *of a router*	All the I/O pins of a router pointing to a specific direction
Port *of a switch*	All the I/O pins of a switch connected to a specific component
Link	The wires and buffers between the ports of two routers
Frame	A packet generated by PEs
Flit	A part of a frame which can be transmitted in one period
Stage	All the buffers storing a single flit
Channel	The data path of a frame, including all the buffer stages in the links and routers on the way
Virtual circuit	A channel using only a portion of the full data width
Virtual channel	A virtual data path of a frame using the full data width in a time divided manner Or, the buffers in a router belonging to a virtual channel



(a) Synchronous VC (b) Asynchronous VC

Figure 5.1: Data flow of VC

5.1 Problems of the virtual channel flow control

Virtual channel (VC) is the most utilized flow control method in synchronous on-chip networks thanks to its capability of alleviating the HOL problem. However, utilizing VC in asynchronous on-chip networks introduces significant design overhead and the throughput improvement is far from satisfactory. Although VC has already been employed in numerous asynchronous on-chip networks [46, 12, 7, 38], all of them use VCs to provide QoS support rather than improve throughput. VC is not a desirable method for high throughput asynchronous on-chip networks.

An asynchronous VC router suffers from three different disadvantages: slow switch allocation, large area overhead, and long pipeline synchronization latency.

Switch allocation

The switch allocation in asynchronous VC routers significantly compromises throughput.

As shown in Figure 5.1a, the transmission of a flit in synchronous VC routers proceeds in three steps: VC allocation (VA), switch allocation (SA) and data transmission (DT) [35]. Only header flits require the VC allocation step during which the VC allocator reserves a VC in the target output port. Since all VCs share the central switch in a time divided manner, all flits need to compete for a path in the switch before transmitting data. In a synchronous VC router, all VCs are synchronized by the global clock. VA, SA and DT are pipelined and accomplished in three clock cycles. As shown in Figure 5.1a, a flit can pre-order a path while another flit is under transmission.

An asynchronous VC router cannot pipeline the switch allocation with data transmission as a path cannot be pre-allocated to a VC before it is released. As a result, the absolute time consumed by each flit is the accumulated delay of SA and DT rather than a single clock cycle in synchronous VC router. Figure 5.1b depicts the asynchronous

Table 5.1: Buffer area

	Number of Latches	Latch Type	Area
synchronous buffer	$D \cdot W$	flip-flop	$2820\ \mu m^2$
async dual-rail	$4D \cdot W$	C-element	$10140\ \mu m^2$
async 1-of-4	$4D \cdot W$	C-element	$8040\ \mu m^2$
async 2-of-7	$3.5D \cdot W$	C-element	$9760\ \mu m^2$

data flow. The latency of switch allocation has been directly added into the critical cycle. Network throughput is compromised significantly.

Area overhead

It is well known that buffers are the major area overhead in synchronous VC routers [87]. Asynchronous buffers consume more area than their synchronous counterparts, leading to an even larger area overhead.

Table 5.1 illustrates the area overhead of synchronous and asynchronous buffers using different data encoding methods. The second and third columns reveal the type and number of latches needed for a buffer to store D data flits with data width of W bits. All asynchronous buffers are 4-phase QDI pipelines using C-elements as storage components. Wide pipelines are built by synchronizing multiple bit-level pipelines. A single asynchronous pipeline stage is a half buffer stage. Two asynchronous pipeline stages are equivalent to one synchronous pipeline stage using flip-flops. A minimum of $2D$ pipeline stages are required for D flits. The C-element tree in the synchronization circuit introduces extra area overhead. The last column in Table 5.1 demonstrates the minimum area[1] of implementing buffers for four 32-bit flits ($D = 4, W = 32$) using the Faraday $0.13\ \mu m$ cell library [44]. Obviously, asynchronous buffers are significantly larger than synchronous buffers.

Synchronization overhead

As described in Section 4.1, synchronization overhead is the extra latency incurred by synchronizing multiple bit-level pipelines into a wide pipeline. In asynchronous VC routers, the central switch is re-allocated in every data cycle. Thus pipeline stages must be synchronized in every data cycle and channel slicing cannot be used.

[1] All gates are implemented using the smallest driving strength.

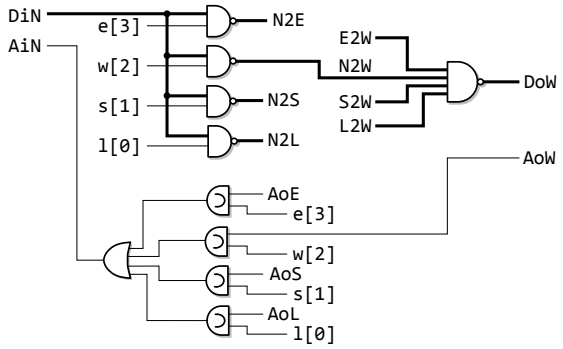


Figure 5.2: Crossbar in asynchronous VC routers

In addition to the synchronization overhead in pipelines, the frequent re-allocation of the central switch also introduces extra area and extra latency in the critical cycle. Figure 5.2 depicts the crossbar implementation in QoS NoC (reproduced from Figure 8.12 in [45]). Since the crossbar configuration signals (e[3], w[2],s[1] and l[0]) are withdrawn with data, extra C-elements must be inserted on the ack lines to ensure *AiN* drops after *AoW* (or other ack signals). If the path is allocated to a frame rather than a flit, such as the wormhole routers in Section 4.4, the configuration signals can be withdrawn after *AiN*$-$ and no extra C-elements are needed.

5.2 Spatial division multiplexing

The latency overhead of VC routers is related to synchronization — the synchronization among bit-level pipelines, and the synchronization between switch allocation and data transmission. Instead of introducing timing concurrency which leads to synchronization, the spatial division multiplexing flow control method brings spatial parallelism which alleviates the HOL problem without introducing extra synchronization. It has been utilized in synchronous on-chip networks to improve throughput [54] or to provide QoS support [71]. It will be shown that using SDM in asynchronous on-chip networks provides better best-effort throughput than using VC.

The structure of an SDM router is shown in Figure 5.3. Every port or buffer is spatially divided into multiple virtual circuits[2] [71]. Assuming the data width of each port is W bits and every port is divided into M virtual circuits, the data width of a single virtual circuit is W/M bits. Every virtual circuit delivers a frame independently in a serialized manner without sharing any resources with other virtual circuits. When

[2]To avoid the ambiguity between virtual channel and virtual circuit, the abbreviation VC represents virtual channel in the whole thesis.

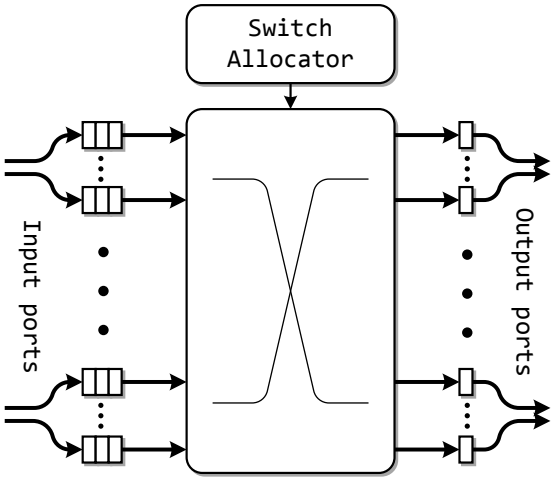


Figure 5.3: SDM router

one frame is blocked in one router, only the virtual circuit allocated to this frame is wasted. Other frames can go through the same port using other unblocked virtual circuits. Hence the HOL problem is alleviated.

SDM routers introduce no synchronization overhead. Since virtual circuits are exclusively allocated to frames, no resource is shared. The path reserved in the central switch holds for the whole transmission duration of a frame. Thus switch allocation is made once per frame introducing no extra latency. Furthermore, the data width of each virtual circuit is a portion of the port data width. SDM, in fact, reduces the synchronization overhead among bit-level pipelines.

Long frame latency and large switch area are the major design overhead of SDM routers. Since frames are serialized to fit the data width of a single virtual circuit, the frame latency is prolonged. The frame latency t_F can be expressed as:

$$t_F = t_R \cdot h + T \cdot l \cdot M + t_A \tag{5.1}$$

where t_R is the router latency; h is the number of hops between frame sender and receiver; T is the period of data paths; l is the number of flits in a frame; M is the number of virtual circuits in one port; t_A is the extra latency introduced by HOL problems. When the number of virtual circuits in one port increases, frame latency rises with the increasing M. However, t_A, T and t_R are reduced as the HOL problem is alleviated and the synchronization overhead is decreased with small data width. The frame latency overhead is not significant when the network size is large (large h), frames are short (small l) or the network is heavily loaded (large t_A).

SDM routers use large central switches. In an SDM router with P ports, $M \times P$ input virtual circuits are connected to $M \times P$ output virtual circuits leading to a $MP \times MP$ crossbar. The area of a crossbar is proportional to the number of cross-points[3] inside it. Accordingly, the area of the crossbars inside wormhole routers and SDM routers can be calculated as:

$$A_{\text{Wormhole}} = P^2W \cdot A_{CP} \tag{5.2}$$

$$A_{\text{SDM}} = M^2P^2(W/M) \cdot A_{CP} = MP^2W \cdot A_{CP} \tag{5.3}$$

where A_{CP} is the equivalent area of a single cross-point. The area of the crossbar in an SDM router is proportional to the number of virtual circuits.

Although SDM introduces large switches, this area overhead is expected smaller than that of VC routers for three reasons:

- The size of the buffers in SDM routers is not increased; therefore, the large central switch is the major area overhead. As shown in the area breakdown of the wormhole router in Table 4.2, switches are smaller than buffers. Hence increasing the size of switches leads to smaller routers than increasing the size of buffers, assuming the same number of VCs/virtual circuits are implemented.

- VC routers also increase the size of switches. If extended switches (Section 3.6.4) are used, such as ANOC [7], MANGO [12] and QNoC [38], the size of the switch is exactly the same as that of the SDM switch when the number of VCs is M. Even when the normal $P \times P$ switch is used as in the QoS NoC router [47], extra multiplexers are added on the outputs of input buffers leading to an area overhead of $MPW \cdot A_{CP}$.

- As it will be presented in Section 6.4, utilizing a 2-stage Clos switch reduces the size of the switch to

  $$A_{\text{SDM+Clos}} = (P^2 + MP)W \cdot A_{CP} \tag{5.4}$$

  which introduces the same area overhead as the extra multiplexer in QoS NoC routers.

[3] A cross-point is a single switching element in an array of elements that comprises a switch.

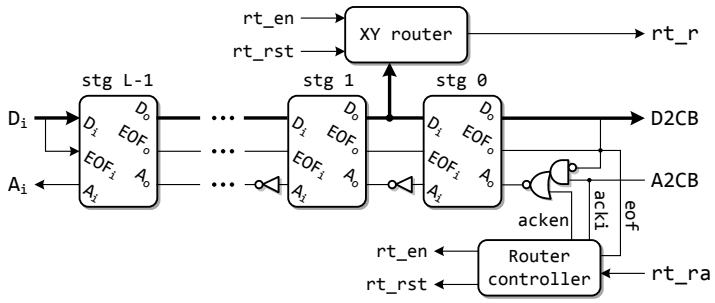


Figure 5.4: Input buffer for a virtual circuit

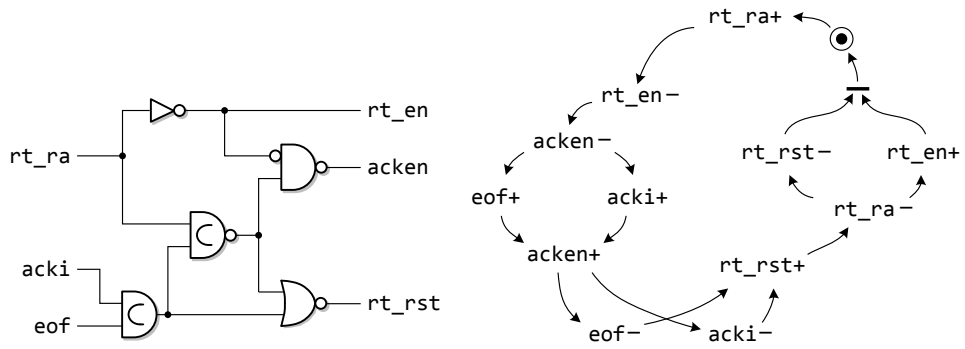
5.3 An SDM router

This section describes the hardware details of a relaxed-QDI SDM router. An SDM router is synthesized into gate-level netlists using the Faraday 0.13 μm cell library [44]. The area and speed performance are revealed accordingly.

5.3.1 Router structure

The overall structure of an SDM router has already been presented in Figure 5.3. The buffer in each input port is divided into M independent input buffers for individual virtual circuits, each of which is W/M bits wide. Every one of them has its own XY router and router controller. The structure of an input buffer for a virtual circuit is shown in Figure 5.4 [115]. It has L stages of buffers. The 0th pipeline stage is connected to the central switch and controlled by the local router controller. Incoming header flits are blocked in the 1st pipeline stage and analysed by the local XY router, whose internal structure has been illustrated previously in Figure 4.11. The decoded route requests *rt_r* are sent to the switch allocator. A path inside the central switch is successfully reserved when a notification is received from *rt_ra*. The local router controller can temporarily pause the pipeline by setting the active low signal *acken* to high when a new header flit is expected.

Figure 5.5 shows the router controller circuit synthesized from its STG using Petrify [29]. Initially the buffer is blocked waiting for a new header flit. Once a path is reserved in the central switch, indicated by *rt_ra*+, the XY router is disabled (*rt_en*−) and the buffer is activated (*acken*−). After the frame is delivered, the tail flit drives both *eof* and *acki* to high. Triggered by the tail flit, the router controller sequentially blocks the buffer and withdraws the tail flit (*acken*+), resets the decoded routing request (*rt_rst*+), and finally restarts the XY router (*rt_en*+) for the succeeding frame.



(a) Implementation (b) STG

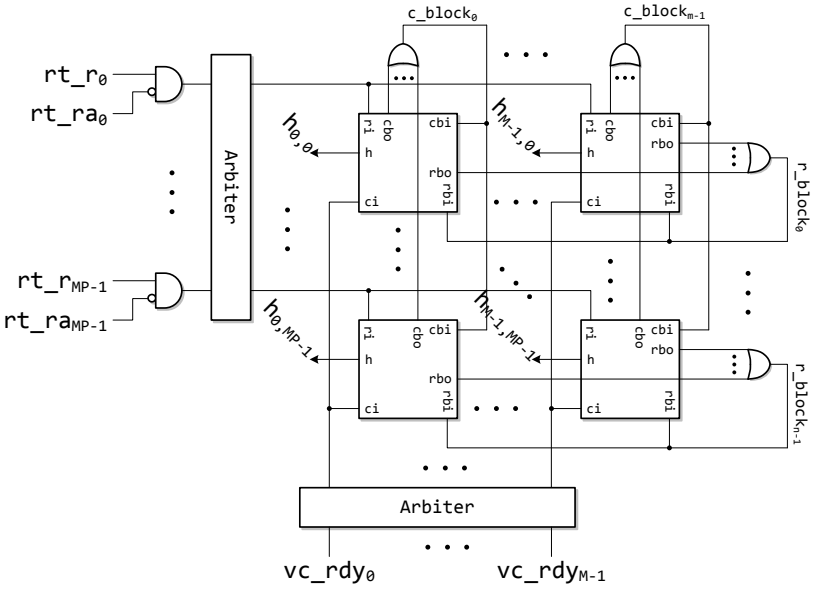
Figure 5.5: Router controller

The output buffer of each virtual circuit contains only one pipeline stage, which decouples the critical cycle from the inter-router channels. The central switch is an $MP \times MP$ crossbar dynamically configured by P switch allocators, one for each output port. Since the router controllers in input buffers guarantee that the decoded routing requests remain stable until the whole frame is delivered, no extra C-elements are needed in the crossbar.

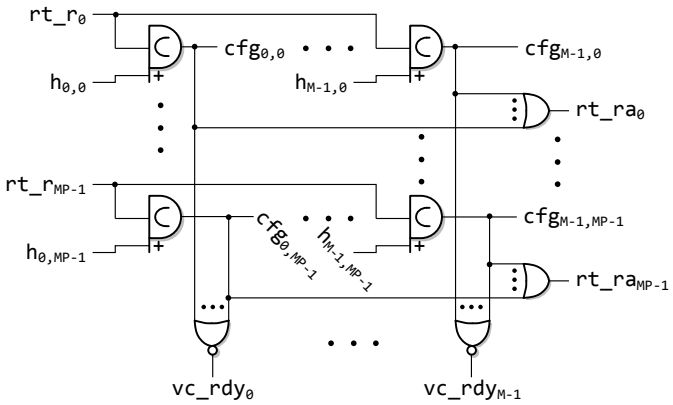
The switch allocator is the most complicated component in an SDM router. Unlike a wormhole router where every output buffer heads to a different direction, an SDM router has up to M virtual circuits implemented in every output port heading to the same direction. The central switch allocator can be separated into P distributed allocators but every distributed allocator is required to allocate M output virtual circuits to MP input virtual circuits, which cannot be done by asynchronous arbiters. So far the only QDI allocator available for this task is the multi-resource arbiter[4] (Section 2.7).

Figure 5.6 demonstrates a switch allocator for one output direction. It has two parts: a multi-resource arbiter in the lower layer (Figure 5.6a) and a configuration capture matrix in the upper layer (Figure 5.6b). The multi-resource arbiter reads requests from *rt_r* and matches them to available virtual circuits indicated by *vc_rdy*. As described in Section 2.7.2, the match process proceeds in a sequential way: only one pair of rt_r_i and vc_rdy_j is matched at one time. The match result is denoted by $h_{j,i}$. In order to match another pair, the corresponding rt_r_i and vc_rdy_j are withdrawn immediately after the result $h_{j,i}$ is captured in the configuration capture matrix.

[4]The multi-resource arbiter is named an arbiter but actually it is an allocator due to its ability to match multiple resources to multiple clients. For the definitions of arbiter and allocator, please refer to Chapter 19 of [35].



(a) Multi-resource arbiter



(b) Configuration capture matrix

Figure 5.6: Switch allocator

The configuration capture matrix is depicted in Figure 5.6b. The switch configuration signals *cfg* are generated by the matrix of C-elements, which are triggered by *h*. When a configuration $cfg_{j,i}$ is produced, the corresponding virtual circuit ready flag vc_rdy_j is withdrawn and the acknowledgement to the input virtual circuit rt_ra_i is set. The acknowledgement is also connected to the multi-resource arbiter in order to release $h_{j,i}$.

5.3.2 Performance

An QDI SDM router [115] has been implemented using the Faraday standard cell library [44] based on the UMC $0.13\ \mu\mathrm{m}$ technology. The router has five bi-directional ports compatible with normal mesh topology. The data width of each port is set to 32 bits. Four virtual circuits are implemented in each port. Every input buffer has two pipeline stages.

In the flit definition for wormhole routers as described in Table 4.1, a header flit contains an 8-bit address field and three bytes of data. In the SDM router, every port and buffer are divided into four virtual circuits, each of which is eight bits wide. Thus every original 32-bit flit in the network using wormhole routers are serialized into four 8-bit flits in the network using SDM routers. For the header flit, the first serialized flit contains the 8-bit address to avoid extra routing latency. The router can provide more than four virtual circuits although extra area and latency overhead is introduced. Assuming N bits are required to decode a route request, using a virtual circuit which is less than N bits wide divides the N bits into multiple flits. In this case, the XY router has to wait for multiple flits before producing a routing request. Undesirable serialization latency is introduced.

The router area derived from the post-synthesis netlist is $71,956\ \mu\mathrm{m}^2$ (equivalent to $17.99\ \mathrm{k}$ gates). Accurate gate delays are obtained from the coarse placer provided by the Synopsys Designer Compiler Topographical Technology [123] in synthesis. The evaluated period of the SDM router is $4.1\ \mathrm{ns}$. The router latency for data flits is $2.49\ \mathrm{ns}$. The XY router and the switch allocator require $0.51\ \mathrm{ns}$ and $3.21\ \mathrm{ns}$ respectively to decode a route request and reserve a path.

5.4 Behavioural level comparison

This section provides behavioural level SystemC [92] models for wormhole, SDM and VC routers; therefore, they can be simulated in large networks with various configurations. Area and latency estimation models are derived from different router architectures. The latency estimation is used in SystemC simulations for accurate throughput performance [115].

5.4.1 Models for wormhole and SDM routers

Area consumption

The area of a router with P ports can be expressed as:

$$A = P \cdot (A_{\mathrm{IB}} + A_{\mathrm{OB}}) + A_{\mathrm{CB}} + A_{\mathrm{A}} \tag{5.5}$$

where A_{IB}, A_{OB}, A_{CB} and A_{A} are the area of an input buffer, an output buffer, the crossbar and all allocators.

As shown in Figure 5.4, an input buffer contains L buffer stages, an XY router and a router controller. A buffer stage is built from multiple 4-phase 1-of-4 pipelines. Supposing the data width is W, every buffer stage needs $2W + 1$ C-elements to store data and the *eof* bit. The completion detection circuit requires extra $0.5W - 1$ C-elements to generate the common ack signal. Accordingly, the area of an input buffer in a wormhole router can be expressed as:

$$A_{\mathrm{IB,WH}} = L \cdot (2.5W A_{\mathrm{C}} + A_{\mathrm{EOF}}) + A_{\mathrm{R}} + A_{\mathrm{RC}} \tag{5.6}$$

where A_{C}, A_{EOF}, A_{R} and A_{RC} represent the area of a 2-input C-element, the extra logic introduced by the *eof* bit, the XY router and the router controller respectively.

An output buffer is a single buffer stage.

$$A_{\mathrm{OB,WH}} = 2.5W A_{\mathrm{C}} + A_{\mathrm{EOF}} \tag{5.7}$$

M virtual circuits are implemented in an SDM router, each of which is W/M bits wide. As a virtual circuit delivers frames independently, it is a fully functional input buffer but with a narrow data width. The area of the input and output buffers of SDM routers can be calculated as follows:

$$A_{\mathrm{IB,SDM}} = M \cdot [L \cdot (\frac{2.5W}{M} \cdot A_{\mathrm{C}} + A_{\mathrm{EOF}}) + A_{\mathrm{R}} + A_{\mathrm{RC}}] \tag{5.8}$$

$$A_{\mathrm{OB,SDM}} = 2.5W A_{\mathrm{C}} + M A_{\mathrm{EOF}} \tag{5.9}$$

As shown in Figure 5.7, a crossbar consists of an AND gate matrix and several OR gate trees. The size of the AND gate matrix and the number of the OR gate trees are determined by the number of ports and the wire count of each port. The crossbar inside a wormhole router has P ports while the one in an SDM router has MP ports.

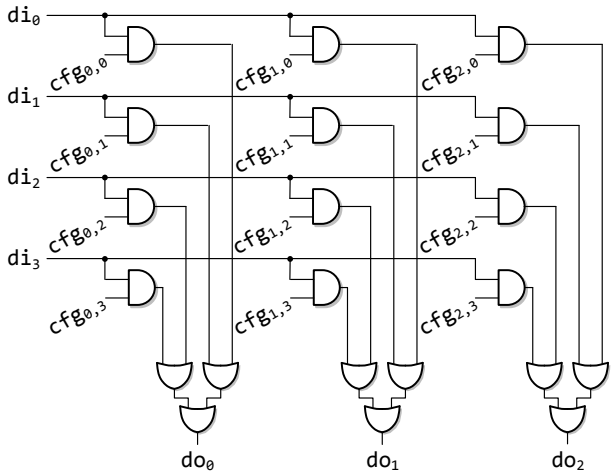


Figure 5.7: A 1-bit 4×3 crossbar

Assuming all 2-input gates have the same area, the area of crossbars is as follows:

$$A_{\mathrm{CB,WH}} = (2W + 2)(2P^2 - P)A_{\mathrm{g}} \tag{5.10}$$

$$A_{\mathrm{CB,SDM}} = (\frac{2W}{M} + 2)(2M^2P^2 - MP)A_{\mathrm{g}} \tag{5.11}$$

where A_{g} is the equivalent area of a 2-input gate.

It is difficult to provide accurate area estimation for allocators. The area depends on the structure of the allocator and the number of clients/resources. Assuming all routers use the multi-resource arbiter, whose area is roughly proportional to the arbitration scheme, the area can be estimated as:

$$A_{\mathrm{A,WH}} = P^2 A_{\mathrm{arb}} \tag{5.12}$$

$$A_{\mathrm{A,SDM}} = M^2 P^2 A_{\mathrm{arb}} \tag{5.13}$$

where A_{arb} is the equivalent area overhead of a single arbitration point for one client and one resource.

Several wormhole and SDM routers have been implemented. Using the area reported from the post-synthesis netlists, the parameters in Equation 5.6 to 5.13 are extracted as follows (in unit of $\mu\mathrm{m}^2$):

$$A_{\mathrm{C}} = 14.7 \quad A_{\mathrm{EOF}} = 11 \quad A_{\mathrm{R}} = 440 \quad A_{\mathrm{RC}} = 45 \quad A_{\mathrm{g}} = 2.45 \quad A_{\mathrm{arb}} = 86$$

The detailed area and the estimation error of the area model are illustrated in Table 5.2. A wormhole router and an SDM router has been designed and synthesized. The crossbar and the switch allocator are simplified by removing unnecessary turn

Table 5.2: Area consumption

	Wormhole			SDM		
	Actual	Estimated	err(%)	Actual	Estimated	err(%)
Input buffers	14,303	14,295	0.0	21,995	21,900	-0.4
Output buffers	5,935	5,935	0.0	6,000	6,100	1.7
Crossbar	4,356	4,366	0.0	21,744	21,697	-0.2
Switch allocator	772	1,376	78.2	22,208	22,016	-0.9
Total	25,366	25,972	2.4	71,956	71,713	-0.3

(unit: μm^2)

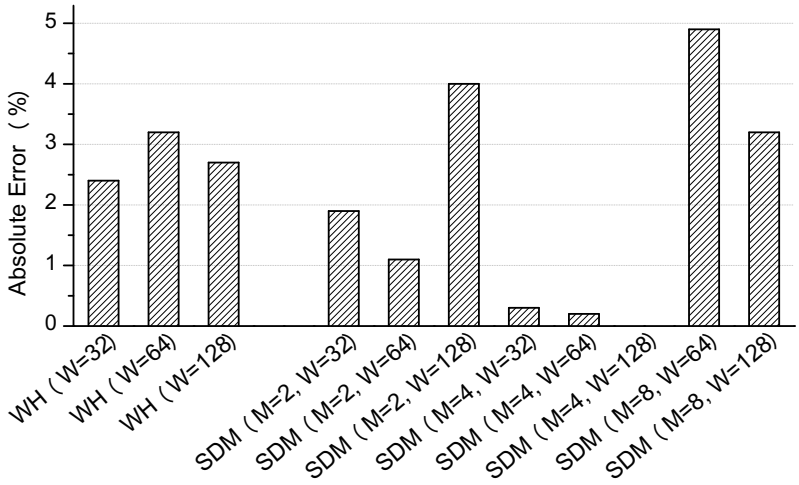


Figure 5.8: Area estimation error

models in the XY routing algorithm (see Section 7.1.2 and Figure 7.5 for a detailed explanation of the turn models); the actual area models have been adjusted accordingly. It is shown that the area models successfully estimate the area of all router components within a maximum error of 1.7% except for the switch allocator in the wormhole router. This significant error is due to the different allocator structure. The multi-way MUTEX arbiters used in wormhole routers consume much less area than multi-resource arbiters. Since allocators take only a small portion of the total area and the overall area of a router is more important than the area breakdown, this estimation error can be tolerated.

Figure 5.8 reveals the area estimation error of routers with various configurations. In all configurations, the estimation error is lower than 4.9% (SDM router $M = 4$ and $W = 64$). It is shown that the area models are adequate to estimate the area of routers in various configurations.

Latency analysis

The throughput of an asynchronous pipeline is constrained by the period of its critical cycle. Hence a latency model for the critical cycle is essential to achieve accurate throughput estimation. As described in Section 4.3, the critical cycle in asynchronous routers is the path around the pipeline stages connected to the central switch. Assuming rising time and falling time are the same in all gates, the latency T of the critical cycle can be expressed as:

$$T = 4t_{\mathrm{C}} + 4t_{\mathrm{CB}} + 2t_{\mathrm{CD}} + 2t_{\mathrm{AD}} + t_{\mathrm{CTL}} \tag{5.14}$$

where t_{C} is the propagation latency of a C-element used to store data, t_{CB} is the propagation latency of the crossbar, t_{CD} is the propagation latency of the completion detection circuit, t_{AD} is the latency of the isochronic fork on the receiver end of the ack line, namely the ack driver latency, and t_{CTL} is the extra latency caused by the router controller.

Linear delay models are used to approximate these latencies. The C-elements are connected to the crossbar. Thus t_{C} is linear with the number of ports of the crossbar.

$$t_{\mathrm{C,WH}} = l_{\mathrm{C}} + k_{\mathrm{C}}(P+1) \tag{5.15}$$

$$t_{\mathrm{C,SDM}} = l_{\mathrm{C}} + k_{\mathrm{C}}(MP+1) \tag{5.16}$$

where l_{c} is the latency of a C-element with zero load and k_{c} is the fanout latency (the extra latency introduced by every extra fanout). Similarly, the ack driver latency t_{AD} is linear with the wire count of one buffer stage.

$$t_{\mathrm{AD,WH}} = l_{\mathrm{AD}} + k_{\mathrm{AD}}(2W+1) \tag{5.17}$$

$$t_{\mathrm{AD,SDM}} = l_{\mathrm{AD}} + k_{\mathrm{AD}}(2W/M+1) \tag{5.18}$$

where l_{AD} and k_{AD} are the latency of the ack driving gate and the fanout latency.

The crossbar contains an AND gate matrix and several OR gate trees. The depth of the OR gate tree is determined by the number of input ports. Assuming all OR gates have the same propagation latency,

$$t_{\mathrm{CB,WH}} = l_{\mathrm{CB}} + k_{\mathrm{CB}} \cdot log_2(P) \tag{5.19}$$

$$t_{\mathrm{CB,SDM}} = l_{\mathrm{CB}} + k_{\mathrm{CB}} \cdot log_2(MP) \tag{5.20}$$

where l_{CB} and k_{CB} are the propagation latencies of an AND gate and an OR gate respectively.

The completion detection circuit contains a C-element tree. The latency of this tree is proportional to its depth. The final common ack signal drives the gates to all input ports; therefore, its fanout is proportional to the number of ports. The latency estimation must consider the impact by both effects.

$$t_{\mathrm{CD,WH}} = l_{\mathrm{CD}} + l_{\mathrm{C}} \cdot log_2(W/2) + k_{\mathrm{CD}} \cdot P \tag{5.21}$$

$$t_{\mathrm{CD,SDM}} = l_{\mathrm{CD}} + l_{\mathrm{C}} \cdot log_2(\frac{W}{2M}) + k_{\mathrm{CD}} \cdot MP \tag{5.22}$$

where l_{CD} and k_{CD} are the zero load latency of a completion detection circuit without any C-elements (the latency of two OR gates) and the fanout latency.

The router controller in wormhole or SDM routers halts the input buffer only once per frame to analyse incoming header flits. Once a path allocation is made for a frame, pipelines run at full speed and no extra control latency is introduced ($t_{\mathrm{CTL}} = 0$). However, the controller in a VC router halts the data path in every cycle to reconfigure the crossbar, which leads to non-zero t_{CTL}.

All parameters are extracted from back-annotated post-synthesis simulations. Router implementations have removed all unnecessary turns in the crossbar according to the XY routing algorithm (see Section 7.1.2 and Figure 7.5 for a detailed explanation of the turn models). Wire latency is counted as part of gate latency automatically using the Synopsys Designer Compiler Topographical Technology [123] in synthesis. The extracted parameters are listed as follows (in unit of ns):

$$\begin{array}{llll} l_{\mathrm{C}} = 0.15 & k_{\mathrm{C}} = 0.01 & l_{\mathrm{CB}} = 0.074 & k_{\mathrm{CB}} = 0.044 \\ l_{\mathrm{CD}} = 0.23 & k_{\mathrm{CD}} = 0.004 & l_{\mathrm{AD}} = 0.17 & k_{\mathrm{AD}} = 0.005 \end{array}$$

The practical speed performance and the estimation errors of the latency models are shown in Table 5.3. Compared with the errors of the area models, the latency estimation causes larger errors for several reasons: the practical latency of a gate is not linear; the load is not exactly proportional to fanout; the propagation time of a gate is related to the input transition time which is difficult to estimate statically.

Figure 5.9 shows the latency estimation errors of routers with various configurations. Most errors are smaller than 6.5%. The significant errors occurs when large data width ($W/M > 32$) is used. These wide pipelines cause design rule violations (max capacitance and max transition time) on ack lines. Buffer trees are inserted to

Table 5.3: Router latency

	Wormhole			SDM		
	Actual	Estimated	err(%)	Actual	Estimated	err(%)
Period	4.22	4.130	-2.1	4.10	3.978	-3.0
Router latency	2.29			2.49		
XY router decoding	0.44			0.51		
Switch allocation	0.78			3.21		
t_{C}	0.22	0.200	-9.1	0.34	0.320	-5.9
t_{CB}	0.16	0.162	1.3	0.26	0.250	-3.8
t_{CD}	0.79	0.846	7.6	0.57	0.594	4.2
t_{AD}	0.53	0.495	-6.6	0.27	0.255	-5.5
t_{CTL}	0.00			0.00		

(unit: ns)

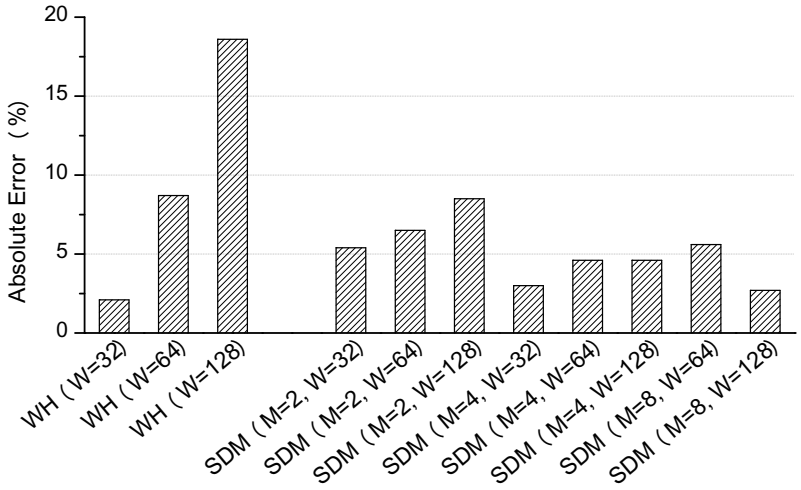


Figure 5.9: Latency estimation error

eliminate these violations in synthesis. Significant errors are introduced as these buffer trees do not fit in the linear latency model. To cope with this problem, all simulations using data width larger than 32 bits in Section 5.4.3 are carefully compensated[5]. It is important to notice that technology and cell libraries have strong impacts on the value of latency parameters. They must be extracted again if a different cell library is used.

[5]When wide pipelines are used ($W/M > 32$), the compensated ack driver latency t'_{AD} is used in simulations. $t'_{\mathrm{AD}} = t_{\mathrm{AD}} + e_{\mathrm{AD}}$, where e_{AD} is the correction factor of t_{AD}. Using the estimation errors in wormhole routers, e_{AD} is extracted as -0.2 ns, -0.74 ns and -1.82 ns for the data widths of 64-bit, 128-bit and 256-bit respectively.

5.4.2 Model for VC routers

A conceptual area and latency estimation model will be provided for VC routers. Assuming the asynchronous VC router uses the internal structure of an ANOC router [7] (Figure 3.13), it comprises P input buffers, an extended $MP \times P$ crossbar, P output buffers, a VC allocator and a switch allocator. It uses the input buffering scheme and each input buffer contains M VCs.

Using the same assumptions in Section 5.4.1, the area model of a VC router is expressed as follows:

$$A_{\mathrm{IB,VC}} = M \cdot A_{\mathrm{IB,WH}} \tag{5.23}$$

$$A_{\mathrm{OB,VC}} = A_{\mathrm{OB,WH}} \tag{5.24}$$

$$A_{\mathrm{CB,VC}} = (2MP^2 - P) \cdot (2W + 2) \cdot A_{\mathrm{g}} \tag{5.25}$$

$$A_{\mathrm{A,VC}} = (M^2P^2 + MP) \cdot A_{\mathrm{arb}} \tag{5.26}$$

The area in Equation 5.26 includes two parts: the VC allocator and the switch allocator. As described in [96], the arbitration scheme of a fair VC allocator is $MP \times MP$ because MP output VCs are dynamically allocated to MP input VCs. The arbitration scheme of the switch allocator is $M \times P$ because an output port is requested by a maximum of M input VCs simultaneously.

A VC router using the input buffer scheme has the same critical cycle traversing the crossbar as wormhole and SDM routers. The latency of the critical cycle can be approximated as follows:

$$t_{\mathrm{C,VC}} = t_{\mathrm{C,WH}} \tag{5.27}$$

$$t_{\mathrm{CD,VC}} = l_{\mathrm{CD}} + l_{\mathrm{C}} \cdot log_2(W/2) + k_{\mathrm{CD}} \cdot MP \tag{5.28}$$

$$t_{\mathrm{AD,VC}} = t_{\mathrm{AD,WH}} \tag{5.29}$$

$$t_{\mathrm{CB,VC}} = t_{\mathrm{CB,WH}} \tag{5.30}$$

Using the parameters provided in Section 5.4.1, the estimated delays for a 32-bit 5-port asynchronous VC router with four VCs are listed as follows (in unit of ns):

$t_{\mathrm{C}} = 0.20$	$t_{\mathrm{CB}} = 0.16$	$t_{\mathrm{CD}} = 0.89$	$t_{\mathrm{AD}} = 0.50$	$t_{\mathrm{CTL}} = 0.78$
period = 5.01		XY router decoding = 0.44		
VC allocator = 3.21		switch allocator = 0.78		

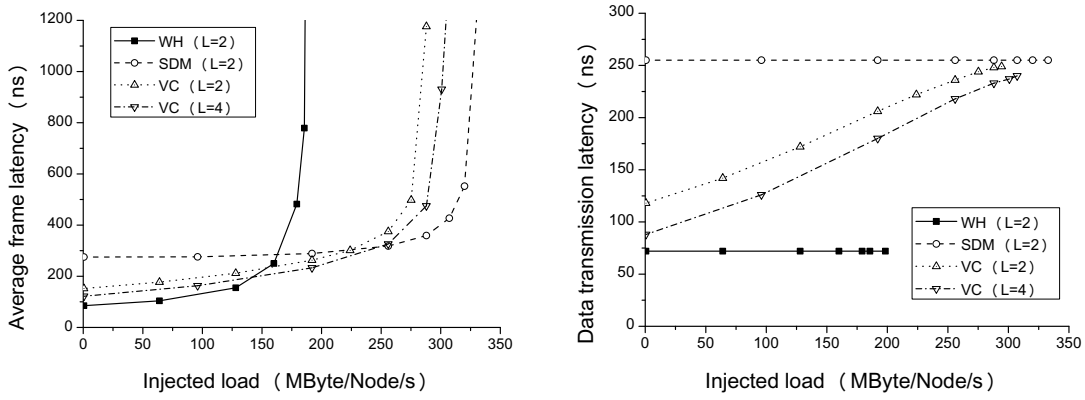
Asynchronous VC routers have the longest period in all router architectures. Both VC and wormhole router suffer from the longest ack driver latency and the deepest C-element tree in the completion detection circuit. The crossbar in a VC router introduces extra latency as an SDM router does. The fact that the crossbar is reconfigured in every cycle introduces extra control latency. The value of this extra control latency t_{CTL} comes from the switch allocation latency of the wormhole router assuming multi-way MUTEX arbiters are used in switch allocators. As a VC allocator has the same arbitration scheme as the switch allocator in an SDM router, their latencies are assumed the same.

5.4.3 Performance analyses

An 8×8 mesh network has been built with latency accurate SystemC models. Since the best-effort performance is the major research target of this thesis, all simulations use the random uniform traffic pattern. Every network node sends frames uniformly to other nodes in a Poisson sequence.

Figure 5.10 [115] shows the average frame transmission latency with various injection rates. All routers are equipped with two stages of input buffers except for the VC router which is also simulated with four stages. The payload size of a frame is set to 64 bytes. Four virtual circuits/VCs are implemented. It is shown that both VC and SDM improve throughput but SDM outperforms VC. The SDM router provides the best saturation throughput of $346\ \mathrm{MByte/Node/s}$, which is around 1.7 times that of the wormhole router. However, the minimum frame latency in an idle network is prolonged because the data width of a virtual circuit is only a portion of the total data width. Frames are delivered in a serialized way. The minimum frame latency of SDM routers is $275\ \mathrm{ns}$, which is 3.2 times that of using wormhole routers and 1.8 times that of using VC routers.

VC routers suffer from the credit loop latency when their input VC buffers are short. Every asynchronous buffer stage is a half buffer stage. Thus a full buffer stage needs two half buffer stages. Most VC routers use the credit based backpressure method [45, 12, 7, 38] as shown in Figure 5.11. The switch allocator in one output port allocates the output buffer to input VCs only when the corresponding input VC buffer in the next router has available buffer space. The credit buffers in one output port record the available buffer space of all input VC buffers in the next router using tokens. A token is consumed when a flit is transmitted to the next router and a token is return when the input VC buffer in the next router transmits a flit. When input VC



(a) Average frame latency (b) Data transmission latency

Figure 5.10: Latency under various network loads ($P = 5$, $W = 32$, $M = 4$)

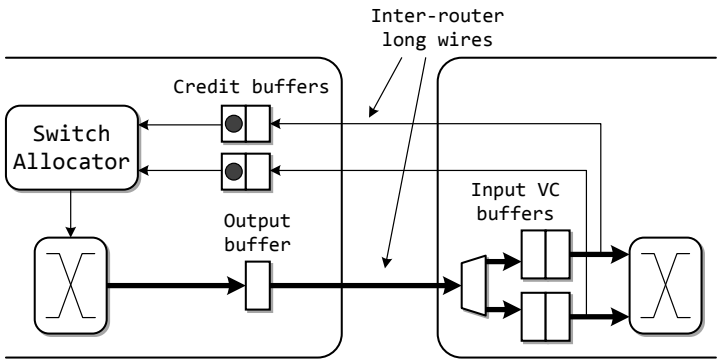


Figure 5.11: Credit based backpressure method

buffers are short, such as only one full buffer stage, the credit buffer of every VC has only one token.

Under low network load, it is likely that only one VC buffer in all VC buffers is utilized. In this case, a new flit has to wait until the token consumed by the previous flit is returned to the corresponding credit buffer. The latency of the loop path is called the credit loop latency. Inferred from the synchronous credit loop calculation (Equation 16.1 in [35]), the credit loop latency t_{crt} in an asynchronous VC router can be expressed as[6]:

$$t_{crt} = Lt_C + 2t_w + t_{CTL} + T \tag{5.31}$$

where t_w is the latency of the inter-router long wires. Adopting the long wire latency in [105], the credit loop latency of the VC router is around 1.3 periods. As a result,

[6]Equation 5.31 is provided as an explanation to the credit loop latency. It is not intended for accurate latency estimation because different switch allocator and input buffer implementations significantly affect the latency calculation.

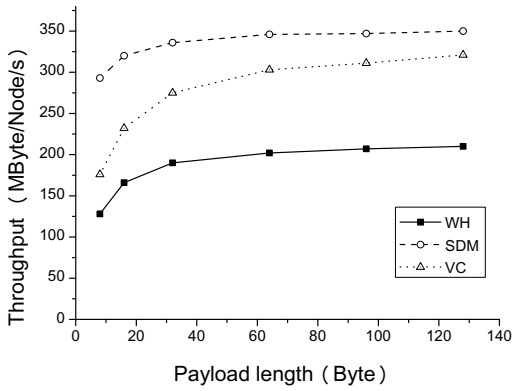


Figure 5.12: Throughput with various payload lengths
($P = 5$, $W = 32$, $L = 2$, $M = 4$)

at least two full buffer stages are required to avoid the credit stall under low network load. Figure 5.10a shows that adding two more half buffers (two stages of full buffers in total) reduces the frame latency to 122 ns; however, it does not raise throughput significantly and introduces extra area overhead by doubling the buffer size. When the network is heavily loaded, multiple VCs are utilized simultaneously.

When a flit of one VC is waiting for a new token, the flits of other VCs can utilize the output buffer as long as they have tokens available in the credit buffers. Thus the credit stall is unlikely to compromise the saturation throughput.

Frame latency comprises two parts: the time consumed to reserve a path to the target PE and the time consumed to transmit data. Figure 5.10b shows the data transmission latency. In wormhole and SDM routers, links and buffers are exclusively allocated to frames. Resources are not shared during data transmission and the data transmission latency is not affected by network load. On the contrary, the switches in VC routers are shared by all VCs in a time divided manner. The data transmission latency increases when the network is heavily loaded. This result reveals the potential of using SDM to reduce the latency jitter when hard latency guaranteed services are required.

Figure 5.12 demonstrates how the payload size affects the saturation throughput. Every frame has a fixed amount of control overhead, such as the target address in the header flit. Short frames are not efficient as the payload transmitted in each frame is small. The saturation throughput rises with payload length. This increase is not linear. The saturation throughput approaches its maximum when payload is larger than 64 bytes. Therefore, the payload size is fixed to 64 bytes in all following simulations. SDM outperforms VC and provides the highest throughput of 350 $\mathrm{MByte/Node/s}$ in the 128 Byte case.

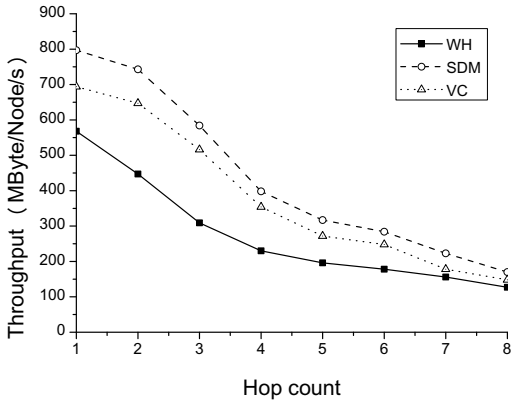


Figure 5.13: Throughput with various communication distances ($P = 5$, $W = 32$, $L = 2$, $M = 4$)

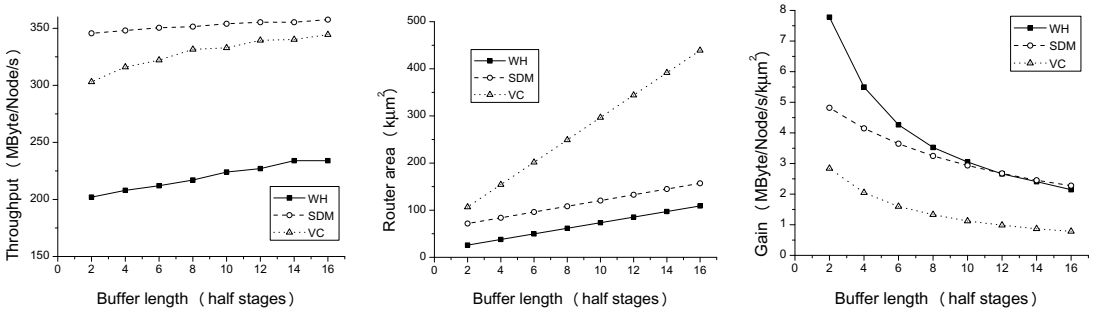
Both VC and SDM alleviate the head-of-line (HOL) problem. It is known that long distance communications are more vulnerable to contention than local communications. Figure 5.13 [115] demonstrates the throughput variation with different communication distances. In this simulation, network nodes send frames uniformly to all nodes certain hops away. The saturation throughput drops significantly with the increasing communication distance. Both VC and SDM achieve better performance increment in local traffic patterns than in long distance communication patterns. When the communication distance is eight hops, using VC shows little throughput boost but SDM still raises the throughput by 33.9%.

Buffer size is an important design parameter. Increasing buffer length boosts throughput as more frames can be stored in the network when contention occurs. However, buffers consume significant area. Figure 5.14 [115] reveals the throughput improvement by increasing the buffer length. Both the saturation throughput and the area overhead rise linearly. Figure 5.14c shows the gain of throughput per area unit. Gain is defined as:

$$\text{Gain} = \frac{\text{throughput}}{\text{router area}} \tag{5.32}$$

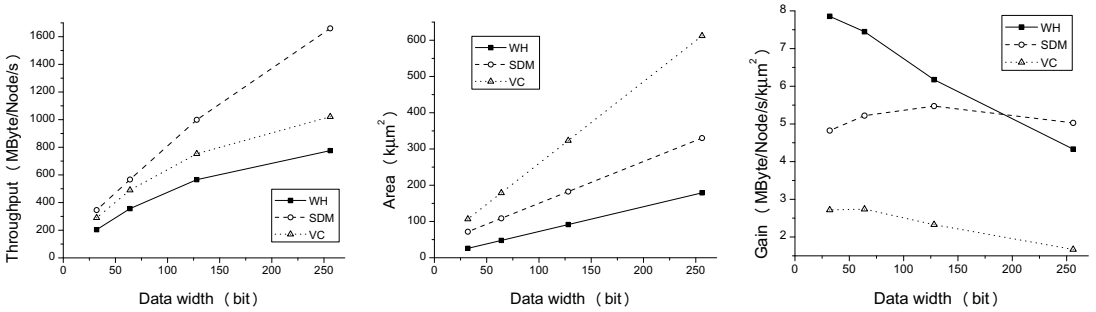
Higher gain indicates better area-to-throughput efficiency. The gain of all router architectures drops along with buffer length. It is not efficient to improve throughput by adding buffers. The basic wormhole router shows the best area efficiency with short buffers. When long buffers are implemented ($L \geq 12$), SDM routers achieve the best area efficiency. In all cases, VC routers suffer from the worst area efficiency.

In addition to adding buffers, increasing data width also raises throughput, but the



(a) Throughput (b) Area (c) Gain

Figure 5.14: Performance with various buffer lengths ($P = 5$, $W = 32$, $M = 4$)

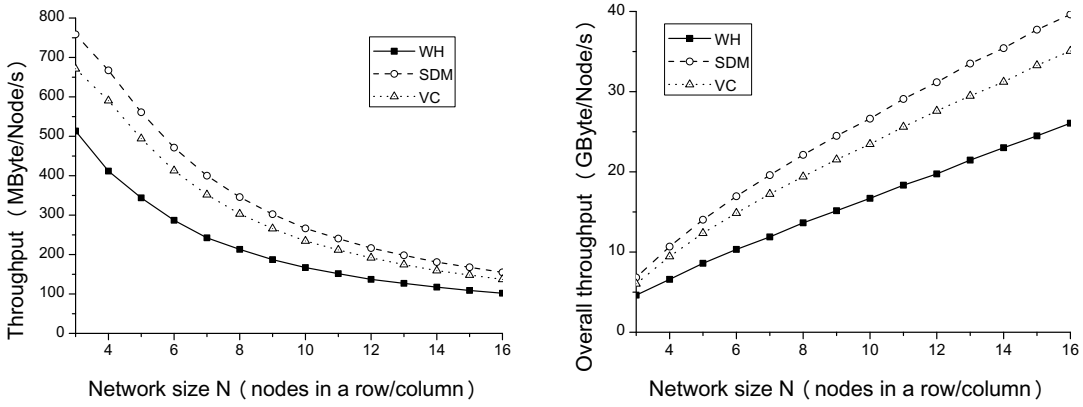


(a) Throughput (b) Area (c) Gain

Figure 5.15: Performance with various port data width ($P = 5$, $L = 2$, $M = 4$)

increase is not linear. Transmitting frames in wide pipelines suffers from the increased control overhead because the number of flits per frame is decreased. The delay incurred by reconfiguring the central switch in a router is independent of data width. In an overloaded network, decreasing the number of flits per frame leads to frequent switch reconfiguration, which in turn compromises throughput. Furthermore, wide pipelines introduce extra synchronization overhead. Figure 5.15 [115] reveals the impact of increasing data width. Router area increases linearly with data width. Compared with wormhole and VC routers, SDM routers show a steadier throughput increment because their C-element trees in the completion detection circuits are shorter. Figure 5.15c shows the gain of throughput per area unit. Although wormhole routers still show the best gain with narrow pipelines, the gain of SDM routers increases with data width until 128 bits. When the data width is over 200 bits, SDM demonstrates the best area efficiency.

Since both VCs and virtual circuits use the wormhole flow control method, VC



(a) Node throughput (b) Overall throughput

Figure 5.16: Network scalability ($P = 5$, $W = 32$, $L = 2$, $M = 4$)

routers and SDM routers have similar scalability to network size with wormhole routers. As provided in Section 3.3.1 in [35], Equation 5.33 gives the upper bound on throughput Θ in random uniform traffic.

$$\Theta \leq \Theta_{\text{ideal}} \leq \frac{2WB_C}{TN^2} \tag{5.33}$$

where B_C is the minimal channel count over all bisections of the network and N^2 is the total number of nodes in a mesh network. Since B_C is linear with N in a mesh network, the upper bound on throughput Θ and the overall throughput of all nodes in the network ΘN^2 can be simplified into:

$$\Theta \leq \frac{k}{TN} + C \tag{5.34}$$

$$\Theta N^2 \leq \frac{kN}{T} + C' \tag{5.35}$$

where k, C and C' are constants. Figure 5.16 [115] reveals the practical throughput and overall throughput of networks with different sizes. Figure 5.16a verifies the inverse relation between Θ and N, and the overall throughput in Figure 5.16b is approximately linear with N. Note that Equation 5.35 provides only the upper bound on overall throughput. Although VC routers have larger period than wormhole routers, they show better overall throughput thanks to their capability of resolving HOL problems. All flow control methods present similar scalability to network size.

Increasing the number of virtual circuits or VCs allows more frames to be delivered concurrently and raises throughput. Figure 5.17 [115] demonstrates the frame latency

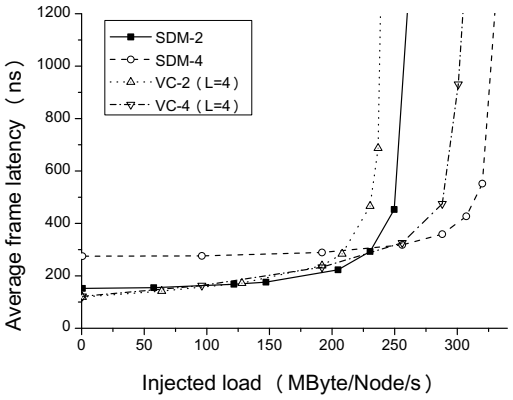


Figure 5.17: Throughput with various number of virtual circuits or VCs ($P = 5$, $W = 32$, $L = 2$)

versus network load for SDM and VC routers with various numbers of virtual circuits and VCs. The estimated router area is listed as follows (in unit of μm^2):

SDM(M=2)	38,153	SDM(M=4)	71,713
VC(M=2)	49,896	VC(M=4)	103,250

Both VC routers and SDM routers benefit significantly from the increased number of VCs/virtual circuits. Their throughput increase by 26.6% and 26.8% respectively. It is also shown that the minimum frame latencies of SDM routers increases with the number of virtual circuits. Since the total data width of a port is fixed, increasing the number of virtual circuits reduces the data width of a single virtual circuit. Data are serialized to fit the small data width leading to increased frame latency.

5.5 Summary

'Virtual channel' is the most utilized flow control method in asynchronous on-chip networks. It can be used to improve network throughput or provide QoS support. However, VC buffers introduce significant area overhead and the frequent switch allocation compromises throughput. Instead of sharing resources using timing division methods such as VC, spatial division multiplexing (SDM) alleviates the HOL problem without compromising throughput. The area overhead of SDM is smaller than VC.

Area and latency models have been provided to analyse wormhole, SDM and VC routers with various configurations. The parameters in these models are extracted from several post-synthesis netlists of wormhole and SDM routers using the Faraday 0.13 μm standard cell library. Latency accurate SystemC models are built to simulate

all router architectures in large networks at high speed. In all test cases, SDM routers outperform VC routers in throughput. Both increasing buffer depth and increasing port data width raise throughput but increasing port data width demonstrates better area to throughput efficiency than increasing buffer depth. When the port data width is larger than 200 bits, SDM routers show the best area efficiency in all router architectures. A design problem of SDM is the frame latency in low network load, which is prolonged due to the serialized data transmission in virtual circuits.

Chapter 6

Area Reduction using Clos Networks

The preceding chapter has proposed using spatial division multiplexing (SDM) rather than virtual channel (VC) in asynchronous on-chip networks for high throughput. It is shown that SDM achieves better throughput and consumes less area than VC. The major area overhead of an SDM router is the central switch, the size of which is proportional to the number of virtual circuits. This chapter provides a solution to this overhead using Clos switching networks instead of crossbars. The chapter will end with an introduction of a novel 2-stage Clos switch which consumes the smallest area — a perfect switch structure for asynchronous SDM routers.

6.1 Clos switching networks

A switching network is a switch architecture which dynamically connects input ports to output ports. Clos networks are a class of multi-stage switching networks [27] providing the theoretically optimal solution for high-radix switches.

Clos networks were first used in telephone networks where high-radix switches are statically configured. The later asynchronous transfer mode (ATM) networks and internet protocol (IP) networks achieve high throughput using packet switching technologies [21]. A Clos network designed for current optical backbone networks has already reached peta-bit throughput [19].

Clos networks have already been utilized in intra- and inter-chip interconnection networks. Transistor scaling increases the available bandwidth on-chip. It is found that a router with many narrow ports is more efficient than a router with a few wide

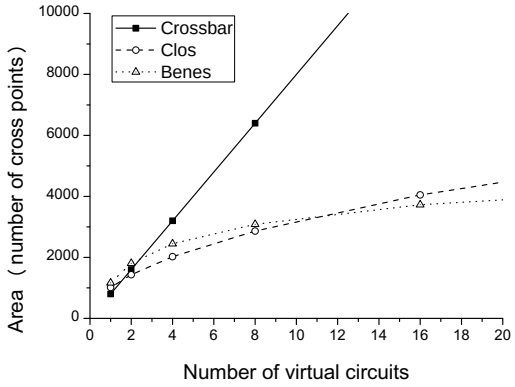


Figure 6.1: Area of different switches

ports [64, 55]. A folded-Clos network is used in the Cray BlackWidow multiprocessor to support high bandwidth communications [102] and Beneš networks (multi-stage Clos networks) [8] are used in a synchronous SDM on-chip network to provide delay-guaranteed services [71].

This chapter tries to reduce the area overhead of the crossbar in an SDM router using Clos networks. Similar work has been done in [71] where a synchronous SDM router uses a Beneš network as the central switch. Although a significant area reduction is reported, the multi-stage Beneš network cannot be reconfigured in run time. This chapter will demonstrate an asynchronous Clos scheduler which can dynamically reconfigure unbuffered 3-stage Clos networks.

Figure 6.1 illustrates the potential area reduction of using Clos networks. As the area of a switch is proportional to the number of cross-points inside the switch, the area overhead of using different switch structures in a 5-port 32-bit SDM router is evaluated in the number of cross-points. It is shown that the single stage crossbar structure is more area efficient than multi-stage switching networks when the switch radix is small. When more than three virtual circuits are implemented, the general 3-stage Clos networks and the multi-stage Beneš networks require smaller area than crossbars. In other words, crossbars are the optimal switch structure for wormhole and VC routers but not SDM routers due to the increasing switch radix. Multi-stage Clos networks have the potential for area reduction in SDM routers.

The internal structure of general 3-stage Clos networks is depicted in Figure 6.2. The terminologies describing the internal components of a Clos network are listed as follows:

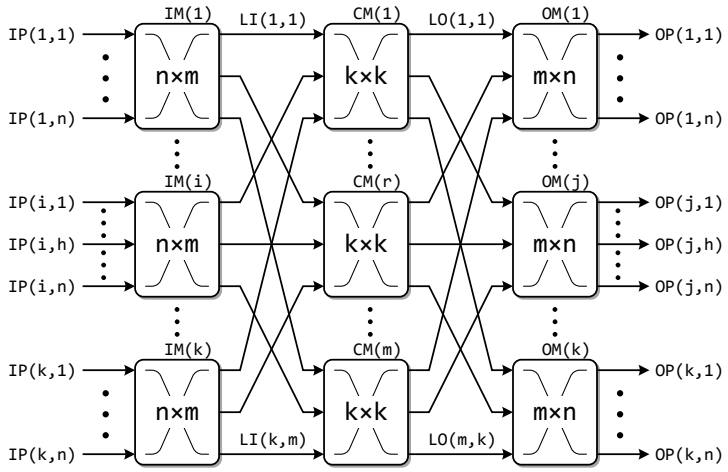


Figure 6.2: General 3-stage Clos network

IM	Input module at the first stage.
CM	Central module at the second stage.
OM	Output module at the third stage.
n	Number of input ports (IPs)/OPs in each IM/OM.
k	Number of IMs/OMs.
m	Number of CMs.
i	Index of IMs ($0 < i \leq k$).
j	Index of OMs ($0 < j \leq k$).
r	Index of CMs ($0 < r \leq m$).
h	Index of IPs/OPs in an IM/OM ($0 < h \leq n$).
IM(i)	The (i)th IM.
OM(j)	The (j)th OM.
CM(r)	The (r)th CM.
IP(i, h)	The (h)th IP in IM(i).
OP(j, h)	The (h)th OP in OM(j).
LI(i, r)	The link from IM(i) to CM(r).
LO(r, j)	The link from CM(r) to OM(j).
$\mathbf{C}(n, k, m)$	A Clos network has m CMs and k IMs/OMs with n IPs/OPs.
N	The total number of IPs/OPs ($N = nk$).

The first stage contains k IMs, each of which is an $n \times m$ crossbar. In the second stage, m CMs are statically connected to IMs and each CM is a $k \times k$ crossbar. The third stage contains k OMs, each of which is an $m \times n$ crossbar statically connected to CMs. When the size of a switching module is large, it can be replaced by an embedded

Clos network and, therefore, the overall Clos network has more than three stages (a well-known class of such Clos networks are Beneš networks [8]).

According to the connection capability, switching networks can be classified into three categories [21]: *Blocking*: the switches have possible connection states such that an available I/O pair cannot be connected because of internal blocking. *Strict non-blocking (SNB)*: the switches ensure the connection of any available I/O pairs without altering any established connections. *Rearrangeable non-blocking (RNB)*: the switches ensure the connection of any available I/O pairs with possible modification of established connections. A three-stage Clos network with n CMs ($m = n$) is a RNB network while it is an SNB network when the number of CMs is larger than $2n - 1$ [21].

Similar to crossbars, the area of a switching network is proportional to the number of cross-points. Both SNB and RNB Clos networks consume the minimum area when $k = \sqrt{2N}$.

$$A_{\text{Clos,SNB}} \geq [2(2N)^{1.5} - 4N] \cdot WA_{CP} \tag{6.1}$$

$$A_{\text{Clos,RNB}} \geq [(2N)^{1.5}] \cdot WA_{CP} \tag{6.2}$$

where W is the wire count of each port and A_{CP} is the equivalent area of a single cross-point. RNB Clos networks consume the minimum area overhead but introduce throughput degradation if established connections are not allowed to be reconfigured.

There are two classes of routing algorithms for Clos networks [20]: *optimal algorithms*, which provide guaranteed results for all matches but with a high complexity in time or implementation, and *heuristic algorithms*, which provide all or partial matches in low time complexity. Although optimal algorithms guarantee the connection of any I/O pairs, they require a global view of all modules and take a long time to reconfigure. They are normally used in statically configured Clos networks [71]. On the other hand, heuristic algorithms are fast and spatially distributed. Most dynamically reconfigurable Clos networks utilize heuristic algorithms [24, 90, 20, 19, 22, 101, 91].

Buffer insertion is a usual way of improving throughput in Clos networks. According to the stage where buffers are inserted, a Clos network can be a space-space-space network without buffers, a memory-space-memory network with buffer insertion in IMs and OMs, or a space-memory-space network with buffer insertion in CMs. Space-space-space networks introduce no buffer overhead but provide the worst throughput. Space-memory-space networks normally show better throughput than memory-space-memory networks because the buffers in CMs resolve the contention in CMs; however,

this scheme requires a re-sequencing function in OMs because data issued to OMs are out-of-order. Memory-space-memory is the most utilized scheme in ATM networks. Buffers in IMs and OMs improve throughput without the out-of-order problem but the OMs are required to speed-up m times to avoid throughput degradation (the detailed comparison of buffer insertion schemes and memory speed-up can be found in [21, 91]).

Only three-stage space-space-space Clos networks are researched in this thesis. The tight area budget of on-chip networks disallows any extra buffers except for the input/output buffers. Nevertheless, the routing algorithms for space-space-space Clos networks can be easily adopted in memory-space-memory or space-memory-space Clos networks.

6.2 Dispatching algorithm

Every CM in a Clos network is shared by all I/O pairs, since every I/O pair has m possible path configurations and each of them goes through a different CM. In the worst case, all the nk IPs would try to utilize the same CM ignoring that one CM is capable of setting up only k paths. Hence an efficient routing algorithm must dispatch requests evenly to all CMs otherwise throughput is compromised. Heuristic algorithms process a request from IP(i_1, h_1) to OP(j_2, h_2) in two stages [101]: *Module matching* first reserves a path from IP(i_1, h_1) to an LO(r, j_2) which is connected to OM(j_2). Then *port matching* connects LO(r, j_2) and OP(j_2, h_2) in OM(j_2). Since it is the module matching stage that chooses the target CM, it determines the request distribution which directly affects throughput. The sub-algorithm used in module matching, namely the dispatching algorithm, is the key research issue of this section.

6.2.1 Concurrent round-robin dispatching algorithm

The data transmitted in the synchronous Clos networks used in ATM and IP networks are routed in units of a cell — a small fraction of a packet with fixed size. Multiple cells are transmitted synchronously from IMs to OMs in one cell time. The reconfiguration of switches proceeds concurrently with data transmission in a pipelined manner. The new configuration generated in the current cell time takes effect in the next cell time. The latency of generating a new configuration for the Clos network is therefore hidden.

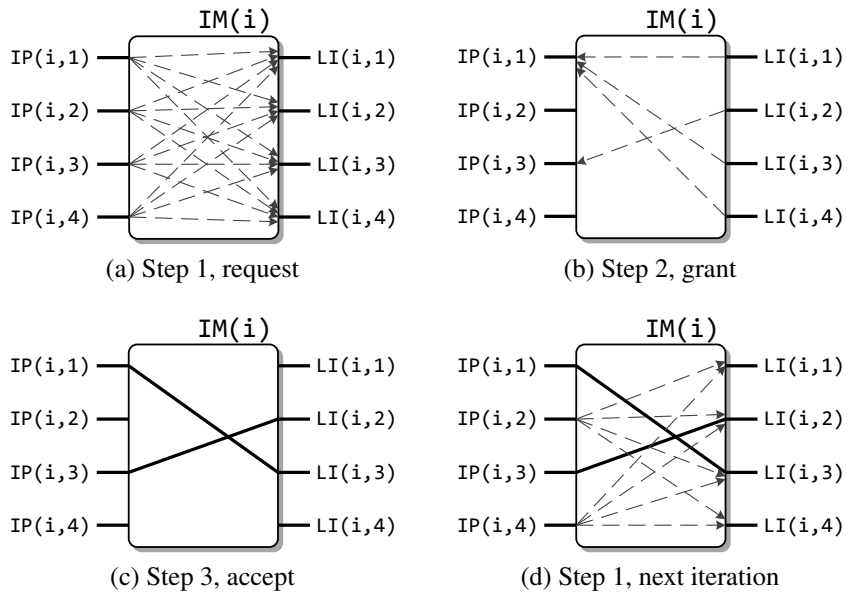
A cell time lasts one or multiple cycles depending on the complexity of the routing algorithm.

The concurrent round-robin dispatching (CRRD) algorithm [90] is one of the classic algorithms extensively utilized in synchronous Clos networks. As indicated by its name, the CRRD algorithm places independent round-robin arbiters on each LI (output link arbiter), IP (input port arbiter) and LO. A simplified description of CRRD is illustrated as follows [90]:

- Phase 1: Matching within IMs.
  - The first iteration
  * Step 1: Non-idle IPs send requests to all output link arbiters.
  * Step 2: Each output link arbiter independently selects an IP.
  * Step 3: Each non-idle IP independently accepts one LI from the received grants.
  - The ith iteration ($i > 1$)
  * Step 1: Unmatched IPs send requests to all output link arbiters.
  * Step 2 and 3: The same as the first iteration.
- Phase 2: Matching within CMs.
  - Step 1: Matched LIs send requests to CMs. Each LO in CMs selects one request and returns a grant.
  - Step 2: In the next cell time, the granted IPs send their cells and other IPs try again.

CRRD ensures that the requests in all IMs are dispatched to all CMs with the same probability using the parallel iterative matching (PIM) algorithm [2] in the matching within IMs.

Figure 6.3 demonstrates an example of the matching within an IM using CRRD. Initially all IPs receive new packets and all LIs are available. As shown in Figure 6.3a, IPs send requests to all available LIs. The output link arbiter in each LI grants one IP according to the received requests. Since these output link arbiters run independently, uneven request distribution where multiple LI arbiters select the same IP can be easily generated as depicted in Figure 6.3b. For those IPs receiving more than one grant, such as IP($i, 1$) in Figure 6.3c, their input port arbiters choose an LI and release others. In



(a) Step 1, request (b) Step 2, grant

(c) Step 3, accept (d) Step 1, next iteration

Figure 6.3: Example of the matching within an IM

this way, the unmatched IPs are able to request again in the next iteration as shown in Figure 6.3d.

The even distribution of CRRD relies on the number of iterations. In the worst scenario when only one match is made in each iteration, an IM needs n iterations to match of all IPs. In practice, the number of iterations is also limited by the cell time. It is common that each iteration takes one clock cycle. A cell time must be longer than n cycles to guarantee the even distribution.

Although the requests from one IM are evenly distributed to all CMs, two requests from different IMs asking for the same OM can be distributed to the same CM competing for the same LO. CRRD produces even request distribution but this distribution is oblivious to the possible contention inside CMs.

6.2.2 Asynchronous dispatching algorithm

Up to now no asynchronous Clos networks have yet been proposed. Reconfiguring an asynchronous Clos network has some fundamental difficulties. The incoming packets arrive asynchronously and should be processed asynchronously. A path is allocated for a packet rather than a cell. Moreover, the scheduler must handle multiple packets concurrently and independently as packets can arrive at any time.

As a solution to these difficulties, a new asynchronous dispatching (AD) algorithm is proposed. In this algorithm, the matching within IMs and the matching within CMs are separated into two independent sub-algorithms running concurrently. All modules are event-driven. Independent arbiters are placed on each LI (output link arbiter), IP (input port arbiter) and LO as the CRRD algorithm does, but these arbiters are multi-way MUTEX arbiters and tree-arbiters.

In the CRRD algorithm, if a request fails to reserve a path due to the contention in CMs, it automatically tries again in the next cell time. However, an asynchronous request cannot withdraw itself until it is served. Directly adopting the CRRD algorithm in asynchronous Clos networks introduces severe arbitration latency because the contention in one CM causes at least one request to wait a whole packet time rather than a cell time even when other CMs are available. To reduce such latency overhead, the AD algorithm introduces a state feedback scheme. Once an LO is occupied or released, the information is broadcast to all IMs. Since IMs are informed of the availabilities of LOs in all CMs, they dispatch requests only to the CMs currently with available LOs. The contention in CMs is accordingly avoided. A simplified description of the asynchronous dispatching algorithm is described as follows [113]:

- Sub-algorithm 1: Matching within IMs.
  - Step 1: A new packet arrives at IP(i, h).
  - Step 2: IP(i, h) waits until at least one target LO is available.
  - Step 3: IP(i, h) sends requests to all output link arbiters leading to the available LOs.
  - Step 4: output link arbiters return grants to IP(i, h).
  - Step 5: IP(i, h) selects a path and withdraws requests to other output link arbiters.
- Sub-algorithm 2: Matching within CMs.
  - Step 1: A request is forwarded from an IM.
  - Step 2: The target LO returns a grant to the IM and reconfigures the CM once it is available.
  - Step 3: The updated states are broadcast to all IMs.

The sub-algorithm running in the IMs uses the same PIM algorithm in CRRD (an asynchronous and event-triggered version) to ensure that requests are evenly distributed to all CMs. The sub-algorithm running in the CMs is also an asynchronous version of the phase 2 in the CRRD algorithm but the state changes of CMs are broadcast to IMs.

An example of the state feedback scheme of the AD algorithm is presented in Figure 6.4. The Clos network in the example is a $\mathbf{C}(3,3,3)$ network. Some links in this network have already been occupied, such as the links on the path from IP$(2,3)$ to LO$(3,3)$ and the path from IP$(3,2)$ to LO$(2,1)$. A new packet is received by IP$(2,1)$ in IM(2) requesting an OP in OM(1). By receiving the state feedback from all CMs, IM(2) knows that CM(2) is not available for the new packet because LO$(2,1)$ is occupied. Thus the request from IP$(2,1)$ heading to OM(1) is not sent to LI$(2,2)$ (which is linked to CM(2)). This is the major difference between CRRD and AD. As other CMs are available, IP$(2,1)$ requests to LI$(2,1)$ and LI$(2,3)$ as illustrated in Figure 6.4a. Obviously the arbiter on LI$(2,3)$ does not respond to this request as LI$(2,3)$ is occupied. As shown in Figure 6.4b, only the arbiter on LI$(2,1)$ returns a grant to IP$(2,1)$. Finally in Figure 6.4c, a request from IP$(2,1)$ is sent to CM(1) through LI$(2,1)$. The LO$(1,1)$ in CM(1) is correspondingly reserved. This new path reconfiguration is then broadcast to all IMs.

The state feedback scheme, which is the major difference between CRRD and AD, can improve throughput by avoiding the contention in CMs. However, it cannot resolve the contention among the requests processed simultaneously because they use the same state feedback. In other words, the state feedback avoids contention between established paths and future requests but cannot resolve the existing contention. If contention occurs, multiple requests from different IMs are sent to the same CM competing for the same LO. In this case, the arbiter on the LO grants only one request and forces others to wait until the granted request is withdrawn. The arbitration latency for the blocked requests is prolonged but they will be served eventually.

It should be noticed that the number of simultaneous requests in asynchronous Clos networks is significantly smaller than synchronous Clos networks due to their asynchronous nature. In synchronous Clos networks, all requests are synchronized; therefore, the number of simultaneous requests is the total number of active requests. On the other hand, asynchronous Clos networks are not synchronized. When the network load is low, the time to establish a path is much shorter than the time to transmit a packet. The process of establishing a path can be recognized as an event. It is rare for

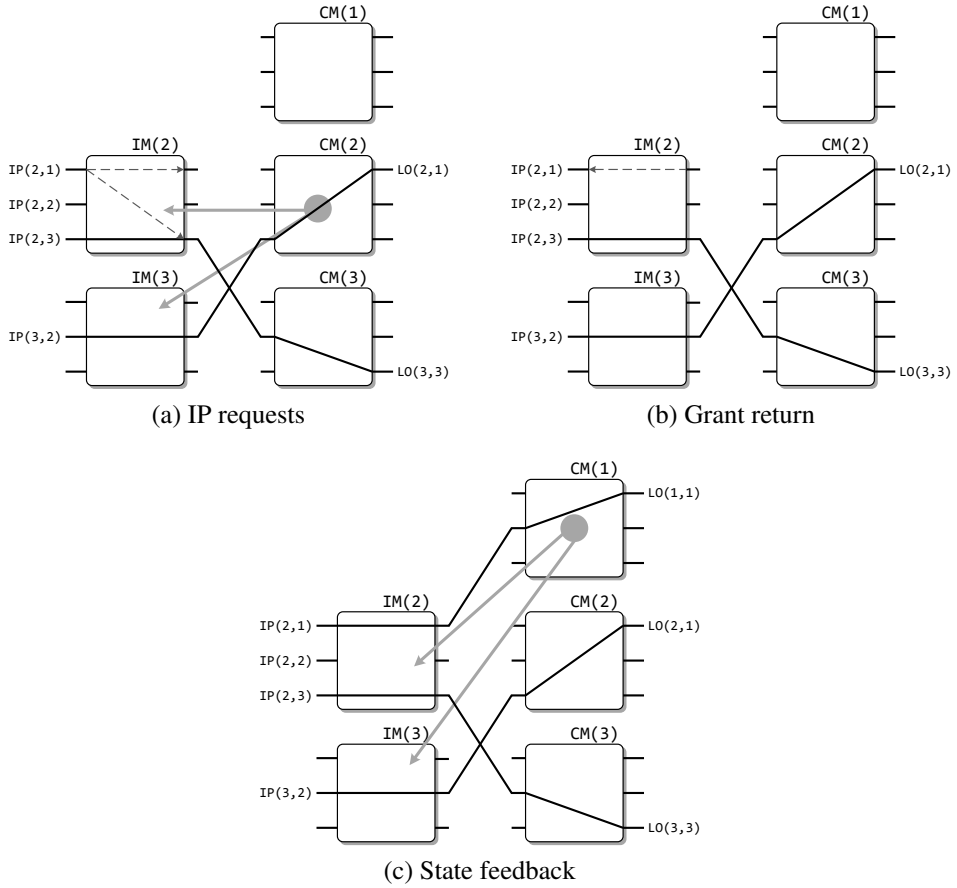


Figure 6.4: State feedback scheme of the asynchronous dispatching algorithm

two events to occur at exactly the same time. When the network is saturated, the number of simultaneous requests increases as many requests are blocked. Nevertheless, the number of simultaneous requests is still much smaller than the number in synchronous Clos networks as nearly half IPs are busy transmitting data (49% throughput in uniform traffic as will be shown in Figure 6.7). Using the placed and routed implementations of the synchronous and asynchronous Clos schedulers in Section 6.3.1, the CM contention rate (the ratio of the number of conflicted requests sent to CMs to the number of all CM requests) of the saturated Clos networks is extracted from post-layout simulations. The rates are 56.7% and 24.9% for the synchronous and asynchronous schedulers respectively. It is shown that the state feedback scheme successfully reduces the contention significantly.

6.2.3 Performance of CRRD and AD

The asynchronous dispatching (AD) algorithm is the first algorithm that reconfigures an asynchronous 3-stage Clos network. It is therefore essential to demonstrate the throughput performance of AD by comparing it against a classic synchronous algorithm, such as CRRD.

Both AD and CRRD are implemented to reconfigure a $\mathbf{C}(4, 8, 4)$ space-space-space Clos network in behavioural level SystemC models [92]. The Clos networks will be injected with various traffic patterns. Some assumptions are employed to produce a fair comparison between synchronous and asynchronous Clos networks [113]:

- *Pseudo-random arbiters are utilized in both models.*
  Synchronous and asynchronous circuits have fundamental differences. The implementation of synchronous dispatching algorithms uses round-robin arbiters while that of asynchronous algorithms uses multi-way MUTEX and tree arbiters. All these arbiters are hardware designs approximating random arbiters. Pseudo-random arbiters are directly utilized in SystemC models.

- *A request is withdrawn immediately after a path is allocated for it.*
  The data transmission in synchronous Clos networks and in asynchronous Clos networks has fundamental differences. In synchronous Clos networks, switch reconfiguration and data transmission are pipelined. Thus network throughput is independent of path allocation latency as long as it is shorter than a cell time. On the contrary, switch reconfiguration and data transmission cannot be pipelined in asynchronous Clos networks because a path reservation cannot be prearranged before it is available. Long path allocation latency leads to low network throughput. As a result, even when the same routing algorithm is implemented in both synchronous and asynchronous Clos networks with the same path allocation latency, asynchronous Clos networks show worse network throughput than synchronous Clos networks. Since routing algorithms (including dispatch algorithms) rather than the whole Clos network are the major research objective in this section, the throughput difference due to the pipelined switch allocation is eliminated by assuming no data transmission (a path is released immediately after allocation).

- *A frame comprises only one cell.*
  A path in a synchronous Clos network is allocated for transmitting a cell while it

is allocated for a frame in an asynchronous Clos network. The frame size is set to one cell to eliminate this difference.

Non-blocking uniform traffic is a synthetic traffic pattern to reveal the sources of throughput loss in different dispatching algorithms. The traffic pattern ensures that no OP is concurrently requested by more than one IP; therefore, no contention is produced by the traffic pattern. The expected saturation throughput should be 100% (all packets injected are transmitted with the minimum delay) in a non-blocking Clos network controlled by an optimal algorithm. Any throughput loss observed in evaluation is caused by the imperfection of heuristic algorithms.

The non-blocking uniform traffic can be described as:

$$E[\rho(t,s,d)] = \frac{E[\rho(t,s)]}{N} \tag{6.3}$$

$$\sum_{s=1}^{N} \rho(t,s,d) \leq 1 \tag{6.4}$$

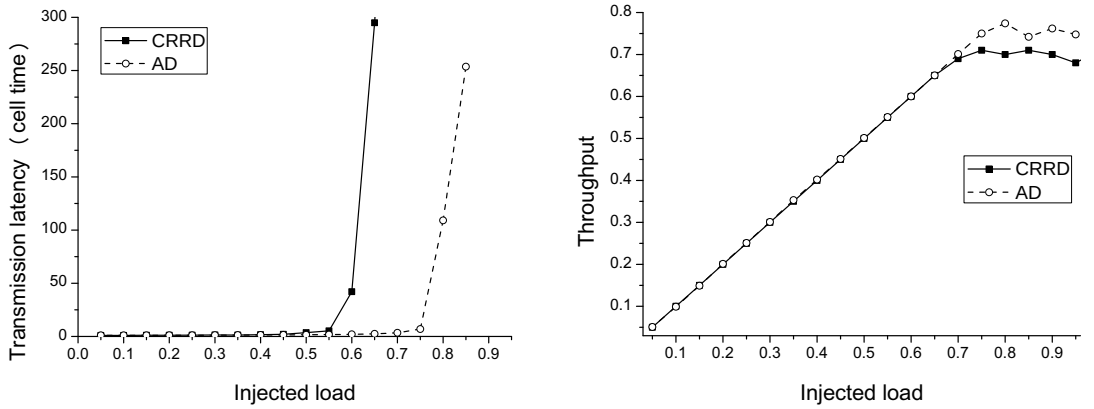
where N is number of IPs/OPs, $\rho(t,s)$ is the normalized load injected in IP(s) at time t, and $\rho(s,d)$ is the normalized load from IP(s) to OP(d) at time t. All IPs are injected complying with a Poisson process with a fixed injection rate ρ,

$$\rho = \int_{t=0}^{\infty} \rho(t,s)dt = E[\rho(t,s)] = \rho(s) \tag{6.5}$$

ρ is the normalized injection rate of the Clos network value from 0 to 1. A packet is injected to an IP in every cell time[1] when $\rho = 1$ and no packet is injected when $\rho = 0$. As described by Equation 6.3, the load of an IP is uniformly distributed to all OPs. Equation 6.4 is a non-blocking constraint ensuring that no OP is overloaded at any time.

Figure 6.5 shows the packet transmission latency and throughput with non-blocking uniform traffic. The number of iterations in the CRRD dispatching algorithm is set to four to ensure the maximum throughput [90]. The saturation throughput of CRRD and AD is 70% and 76% respectively. AD demonstrates better throughput than CRRD thanks to the state feedback scheme. Nevertheless, both algorithms show throughput

[1]Since a packet comprises one cell, a cell time is the minimum transmission latency of a packet. It is a virtual time unit in behavioural simulations but practical delays are back-annotated in Section 6.3.2.



(a) Transmission latency (b) Throughput

Figure 6.5: Performance with non-blocking uniform traffic

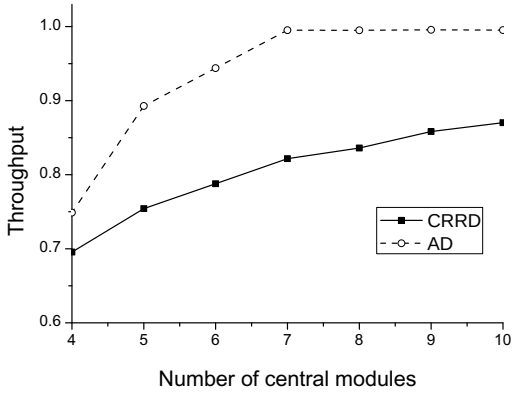


Figure 6.6: Throughput with various number of central modules

loss even when the traffic is non-blocking. There are two reasons for this throughput loss: firstly, established paths cannot be modified; therefore, the RNB $\mathbf{C}(4, 8, 4)$ network is blocking. Secondly, heuristic algorithms, such as the CRRD algorithm, produce contention.

Increasing the number of central modules converts a RNB network into an SNB one. Since no alterations to the established paths are needed to build a new path in an SNB network, the contention caused by dispatching algorithms is the only source of throughput loss. Figure 6.6 depicts the throughput increase gained by adding extra central modules. $\mathbf{C}(4, 8, m \geq 7)$ Clos networks are SNB. It is shown that the AD algorithm introduces no throughput loss in an SNB network but the CRRD algorithm cannot provide 100% throughput even with ten central modules.

The traffic patterns in practical applications are blocking. Uniform traffic is one of the most analysed patterns. It can be defined by Equation 6.3 and Equation 6.5. All

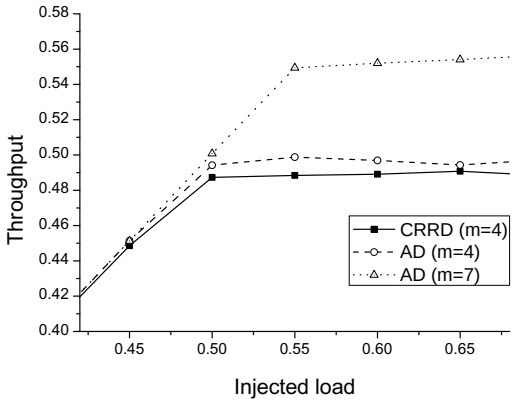


Figure 6.7: Throughput with uniform traffic

IPs are injected with a packet sequence complying with the same Poisson process as the non-blocking uniform traffic. The load on one IP is also uniformly distributed to all OPs but OPs are not protected by the non-blocking constraint described in Equation 6.4. Consequently, OPs are overloaded occasionally.

Figure 6.7 reveals the throughput with uniform traffic. The saturation throughput of using CRRD and AD in a $\mathbf{C}(4, 8, 4)$ Clos network is 48.9% and 49.7% respectively. AD shows 0.8% higher throughput than CRRD. It is known that the optimum saturation throughput of an input-queued switch[2] is 58.6% [62]. As shown in Figure 6.7, using AD in an SNB Clos network achieves 55.4% throughput, which is only 3.2% lower than the optimum throughput. Reducing the number of central modules introduces 5.7% throughput loss.

In summary, the behavioural level simulations shows that the asynchronous dispatching algorithm demonstrates better throughput than the concurrent round-robin dispatching algorithm.

6.3 Asynchronous Clos scheduler

This section will provide the hardware detail of the first asynchronous scheduler which is used to dynamically reconfigure a 32-port $\mathbf{C}(4, 8, 4)$ space-space-space Clos network using the asynchronous dispatching algorithm [113].

[2]The original analyses were done on crossbars using input queues (FIFOs). A space-space-space Clos network is equivalent to an input queued crossbar.

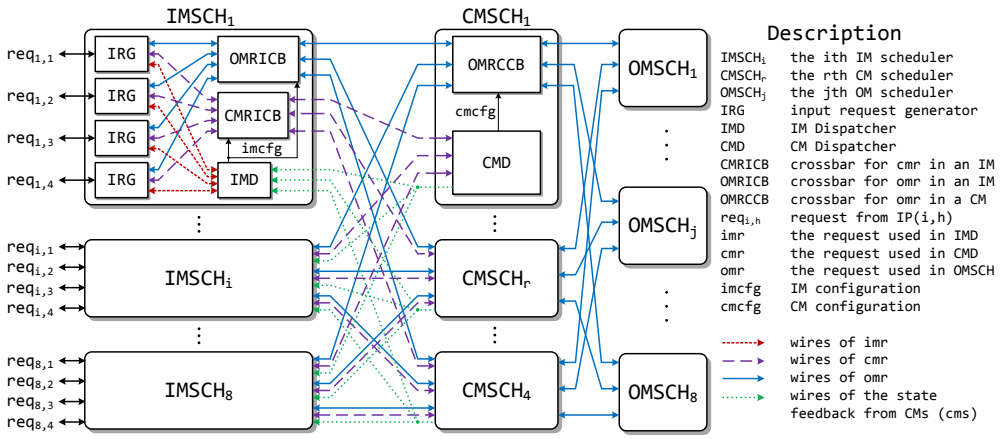


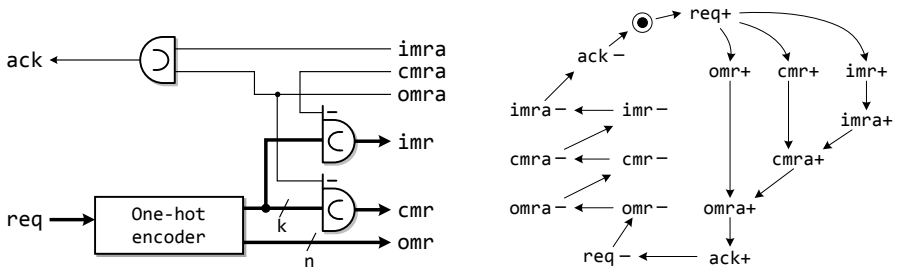
Figure 6.8: An asynchronous Clos scheduler for $\mathbf{C}(4, 8, 4)$

6.3.1 Implementation

The overall structure of the Clos scheduler is shown in Figure 6.8. It adopts a distributed hardware structure where every switch module has its own scheduler, such as the input module scheduler (IMSCH), the central module scheduler (CMSCH) and the output module scheduler (OMSCH).

An IMSCH comprises n input request generators (IRGs), an input module dispatcher (IMD) and two $n \times m$ bidirectional crossbars. An IRG converts the request from an IP (*req*) into three independent requests — IM request (*imr*), CM request (*cmr*) and OM request (*omr*) — for IMSCH, CMSCH and OMSCH respectively. It also controls the timing of these requests in order to enforce safe reconfiguration operations. An IMD dynamically reconfigures an input module adhering to the asynchronous dispatching algorithm. It receives the *imr* signals generated by IRGs and produces the reconfiguration signal *imcfg*. Meanwhile, the *cmr* forwarding crossbar (CMRICB) and the *omr* forwarding crossbar (OMRICB) deliver *cmr* and *omr* to the second stage, the central modules, according to *imcfg*. Note that all request signals have their ack lines, namely *ack* for *req*, *imra* for *imr*, *cmra* for *cmr* and *omra* for *omr*.

An CMSCH includes a central module dispatcher (CMD) and a $k \times k$ bidirectional crossbar — the *omr* forwarding crossbar in CM (OMRCCB). A CMD dynamically reconfigures a central module and OMRCCB adhering to the asynchronous dispatching algorithm. It receives the *cmr* forwarded from IMs, generates the local CM reconfiguration *cmcfg* and broadcasts its state back to IMs through the CM state feedback signal *cms*.



(a) Schematic (b) STG

Figure 6.9: Input request generator

The structure of an OMSCH is similar to that of a CMD. It receives the *omr* forwarded from CMs and dynamically reconfigures an output module.

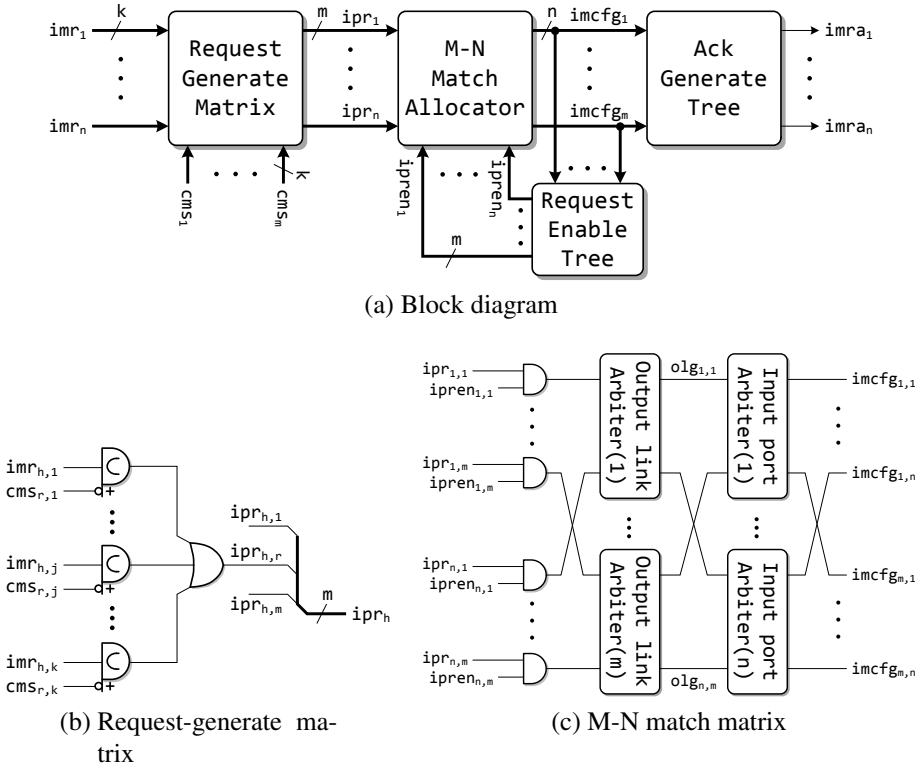
Input request generator

Any IP in a Clos network can request any one of the nk OPs. As shown in the basic structure of a Clos network depicted in Figure 6.2, the nk OPs are grouped into k OMs. The asynchronous dispatching algorithm allocates a path from the requesting IP to the target OM. Then the OMSCH allocates a path towards the target OP. Accordingly, the request from an IP should be converted into two requests: one for the dispatching algorithm to identify the OM and the other one for the OMSCH to identify the OP. This work is handled by the input request generator (IRG) connected to each IP.

The structure of an IRG is illustrated in Figure 6.9. A one-hot encoder translates an incoming request into two requests encoded in one-hot: a 1-of-k request for *imr* and *cmr* and a 1-of-n request for *omr*. C-elements are added to request and ack lines to ensure the timing order as shown in Figure 6.9b. The requests to all switch modules are sent simultaneously but the order of releasing switch modules has to be guaranteed. Specifically, the OM must be released before the CM and the CM must be released before the IM. Otherwise, the next request can be mis-routed.

Input module dispatcher

An input module dispatcher (IMD) dynamically reconfigures an IM using the asynchronous dispatching algorithm. It receives *imr* from IRGs, allocates LIs to IPs according to the state feedback *cms* from CMs, and generates the configuration *imcfg* along with the ack line *imra*. As shown in Figure 6.10a, it consists of four components: a request-generate matrix, an M-N match allocator, a request enable tree and an



(a) Block diagram

(b) Request-generate matrix

(c) M-N match matrix

Figure 6.10: Input module dispatcher

ack generate tree. The request-generate matrix filters out the requests to unavailable CMs. The M-N match allocator randomly allocates LIs to incoming requests. Consequently the requests are acknowledged by the ack generate tree. The algorithm running in the M-N match allocator has a feedback loop inside which the feedback signal *ipren* is generated by the request enable tree.

Figure 6.10b demonstrates the internal structure of the request-generate matrix. An incoming request from IP(h) is denoted by k bits, $\{imr_{h,k}, \cdots, imr_{h,1}\}$, each of which denotes a request to a target OM. The state feedback from CMs is identified by a $m \times k$ matrix *cms* where a positive $cms_{r,j}$ represents that the LO(r, j) in CM(r) is already occupied. Since any I/O pair has m different paths through the m CMs, every $imr_{h,j}$ is verified with m LOs denoted by $\{cms_{m,j}, \cdots, cms_{1,j}\}$ using the asymmetric C-elements depicted in Figure 6.10b. The output of the request-generate matrix is a matrix of request bits *ipr* where a positive $ipr_{h,r}$ denotes an active request from IP(h) to CM(r). Every $ipr_{h,r}$ is or-reduced from the results of the C-elements. Since incoming requests are encoded one-hot, only one bit in the k bits imr_h can be high. The or-reduced *ipr* is able to represent all active requests.

Figure 6.10c shows the M-N match allocator which is able to match m resources

to n clients concurrently. It is an asynchronous implementation of the parallel iterative matching algorithm [2] using a structure similar to the speed-independent forward acting $n \times m$ arbiter [95]. It consists of two columns of multi-way MUTEX arbiters: one column of m output link arbiters and another column of n input port arbiters. Input requests *ipr* are shuffled and sent to output link arbiters. The output link arbiter of every LI chooses an IP independently. Since an IP can initiate multiple requests in the *ipr* matrix if multiple CMs are available, the arbitration results of the output link arbiters, represented by the *olg* in Figure 6.10c, have to be arbitrated again using the input arbiters in the second column. They ensure that one IP is matched with one and only one LI. In case when multiple output link arbiters choose the same IP, the configuration *imcfg* would withdraw the corresponding request enable signals *ipren*, which consequently withdraws the requests to the output link arbiters not selected by the input arbiter. These LIs can therefore be used by other IPs.

The generation of *ipren* from *imcfg* is processed in the request enable tree. An OR gate tree is built for each *ipren* using Equation 6.6:

$$ipren_{h,r} = \neg(\bigcup_{l=1,l\neq r}^{m} imcfg_{l,h}) \tag{6.6}$$

Therefore, $ipren_{h,r}$ is immediately withdrawn when any LI but LI(r) is allocated to IP(h).

The ack line *imra* is also generated from *imcfg* in the ack generate tree. Similar to the request enable tree, an OR gate tree is built for each *imra* using Equation 6.7:

$$imra_h = \bigcup_{l=1}^{m} imcfg_{l,h} \tag{6.7}$$

It should be pointed out that the M-N match allocator is relaxed-QDI rather than QDI. The multi-resource arbiter [53, 52, 104] (Section 2.7.2) is the only QDI allocator available so far that can allocate multiple resources to multiple clients. A modified version of the multi-resource arbiter was used in the previous Clos scheduler published in [113]. However, it has several design problems: The area of the multi-resource arbiter is so large that the area of IMDs consumes 67.2% of the total area of the Clos scheduler presented in [113]. The handshake protocol of the multi-resource arbiter is not fully satisfied in the Clos scheduler and the ring based multi-resource arbiter

proposed in [104] generates livelock[3]. The allocation speed of a multi-resource arbiter is slow because only one IP is served at one time. Contrarily, the M-N match allocator serves multiple IPs concurrently. It uses a similar structure as the forward acting $n \times m$ arbiter [95] but is simplified using timing assumptions.

The timing assumption required for the M-N match allocator is due to the unguarded withdrawal of requests and *olg*. Figure 6.11 shows the STG of a 2×2 M-N match allocator with two requests (ipr_1 and ipr_1) and two resources. The state feedback is not considered in this STG for simplicity reasons; therefore a request is always forwarded to all output link arbiters. The withdrawal of requests and *olg* is shown in slim blue lines. When both output link arbiters have chosen the same request, the generated *imcfg* will release one of them. As an example, assuming both output link arbiters have chosen ipr_1, $olg_{1,1}$ and $olg_{1,2}$ are driven high. Only one of them can pass the input arbiter (1). Assuming the second resource is selected, $imcfg_{2,1}$ is set and a token is produced to release $olg_{1,1}$. In a more complicated situation, $olg_{1,1}$ may not be high as output link arbiter (1) is currently reserved by ipr_2. However, a token is still produced to withdraw the duplicated ipr_1 as ipr_2 can drop at any time making the output link arbiter (1) select ipr_1.

To prohibit any false configuration from being produced, the withdrawal procedure must finish before $ipr-$, which is denoted in Figure 6.11 by dash lines coloured in red. However, implementing such completion detection leads to a significant number of C-elements and OR gates inserted into the allocator. Considering that the withdrawal procedure should be finished as soon as possible in order to allocate the resources to other requests, the latency of the withdrawal procedure is actually the equivalent cycle time of an iteration in the asynchronous PIM algorithm. This latency is far shorter than the serving time of a request. It is safe to let the withdrawal procedure be unguarded. The timing assumption of the safe but unguarded request withdrawal can be expressed as follow:

$$t_{imcfg+ \rightarrow olg-} < t_{imcfg+ \rightarrow ipr-} \tag{6.8}$$

Considering the timing sequence shown in Figure 6.9b, the right side of Equation 6.8 is the accumulative latency of reserving the rest of a path in CMs and OMs, along with the whole data transmission. The left side of Equation 6.8 is merely the accumulative

[3]If the multi-resource arbiter is utilized in IMD, the state feedback signals *cms* are used as client request signals. A client request cannot be released before receiving an ack generated from *imcfg*. Since the state of CMs can be altered because of other IMs, the non-withdrawal requirement is not satified. A false client request can occur and trigger livelock especially when the ring based multi-resource arbiter is used.

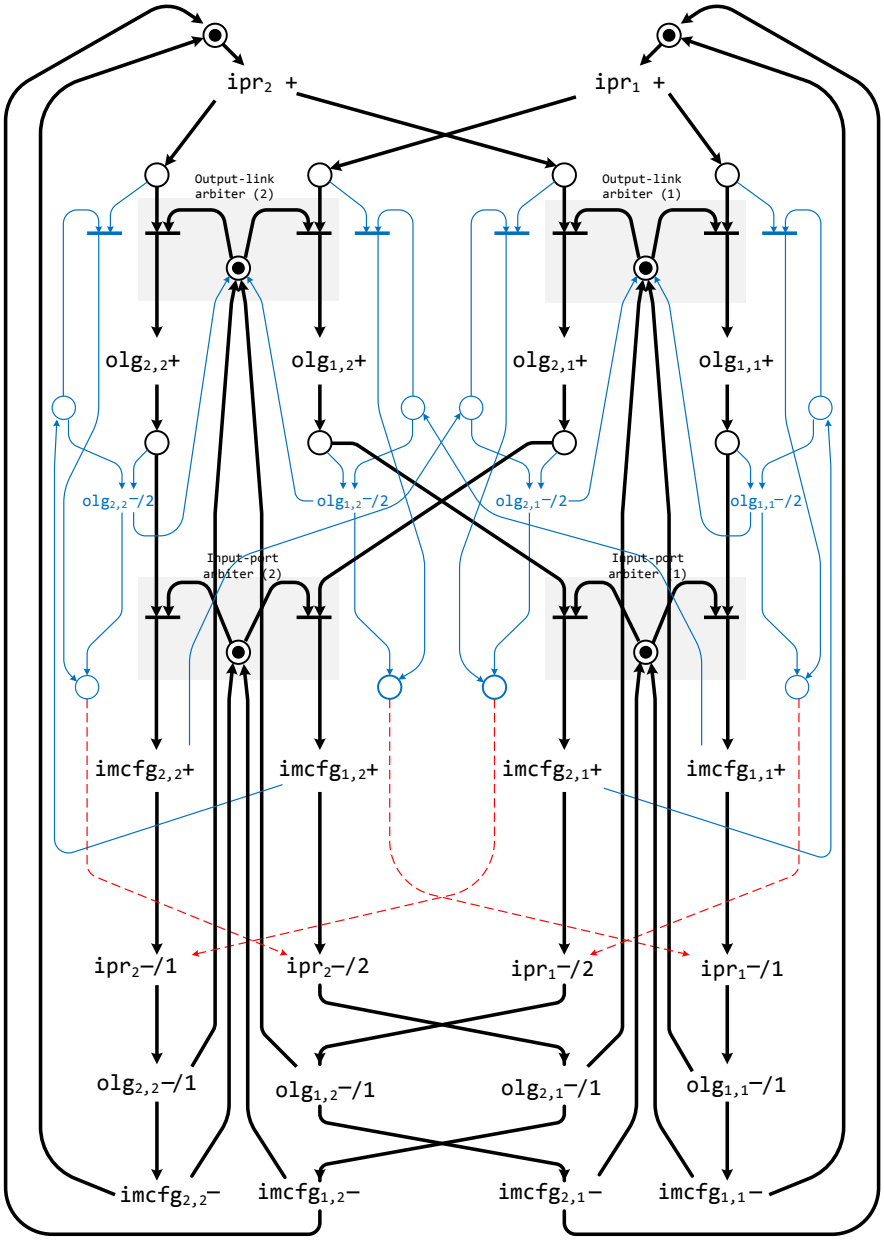


Figure 6.11: STG of a 2×2 MNMA

latency of the request enable tree and an output link arbiter. The speed evaluation in Section 6.3.2 will demonstrate that the latency of the right side is far longer than the left side even without data transmission. Enough timing margin has been provided by the design to ensure the timing assumption without inserting matched delay lines.

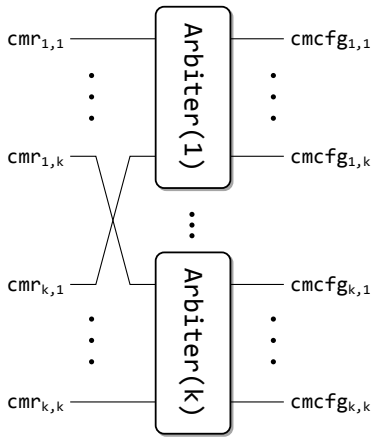


Figure 6.12: Central module dispatcher

Central module dispatcher and output module scheduler

A central module dispatcher dynamically reconfigures a central module according to the *cmr* forwarded from input modules. It also broadcasts its state alteration to all input modules. The bidirectional crossbar OMRCCB in a central module scheduler delivers *omr* to output modules. An output module scheduler dynamically reconfigures an output module corresponding to the forwarded *omr*.

A central module dispatcher and an output module scheduler have similar internal structures. They all contain a group of arbiters to generate the dynamic configuration and groups of OR gate trees. Figure 6.12 demonstrates the arbiters in a central module dispatcher. They receive *cmr* and produce *cmcfg*. In an output module scheduler, the same structure is used to produce *omcfg* according to *omr*.

A central module dispatcher also generates the ack lines *cmra* and the state feedback *cms*. Each bit of these signals is produced by an OR gate tree using the following equations:

$$cmra_i = \bigcup_{l=1}^{k} cmcfg_{l,i} \tag{6.9}$$

$$cms_j = \bigcup_{l=1}^{k} cmcfg_{j,l} \tag{6.10}$$

Table 6.1: Area consumption

	Asynchronous Scheduler	Asynchronous Scheduler in [113]	Synchronous Scheduler [90]
One IMD	3,862	21,882	3,186
One CMD	4,879	8,437	6,498
One OMSCH	985	1,258	1,375
One IRG	160	196	
Total	88,057	260,740	115,262

(unit: μm^2)

Similarly, the ack lines for *omr* are produced by OR gate trees in output module schedulers using the following equation:

$$omra_r = \bigcup_{l=1}^{n} omcfg_{l,r} \tag{6.11}$$

6.3.2 Performance

The asynchronous Clos scheduler for a $\mathbf{C}(4, 8, 4)$ space-space-space Clos network has been implemented and compared against a reproduction of the synchronous Clos scheduler using the CRRD algorithm [90]. Both designs are written in Verilog HDL, synthesized, placed and routed with commercial EDA tools using the Faraday $0.13\ \mu m$ standard cell library. Accurate latency, throughput and power consumption are obtained from back-annotated post-layout simulations.

The synchronous Clos scheduler runs at a maximum of $300\ \mathrm{MHz}$ after optimization. The clock period in post-layout simulations is set to $3.5\ \mathrm{ns}$ ($285\ \mathrm{MHz}$) to leave a timing margin and avoid significant area overhead on optimizing critical paths. The number of iterations in the CRRD algorithm is dynamically reconfigurable but fixed to four in simulations for the maximum throughput performance. The cell time in the synchronous Clos network is $17.5\ \mathrm{ns}$ according to the following equation:

$$t_{\text{cell}} = t_{\text{clock}} \cdot (I + 1) \tag{6.12}$$

where I is the number of iterations.

The post-layout area of both Clos implementations are illustrated in Table 6.1. The area of the asynchronous Clos scheduler using the multi-resource arbiter is also listed in Table 6.1.

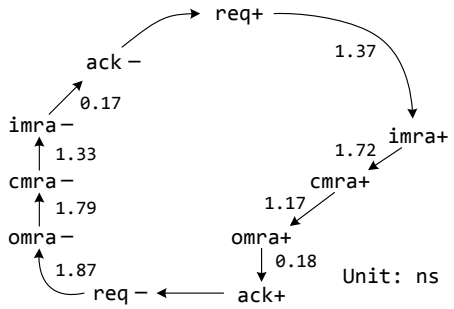


Figure 6.13: Detailed allocation latency

It is shown that the asynchronous Clos scheduler is smaller than its synchronous counterpart. This small area is caused by several reasons:

- The CRRD algorithm runs in multiple iterations. The intermediate states between iterations are stored in flip-flops.
- Since the path allocation and the data transmission are pipelined, the Clos configuration generated for the next cell time is stored in flip-flops.
- A clock tree is need to drive all flip-flops.
- The storage elements in asynchronous circuits are C-elements which are smaller than the flip-flops used in synchronous circuits.

Due to these reasons, all asynchronous scheduler components consume less area than synchronous ones, except for the input module dispatcher. The request-generate matrix in every IMD filters out the requests to unavailable LOs using the state feedback from CMs. Each the request-generate matrix contains $n \times k \times m$ asymmetric C-elements, which cause the extra area. It is also shown that the new asynchronous Clos scheduler using the M-N match allocator achieves a significant area reduction of 66% from the previous implementation which uses the multi-resource arbiter [113].

The detailed latency of allocating a path using the asynchronous Clos scheduler is labelled in the simplified STG shown in Figure 6.13. The transitions from *req*+ to *ack*+ denote that a path is reconfigured for an incoming packet while the transitions from *req*− to *ack*− denote that the path is safely released. The delay of the data transmission of a packet, occurring between *ack*+ and *req*−, is variable depending on the payload size of the packet. The delays labelled in the STG are averaged from allocating paths from different IPs to various OPs in an idle Clos network. Extra delays can be introduced when the Clos network is loaded with heavy traffic.

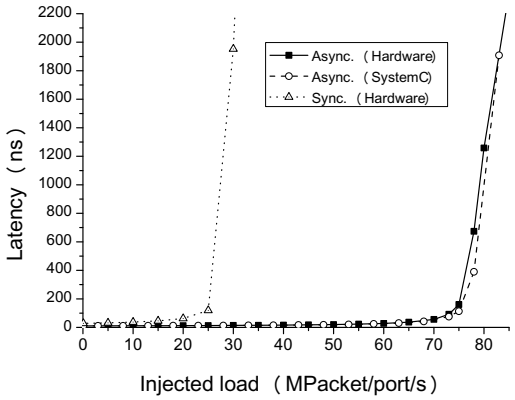


Figure 6.14: Latency of Clos schedulers

As shown in Figure 6.13, the asynchronous Clos scheduler can reserve a path in 4.44 ns and release it in 5.16 ns. The minimum allocation period is 9.6 ns, which is shorter than the 10.1 ns period of the previous design [113]. Apropos of the timing assumption of the safe but unguarded request withdrawal in the M-N match allocator, the latency $t_{imcfg+\rightarrow olg-}$ in Equation 6.8 is around 0.69 ns, which is far shorter than the 6.73 ns latency of the right side without data transmission.

Assuming requests are withdrawn immediately after paths are reconfigured (the same assumption used in behavioural simulations), Figure 6.14 reveals the allocation latency with different injected loads. The detailed delays in Figure 6.13 are back-annotated into the behavioural level SystemC model used in Section 6.2.3. The latency of the back-annotated model is also shown in Figure 6.14. All schedulers are injected with uniform traffic pattern.

As shown in Figure 6.14, the back-annotated SystemC model of the asynchronous Clos network accurately matches the post-layout simulation. It is also shown that the asynchronous Clos scheduler provides larger saturation throughput with shorter allocation latency than the synchronous Clos scheduler. The asynchronous Clos scheduler can allocate a maximum of 77.3 MPacket/port/s while it is 27.3 MPacket/port/s for the synchronous Clos scheduler with four iterations.

Figure 6.15 reveals the power consumption of both schedulers with different injected load. The asynchronous Clos scheduler consumes significantly lower power than the synchronous Clos scheduler. The clock tree in the synchronous Clos scheduler consumes 8.34 mW, which is 63.6% of the total power consumption when the scheduler is injected with 20 MPacket/port/s load. In average, allocating 1 M packets in the asynchronous Clos scheduler consumes 64.4 µJ, which is 35.8% of the energy consumed by the synchronous Clos scheduler allocating the same number of packets.

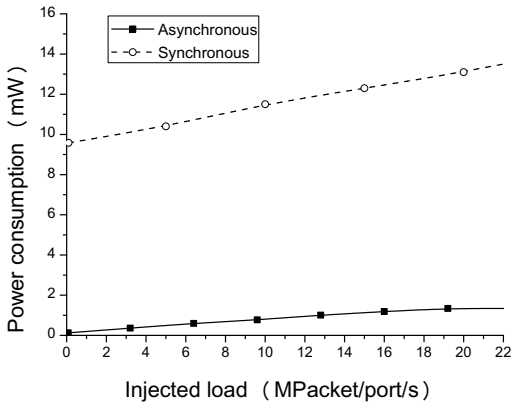


Figure 6.15: Power consumption of Clos schedulers

The average energy for 1 M packets in the synchronous Clos scheduler is 180 µJ.

6.4 2-stage Clos swtich

It has been shown at the beginning of this chapter that using Clos networks reduces the area overhead of SDM routers. However, simply substituting the crossbar in an SDM router with a space-space-space Clos network is not the best solution. A 3-stage Clos network consumes the minimum area when $k = \sqrt{2N}$. The optimum k for a practical switch in an SDM router is usually not an integer, which leads to non-minimum Clos networks. On the other hand, the crossbar of a router using DOR routing algorithms can be simplified by removing disabled turns (see Section 7.1.2 and Figure 7.5 for a detailed explanation of the turn models). Such area reduction cannot be directly applied to a 3-stage Clos network.

A novel 2-stage Clos switch structure for a 5-port SDM router is depicted in Figure 6.16. Rather than building an optimum Clos network by setting the number of input modules to $\sqrt{2N}$, it is fixed to the number of directions. In a mesh network using 5-port routers, this number is fixed to five (**s**outh, **w**est, **n**orth, **e**ast and **l**ocal). Every direction has an input module sized $M \times M$, where M is the number of virtual circuits implemented in each direction. The central stage comprises M 5×5 central modules. An output module can be implemented for each direction. However, the virtual circuits in each output module are equivalent as they have the same output direction. LOs can be directly connected to OPs; therefore, output modules are not necessary and can be removed. The 2-stage Clos network is RNB because the number of central modules is equal to the number of virtual circuits in an input module. The area of a switch is

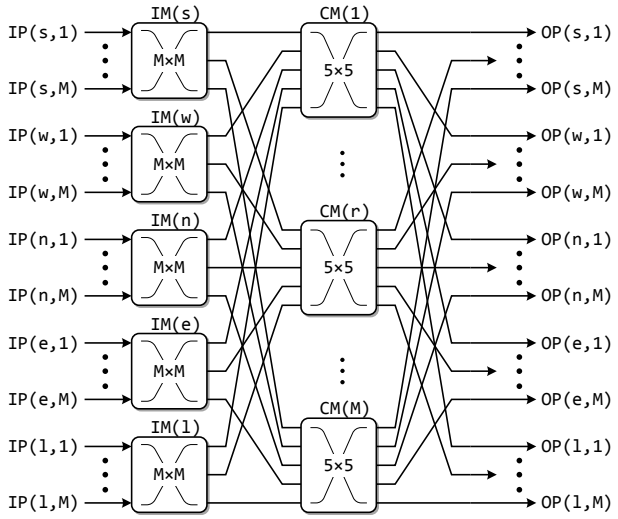


Figure 6.16: 2-stage Clos switch

proportional to the number of cross-points. The area of the 2-stages Clos switch is:

$$A_{\text{Clos},2-\text{stg}} \leq (PM^2 + MP^2) \cdot (W/M) \cdot A_{CP} \tag{6.13}$$

where P is the number of directions, W is the wire count of each virtual circuit, and A_{CP} is the equivalent area of a single cross-point.

Compared with the space-space-space Clos network, the 2-stage Clos switch has several advantages:

- As shown in Figure 6.17, the 2-stage Clos switch consumes the smallest area in 5-port SDM routers as long as $M \leq 18$.
- The 2-stage structure simplifies the Clos scheduler and reduces the allocation latency.
- Every central module is equivalent to the crossbar in a wormhole router. The disabled turns can be removed in central modules for further area reduction.
- The latency of transmitting data through the switch is also reduced due to the removal of output modules.

Similar 2-stage Clos networks have been utilized in optical packet switching networks [22] and ATM switches [101]. An SDM router using this 2-stage Clos switch will be presented in Chapter 7.

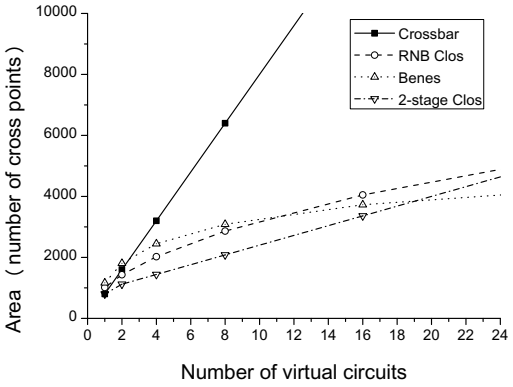


Figure 6.17: Area of different switches (including the 2-stage Clos switches)

6.5 Summary

Spatial division multiplexing significantly improves the throughput of asynchronous on-chip networks but the area of the central crossbar in an SDM router increases proportionally with the number of virtual circuits. Substituting the crossbar with a Clos switch reduces the area overhead. The first asynchronous dispatching algorithm has been proposed and implemented to reconfigure general 3-stage space-space-space Clos networks. Thanks to the state feedback scheme, the asynchronous dispatching algorithm avoids the contention between input modules and central modules. Both behavioural and practical implementations show that the asynchronous dispatching algorithm outperforms the synchronous concurrent round-robin dispatching algorithm in throughput, allocation latency and power consumption. A novel 2-stage Clos switch has been proposed for SDM routers. This rearrangeable non-blocking Clos switch has the smallest area among all the switch structures as long as the number of virtual circuits in each direction is fewer than 18.

Part III

Performance Evaluation and Conclusion

Chapter 7

An Asynchronous SDM Router

The previous chapters have proposed several new techniques to improve the throughput of asynchronous routers. In the physical layer, channel slicing removes the unnecessary synchronization among sub-channels. The lookahead pipeline reduces the period of a pipeline using the early evaluated acknowledgement. In the switching layer, it is found that using SDM achieves better throughput than using VCs. Using Clos switches inside SDM routers reduces the area overhead significantly. In this chapter, a new SDM router will be presented using all the techniques introduced in previous chapters.

7.1 Router structure

Chapter 5 has demonstrated an SDM router using a crossbar as the central switch. As described in Chapter 6, 2-stage Clos switches can be used to reduce the total area. A new SDM router using the 2-stage Clos switch is shown in Figure 7.1.

Compatible with the mesh topology, the router has five bidirectional ports (directions): south, west, north, east and local. Every input port is connected to an input buffer and every output port is connected to an output buffer. The input buffer contains several pipeline stages to buffer incoming flits. A pipeline stage is implemented in each output buffer to decouple the timing between the internal switch and external links. Input and output buffers are spatially divided into M virtual circuits operating independently. The channel slicing technique has been utilized in each virtual circuit and the lookahead pipeline is used to implement the pipeline stages connected to the central switch in the same way as described in Section 4.4.1. Input and output buffers are dynamically connected through the 2-stage Clos switch, which is controlled by a Clos scheduler using the asynchronous dispatching algorithm.

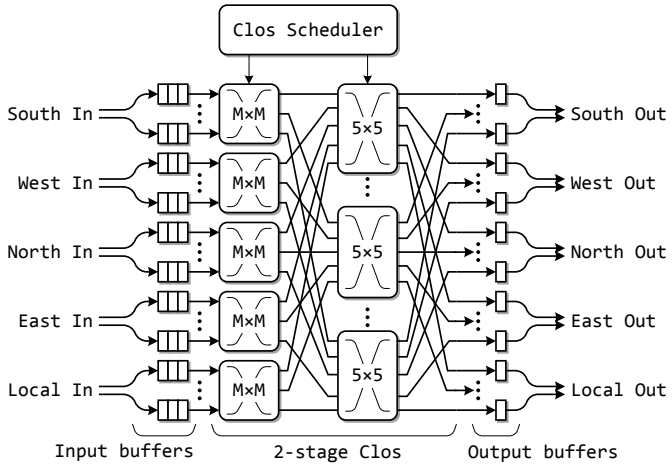


Figure 7.1: Asynchronous SDM router using 2-stage Clos switch

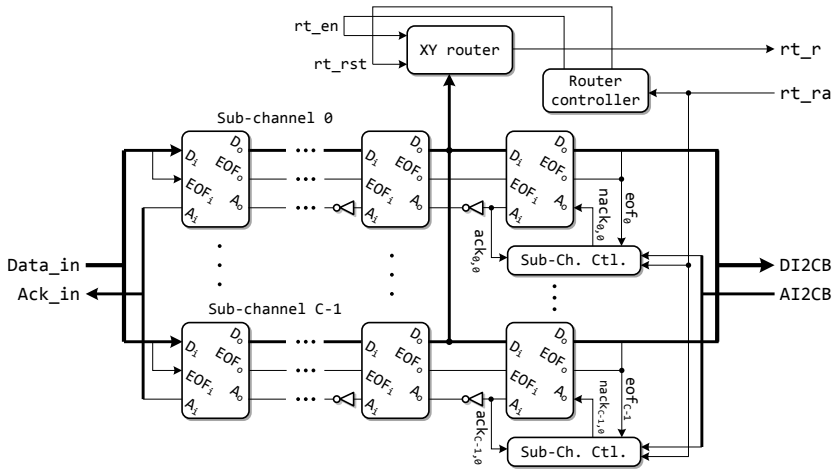
7.1.1 Input and output buffers

The internal structure of the input buffer for a virtual circuit is illustrated in Figure 7.2. It is developed from the input buffer of an SDM router as shown in Figure 5.4.

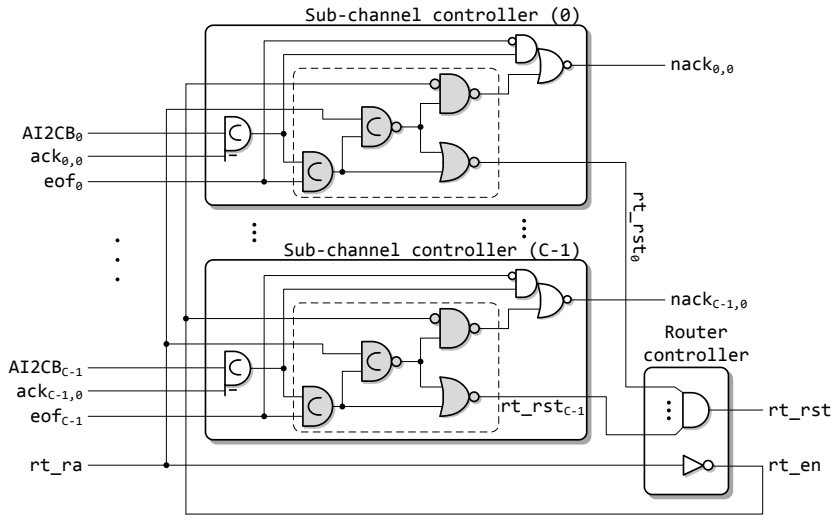
Every pipeline stage is spatially divided into C sub-channels. As described in Section 4.2 and Section 4.4, every sub-channel is a single multi-rail pipeline. In this implementation, it is a 4-phase 1-of-4 pipeline with its own *eof* bit and ack line. The data width of each sub-channel is two bits; therefore, the total data width of a virtual channel is $2C$.

Sub-channels run independently most of the time except for when a tail flit is detected and, therefore, a new header flit is going to be analysed by the XY router (the same one used in Section 4.4 and Section 5.3). The control sequence has been depicted by the STG depicted in Figure 5.5b. Individual sub-channels have to detect the tail flit independently using their own *eof* bits and pause themselves afterwards. Corresponding control logic in the original router controller (Figure 5.5a) is duplicated and developed into sub-channel controllers located in sub-channels.

The connection between sub-channel controllers and the router controller is shown in Figure 7.2b. The last pipeline stage of each sub-channel is controlled by a sub-channel controller. It reads the ack from the central switch (*AI2CB*), the *eof* output, the ack line driven by the last pipeline stage (ack_0) and the ack from the Clos scheduler (*rt_ra*). In response, it generates the ack line driving the last pipeline stage ($nack_0$) and its own router reset flag *rt_rst*. In the sub-channel controllers shown in Figure 7.2b, the gates coloured in grey are duplicated from the original router controller in Figure 5.5a. They ensure that the operation of each sub-channel complies with the STG of the



(a) Buffer structure



(b) Control logic

Figure 7.2: Input buffer for a virtual circuit

original router controller. The asymmetric C-elements on *AI2CB* and ack_0 wires are the asymmetric C-elements added in the lookahead pipeline as shown in Figure 4.6a. It prohibits the pipeline stage from capturing new data before releasing the old one. The ack line driving the last pipeline stage, $nack_0$, is generated using the same logic as shown in Figure 7.2. It allows the sub-channel to run freely until a tail flit is detected.

All sub-channels are re-synchronized through the router controller, which is also shown in Figure 7.2b. Signal rt_rst_i turns high when sub-channel i has detected the tail flit. The AND gate in the router controller guarantees that the XY router is reset only when every sub-channel has received its share of the tail flit. This AND gate should be replaced with a C-element tree in a strictly QDI design. Considering the long latency

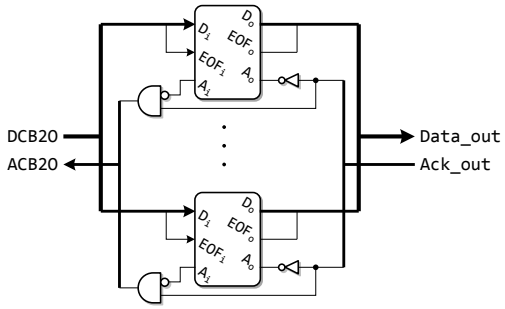


Figure 7.3: Output buffer for a virtual circuit

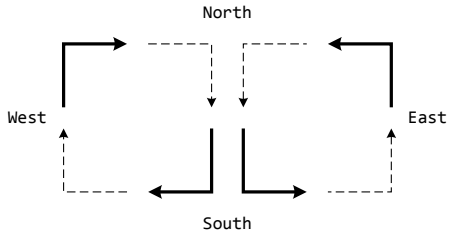


Figure 7.4: The turn model of the XY routing algorithm

of analysing the header flit and reserving a path in the 2-stage Clos switch, it is safe to use the AND gate to reduce area and latency. The router controller generates the *rt_en* signal from *rt_ra*.

The structure of the output buffer of a virtual circuit is shown in Figure 7.3. It contains one pipeline stage which decouples the timing between the central switch and the long inter-router links. Like the input buffers, the buffer stage in output buffers is spatially divided into sub-channels. The AND gate in each sub-channel generates the early evaluated ack line required by the lookahead pipeline. The same circuit has been used in Figure 4.8.

7.1.2 2-stage Clos switch for SDM routers

The 2-stage Clos switch and its scheduler are optimized assuming the XY routing algorithm is used. As described in Section 6.4, each central module in the 2-stage Clos switch is equivalent to a crossbar in a wormhole router. It can be optimized by removing the turns forbidden by the routing algorithm. The turn model of the XY routing algorithm is depicted in Figure 7.4. Only the packets from south and north are allowed to change directions (dimensions) while the packets from west and east travel straight only. The corresponding turns are disabled and depicted in dashed lines.

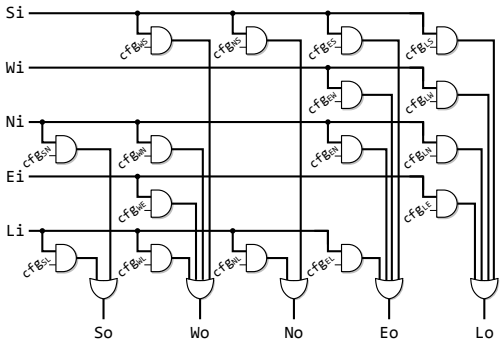


Figure 7.5: An optimized central module

An example of the optimized central module is shown in Figure 7.5. The structure remains the same as a normal crossbar as Figure 5.7 but the AND gates for those disabled turns are removed. Self-turns are also removed as it is normally disallowed to return a packet to its incoming direction. In total, nine out of 25 turns are removed in each central module. The area reduction in the whole 2-stage Clos switch is around 13.8% $\sim$ 25.7% when 2 $\sim$ 8 virtual circuits are implemented in each direction.

Thanks to the 2-stage structure, the 2-stage Clos scheduler shown in Figure 7.6 is much simpler than a 3-stage Clos scheduler (Figure 6.8). All the components for the OM requests *omr*, including the crossbars delivering *omr* (OMRICBs and OMRCCBs) and the OM schedulers, are removed. Incoming requests are the *rt_r* signals from the XY routers in input buffers. The path allocation and release sequence is still controlled by the input request generator (IRG) connected to each input request. As shown in Figure 7.7, two rather than three requests are generated in each IRG: *imr* for the IM dispatcher (IMD) and *cmr* for the CM dispatcher (CMD). IMDs and CMDs dynamically reconfigure IMs and CMs using the asynchronous dispatching algorithm proposed in Chapter 6.

The modified input request generator in an SDM router is demonstrated in Figure 7.7. The incoming request *rt_r* is already coded in one-hot. It is directly forwarded to IMDs and CMDs. The C-elements ensure the STG shown in Figure 7.7b is followed during the path allocation and release process.

The internal structures of IMDs and CMDs are basically the same as those described in Section 6.3.1. Thanks to the removal of disabled turns in CMs, the number of asymmetric C-elements in the request generate matrix in an IMD (Figure 6.10a) is significantly reduced in the IMDs for west and east inputs.

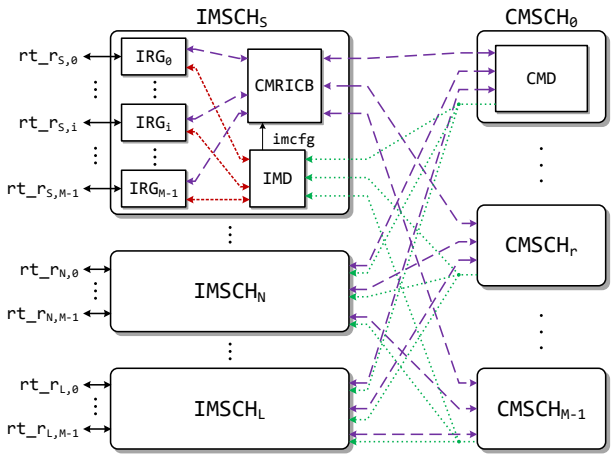
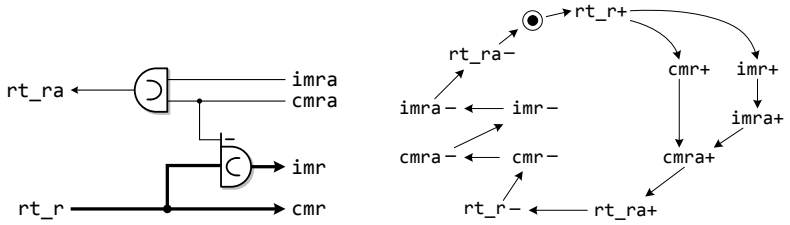


Figure 7.6: Scheduler for the 2-stage Clos switch



(a) Schematic (b) STG

Figure 7.7: The input request generator in an SDM router

7.2 Implementation

7.2.1 Implementation detail

The new SDM router has been implemented into layout using the Faraday 0.13 µm standard cell library. The data width of each direction is 32 bits. Each direction has four virtual circuits, each of which is eight bits wide. The buffer depths of input and output buffers are set to one stage which is the minimum size.

Figure 7.8 demonstrates the floor plan of a router tile. The router is placed in the centre of a 2.5×2.5 mm^2 tile which provides enough space for an embedded processing element [127] in 0.13 µm technologies. The long-range links between routers incur long latency. Two pipeline stages are inserted symmetrically on all long-range links to ensure they are not the throughput bottleneck. The pin order and positions of all top-level I/O ports are manually specified in a way that tiles can be aligned seamlessly into a mesh network. The size of the router placement guide in the centre of Figure 7.8 is proportional to the post-synthesis area of the router. The area ratio of the post-synthesis

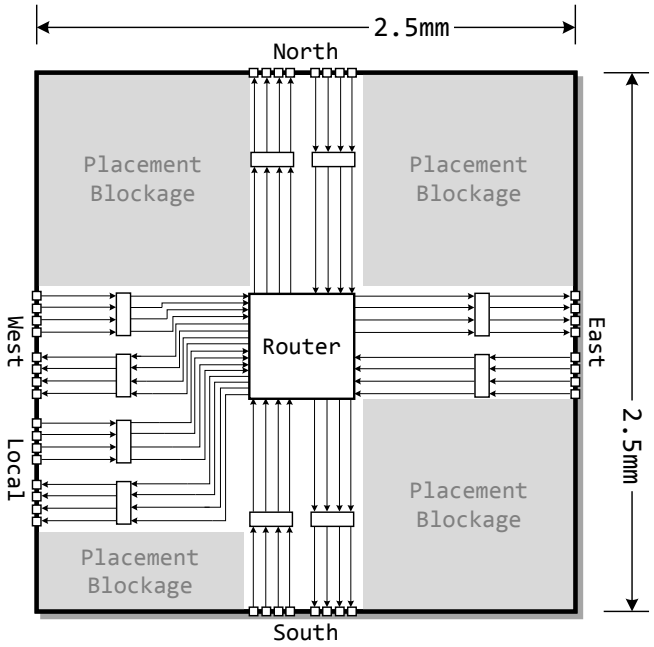


Figure 7.8: Floor plan of a router tile

router area to the router placement guide size is set to 40% leaving 60% extra space to avoid wire congestion and performance degradation. This 40% area ratio is relatively large compared to the ratio of synchronous circuits. It is believed that asynchronous circuits are likely to generate wire congestion due to the feedback wires in C-elements and handshake circuits. The area ratio of the ANOC router implementation is 32% [7], which is even smaller than the 40% used in this research. Several placement blockages are placed inside the tile to direct the cell placement.

To demonstrate the area overhead and speed improvement of using the techniques developed in this research (channel slicing technique, the lookahead pipeline, the SDM flow control method and the 2-stage Clos switch), the new SDM router (SDM-Clos ChSlice+LH) has been compared against several router implementations, including a wormhole router using synchronized pipelines (WH), a wormhole router using channel slicing and lookahead pipelines (WH ChSlice+LH), an SDM router using synchronized pipelines (SDM), an SDM router using channel slicing and lookahead pipelines (SDM ChSlice+LH) and a VC router (VC). All routers are configured with the same data width, depth of buffers and number of virtual circuits/VCs. The VC router is a reproduction of the VC router in the QoS NoC [45]. It has been modified to support best-effort traffic. The detailed modifications are presented in Appendix B. The floor plans of these routers comply with the plan shown in Figure 7.8. The size of the router

Table 7.1: Area consumption

	Input Buffer	Output Buffer	Central Switch	Switch Allocators	Router Overall	Placement Guide
VC	72,778	11,376	30,815	55,393	222,227	525,625
WH	8,870	8,403	9,881	909	28,451	62,500
WH ChSlice+LH	17,147	11,034	17,754	909	48,170	93,025
SDM	18,400	9,462	50,941	83,382	163,620	366,025
SDM ChSlice+LH	23,356	9,016	69,701	83,485	187,872	403,225
SDM-Clos ChSlice+LH	19,995	8,893	33,417	19,884	84,054	189,225

(unit: μm^2)

placement guide and the pin positions are always regenerated for different configurations.

7.2.2 Area consumption

The area breakdown of different router implementations is revealed in Table 7.1. All values are obtained from the post-layout area report which does not include the area of the pipelines inserted between routers (whose area is revealed in Section 8.1.2 and Appendix C). Route congestion has affected the area. The size of the router placement guide is also shown in the last column of Table 7.1 to demonstrate the overall die size required to avoid route congestion.

The wormhole router using synchronized pipelines consumes the smallest area. Using channel slicing and lookahead pipelines enlarges the size of input buffers and the switch. As explained in Section 4.4.2, extra area is consumed by the sub-channel controllers in input buffers and the increased wire count in the switch. However, the area increase in output buffers is not expected as it is inconsistent with the claims in Section 4.4.2. Channel slicing and lookahead pipelines significantly reduce the period. It is found that the pipeline stages inserted on the long-range links between routers have been greatly optimized in the place and route process because they have longer periods than the critical cycles inside routers. This optimization leads to large driving cells in output and input buffers which cause the extra area. Inserting more pipeline stages on the long-range links can alleviate this effect.

The SDM router using synchronized pipelines introduces extra area overhead in input buffers, the central switch and the switch allocator. As analysed in Section 5.4.1, every virtual circuit is a fully functional input buffer with its own XY router along with the router controller. The size of the crossbar increases proportionally to the number

of virtual circuits. The switch allocator consumes more area than the wormhole router due to the larger arbitration scheme. Using channel slicing and lookahead pipelines in an SDM router introduces extra area overhead in the same way as they do in a wormhole router.

As shown in Table 7.1, using the 2-stage Clos switch in the SDM router significantly reduces the area of the switch and the switch allocator. The total area reduction is around $100,000\ \mu\mathrm{m}^2$, which is 53.2% of the overall area.

It is shown clearly that the VC router consumes the largest area overhead in all router structures. The conceptual area estimation model for VC routers in Section 5.4.2 actually under-estimates the area. Through a detailed analysis of the internal structure of the VC router in the QoS NoC [47, 45], the large area overhead of this VC router is because of following reasons:

- As shown in the area model in Section 5.4.2, VC duplicates input buffers, enlarges the central switch as SDM routers do, and introduces a VC allocator besides the switch allocator.
- Extra latches are used to store the VC number and the flit type for each flit.
- Separated pipelines and control logic are required to handle credits (Section 8.5.1 in [45]).
- C-elements are used in the crossbar because the configuration bit is released before the withdrawal of the ack line (Figure 5.2 in Section 5.1).
- Several pipeline stages have been inserted in input buffers (Figure 8.7 in [45]) and output buffers (Figure 8.16 in [45]) to reduce switch request cycle time and decouple it from data transmission.

7.2.3 Router speed

Table 7.2 demonstrates the speed performance of different router implementations. The delays are obtained from post-layout simulations. Wire resistance and capacitance have been properly extracted and back-annotated into simulations.

The period of the wormhole router using synchronized pipelines is $3.68\ \mathrm{ns}$, which is equivalent to $272\ \mathrm{MHz}$. The XY router needs $0.61\ \mathrm{ns}$ to analyse a header flit and the switch allocator reconfigures a path in $1.06\ \mathrm{ns}$ if no contention occurs. The latency for a data flit traversing the router is around $0.96\ \mathrm{ns}$ in average.

Table 7.2: Router latency

	Cycle Period ns	Equivalent Frequency MHz	Router Latency ns	XY Router ns	Switch Allocators ns
VC	5.29	189	5.14	1.96	1.64
WH	3.68	272	0.96	0.61	1.06
WH ChSlice+LH	2.24	446	0.92	0.64	1.24
SDM	3.27	306	1.29	0.68	2.09
SDM ChSlice+LH	2.86	350	1.18	0.68	2.01
SDM-Clos ChSlice+LH	2.67	375	1.29	0.63	2.13

Using channel slicing and lookahead pipelines in wormhole routers reduces the period to 2.24 ns, which is equivalent to 446 MHz. Channel slicing and lookahead pipelines increase the equivalent frequency by 63.9%. It should be noticed that in this router configuration, throughput is constrained by the long-range links rather than the critical cycle inside the router. Inserting more pipeline stages on the long-range links can further reduce the period.

It is shown that the SDM router using synchronized pipelines achieves shorter period than the wormhole router using synchronized pipelines. The latency reduction obtained from the reduced C-element trees inside completion detection circuits overcomes the extra latency introduced by the large central switch. The period is 3.27 ns, which is equivalent to 306 MHz. Other than the reduced period, SDM also improves throughput by alleviating the HOL problem. The throughput performance of SDM routers will be presented in the next chapter. Consistent with the latency models in Section 5.4.1, the enlarged central switch leads to longer router latency and allocation latency compared with wormhole routers.

Using channel slicing and lookahead pipelines in SDM routers has the same effect as in wormhole routers. Period and router latency are reduced accordingly. The equivalent frequency of the SDM router using channel slicing and lookahead pipelines is 350 MHz, which is 14.4% higher than the SDM router using synchronized pipelines.

It is already known that using the 2-stage Clos switch reduces the area of an SDM router. It is believed that period and router latency should be prolonged as another stage of switches is inserted in the critical cycle. As shown in Table 7.2, the router latency is prolonged as expected but the period is actually reduced. This reduction can be caused by two reasons: first, the size of the central switch is reduced by 50% (as revealed in Table 7.1); correspondingly the average distance between two pipeline

stages is reduced. Second, using the 2-stage Clos switch reduces the fan-out of data wires as well as the load of driving cells. To sum up, the period reduction is marginal and the speed degradation introduced by Clos switches is moderate.

The VC router presents the worst speed performance. Its period is the longest due to its extra switch reconfiguration overhead. The long router latency is caused by its deep pipelines in the router. Every VC buffer is implemented with the minimum depth of two pipeline stages. One extra pipeline stage is required in each input buffer to decouple the switch allocation from data transmission [45]. The total number of pipeline stages is four, two times of the depth of the minimal wormhole routers or SDM routers. The XY router in VC routers runs in parallel with VC buffers. The long XY router latency is due to the extra pipeline delay but this does not compromise throughput. Detailed performance comparisons will be presented in the next chapter.

7.3 Summary

This chapter has presented a novel asynchronous spatial division multiplexing router utilizing all the techniques proposed in Part II. Several asynchronous routers using different flow control methods and various pipeline styles have been implemented into layout. All routers are placed in the centre of a 2.5×2.5 mm tile with carefully specified I/O mapping for network integration. The area breakdown and latencies of various router implementations are revealed with explanations of the area and latency overhead of different router structures.

The evaluation in this chapter reveals the physical performance of different router implementations without injecting them with any traffic. In the next chapter, routers are evaluated with synthetic traffic models to demonstrate their capability of delivering frames. They will be evaluated in the environments of a single router and networks.

Chapter 8

Performance Evaluation

This chapter will evaluate the performance of different router structures with synthetic traffic models. The evaluation will be processed in two steps: single router evaluation and network evaluation. In the single router evaluation, routers are examined individually with random uniform frame streams loaded to all input ports. This evaluation will demonstrate the theoretical saturation throughput of all router structures and the energy consumed for a frame going through a single router. The network evaluation will examine the network throughput and the power consumption of different routers in an 8×8 mesh network with random uniform traffic. A 'real world' example of a 4×3 mesh network simulating an MPEG-4 MPSoC system is also adopted using the traffic data from [10, 65] as this is available.

8.1 Single router evaluation

8.1.1 Test environment

The design under test in the single router evaluation is one of the asynchronous routers implemented in Section 7.2. A total of five frame generators are connected to the five input ports. Every output port is linked with a frame sink. Frames are generated randomly by frame generators and consumed by frame sinks. The frame flow in all frame generators and sinks is recorded in a global bookkeeping database.

A frame generator produces a sequence of frames complying with a predefined Poisson process. In the single router evaluation, frames head to all output directions according to the frame direction distribution defined in Table 8.1. As routers use the XY routing algorithm, some directions are correspondingly prohibited in the table. The

Table 8.1: Frame direction distribution

	load	to south	to west	to north	to east	to local
from south	1.25ρ	0	0.25ρ	0.5ρ	0.25ρ	0.25ρ
from west	0.5ρ	0	0	0	0.25ρ	0.25ρ
from north	1.25ρ	0.5ρ	0.25ρ	0	0.25ρ	0.25ρ
from east	0.5ρ	0	0.25ρ	0	0	0.25ρ
from local	1.5ρ	0.5ρ	0.25ρ	0.5ρ	0.25ρ	0

distribution is defined in a way that all output ports are loaded with an equal amount of traffic coming evenly from all possible input directions. The payload size is fixed to 64 bytes for high throughput [115]. Every newly generated frame is updated to the central bookkeeping database with information including the current time stamp, the target direction, payload size and a 64-bit unique key for trace tracking. A FIFO is inserted between every pair of frame generator and input port. The size of this FIFO is set to 500 frames, approximating an infinite input buffer.

A frame sink receives frames coming from a certain output direction. The integrity of every incoming frame is verified by comparing the unique key regenerated from the frame against the records in the global bookkeeping database. The frame transmission latency and the overall throughput are updated if a match is found in the database, otherwise an error message is produced for debug purpose and the simulation stops.

The global bookkeeping database records all active frames in a hash table using the 64-bit unique key as the record identifier. Information about the delivered frames, such as frame transmission latency, path reservation latency, payload transmission latency, overall throughput and delay histogram, are accumulated and reported to a log file according to different test configurations. In the single router evaluation, frame transmission latency and overall throughput are traced and reported.

Frame generators, frame sinks and the global bookkeeping database are written in parameterized SystemC models which are used to test all router structures in this thesis. The SystemC test bench is co-simulated with the back-annotated post-layout netlists (Verilog-HDL) generated in Chapter 7 using Cadence NC-Simulator. Detailed signal switch activities are collected during the simulation and the power consumption of all routers are obtained by the post-simulation analysis in Synopsys PrimeTime PX.

Table 8.2: Single router performance

	Tile Area μm^2	Router Area μm^2	Min Frame Latency ns	Saturation Throughput MByte/s	Tile Power mW	Router Power mW
VC	243,940	222,277	93.7	579.8	21.5	12.4
WH	47,167	28,451	64.0	636.6	14.9	6.29
WH ChSlice+LH	71,873	48,170	40.3	995.8	37.3	17.6
SDM	182,644	163,620	223.5	877.2	25.0	13.3
SDM ChSlice+LH	208,300	187,872	179.8	1,032.5	39.8	22.6
SDM-Clos ChSlice+LH	104,987	84,054	187.4	964.3	32.5	15.8

8.1.2 Performance

The preliminary results of all router structures are provided in Table 8.2. The data width of all routers is 32 bits and four virtual circuits or virtual channels are implemented in SDM or VC routers. The meanings of the columns in Table 8.2 are as follows:

- *Tile area*: the post-layout area of a tile, including the router and the pipeline stages inserted between routers.
- *Router area*: the post-layout area of a router (the same values were shown in Table 7.1 as *router overall*).
- *Minimum frame latency*: the minimum latency required by a router to deliver a frame when no contention occurs.
- *Saturation throughput*: the maximum data rate provided by a router when all its input ports are overladen with uniform traffic.
- *Tile power*: the total power consumption of a tile, including the router and the pipeline stages inserted between routers, when the router is overloaded.
- *Router power*: the power consumption of a router when it is overloaded.

As explained in Section 7.2, channel slicing, the lookahead pipeline style and the spatial division multiplexing flow control method increase area. Using the 2-stage Clos switch in SDM routers reduces area overhead significantly. All routers are smaller than the VC router with four virtual channels.

The minimum frame latencies of wormhole routers (WH and WH ChSlice+LH) and VC routers (VC) are approximately proportional to the period shown in Table 7.2.

It can be explained by Equation 5.1. The frame latency consists of two parts: the time consumed to reserve a path and the time consumed to transmit data. When the network is free of contention, the path reservation latency is proportional to $t_R \cdot h$, where t_R is the router latency and h is the number of hops between frame sender and receiver. Since the hop count is only one in the single router evaluation, the data transmission latency, which is proportional to router period, dominates the frame latency. The data width of a virtual circuit is a quarter of the total data width in the SDM routers (SDM and SDM ChSlice+LH) with four virtual circuits. The minimum frame latency is accordingly prolonged to approximately four times the minimum frame latency of the wormhole router (WH and WH ChSlice+LH).

It is shown in the throughput performance that channel slicing, the lookahead pipeline style and SDM enlarge the saturation throughput of a single router. The SDM router using channel slicing and lookahead pipelines (SDM ChSlice+LH) provides the best saturation throughput of $1032.5\ \mathrm{MByte/s}$. The 2-stage Clos switch compromises the saturation throughput by 6.6% because the rearrangeable non-blocking Clos switch is blocking when established paths cannot be reallocated.

When routers are connected into a network, router throughput decreases due to the contention caused by the communication pattern and the head-of-line problem. The practical throughput of a router in a network can never reach the saturation throughput shown in Table 8.2. The capability of different flow control methods in handling the head-of-line problem will be demonstrated in the network evaluation (the next section). The VC router (VC) shows the worst router throughput due to its longest period. It is only 56% of the saturation throughput provided by the SDM router using channel slicing and lookahead pipelines (SDM ChSlice+LH).

The power consumption of an asynchronous router is proportional to its throughput and it is also related to its internal structure. It is known that using channel slicing, lookahead pipelines and SDM leads to extra power consumption. These techniques increase the saturation throughput which, in turn, consumes extra power. On the other hand, they may consume more power due to their router structures. Figure 8.1 reveals the energy efficiency of different routers. The efficiency is calculated as the saturation throughput divided by the tile power consumption. It can be understood as the average amount of data delivered by a unit of energy ($\mathrm{MByte/mJ}$). As shown in Figure 8.1, the wormhole router using synchronized pipelines (WH) provides the highest energy efficiency thanks to its simple router structure. All techniques that improve throughput compromise energy efficiency. Compared with SDM (SDM), using channel slicing

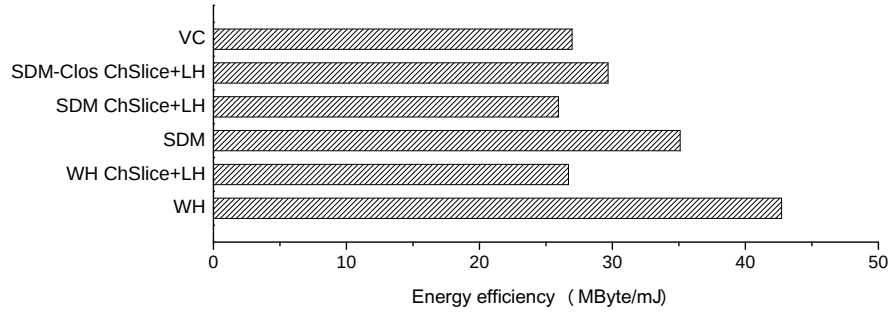
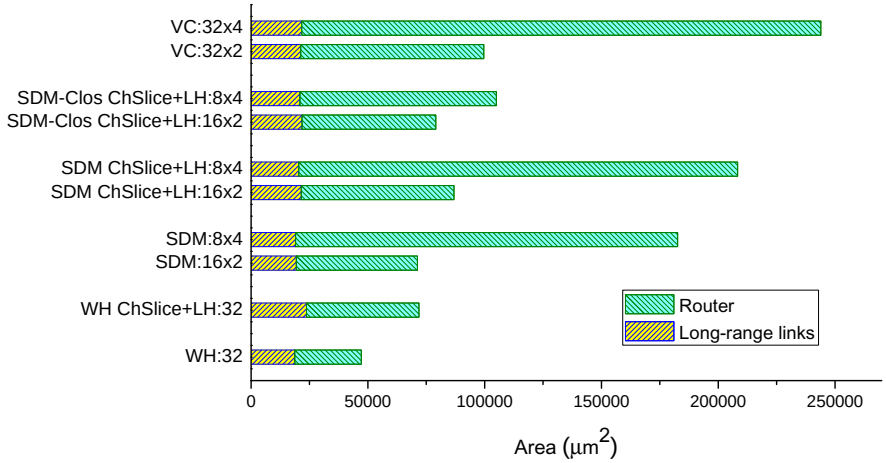


Figure 8.1: Energy efficiency of a single router

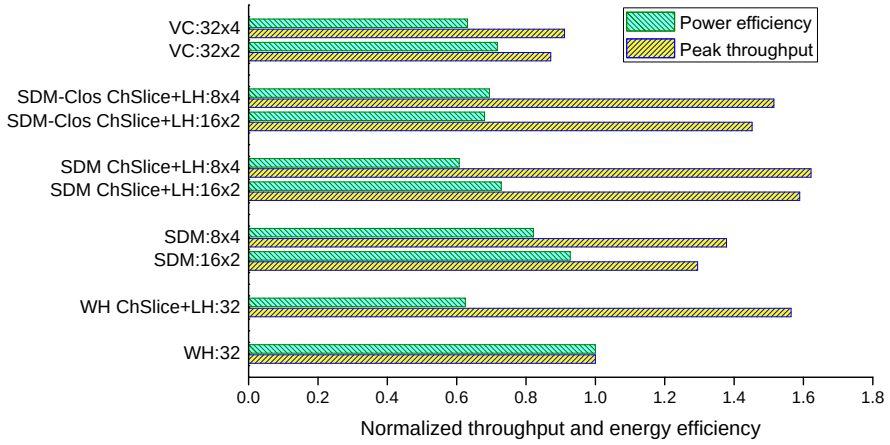
and lookahead pipelines (WH ChSlice+LH) consumes more power. The 2-stage Clos switch is the only technique that improves energy efficiency instead of compromising it. It is also shown in Figure 8.1, the virtual channel flow control method compromises the energy efficiency. VC routers have deeper pipelines than other router structures (two more pipeline stages in this thesis), which leads to more transitions for a router to deliver a flit. The energy efficiency of the VC router is similar to the efficiency of the wormhole router using channel slicing and lookahead pipelines.

The remainder of this section will reveal the performance change of using different numbers of virtual circuits or VCs and different data widths. To identify the data width and the number of virtual circuits or VCs in each test case, an extra label is added at the end of the name of each case. For a wormhole router using W-bit ports, "$:W$" is added to denote the data width W. When virtual circuits or VCs are utilized, "$:W$xM" is used to denote that M virtual circuits or VCs are implemented and each virtual circuit or VC is W bits wide. Notice that the total data width of an SDM router is $W \cdot M$ while it is W for a VC router because the ports in SDM routers are spatially divided to virtual circuits.

Figure 8.2 reveals the single router performance using various numbers of virtual circuits or VCs in SDM routers or VC routers. Figure 8.2a demonstrates the area overhead of adding virtual circuits or VCs. The area overhead of the long-range links has no significant difference in all router implementations except the wormhole router using channel slicing and lookahead pipelines (WH ChSlice+LH:32). This router has the shortest period which makes the long-range links the critical cycle. The pipeline stages on long-range links are therefore optimized in the place and routing procedure leading to the extra area overhead. In nearly all router implementations, the router area is doubled when the number of virtual circuits or VCs is doubled. The only exception is the SDM router using the 2-stage Clos switch (SDM-Clos ChSlice+LH). The area



(a) Area consumption



(b) Throughput and power

Figure 8.2: Router performance with various number of virtual circuits/VCs

reduction compared with the SDM router using crossbars is 55.3% in the four virtual circuits case (SDM-Clos ChSlice+LH:8x4) while it is only 12.4% in the two virtual circuits case (SDM-Clos ChSlice+LH:16x2). It is more beneficial to use 2-stage Clos switches in SDM routers when the number of virtual circuits is larger than four.

Taking the performance of the wormhole router using synchronized pipelines (WH:32) as the baseline case, the normalized throughput[1] and energy efficiency performance is depicted in Figure 8.2b. It should be noticed that the throughput increase of adding extra virtual circuits or VCs in the single router evaluation is marginal because no head-of-line problem is generated. The small throughput increase is because the contention rate of the central switch is slightly reduced with extra virtual circuits or

[1]The calculation of throughput efficiency is defined in Equation 5.32 where it is called gain. Instead of using router area as the denominator in Equation 5.32, tile area is used here.

VCs[2]. Adding extra virtual circuits or VCs compromises energy efficiency. However, using the 2-stage Clos switch can alleviate this effect to some extent.

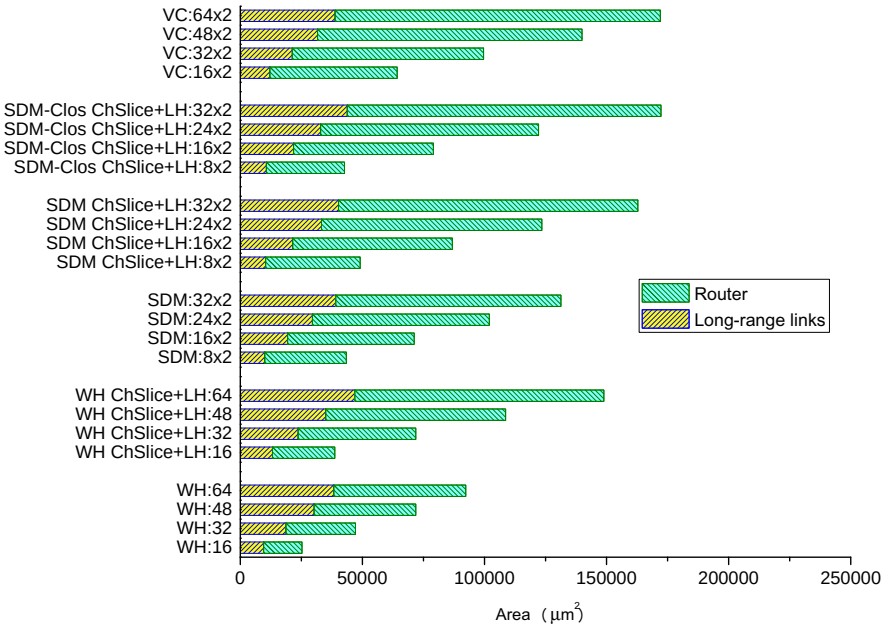
Increasing data width is an effective way of providing extra throughput. As analysed in Section 5.4.3, increasing data width is more area efficient than increasing the depth of buffers. Figure 8.3 demonstrates the single router performance with various data widths. Two virtual circuits or VCs are implemented in SDM routers or VC routers. The overall data width increases from 16 bits to 64 bits.

In the area consumption shown in Figure 8.3a, the area of long-range links is proportional to data width. Tile area increases linearly. Since the number of virtual circuits is only two, the area reduction of using 2-stage Clos switches is not significant. VC routers always consume the largest area until the data width increases to 64 bits. In this case, the VC router still consumes the largest area but the SDM router using the 2-stage Clos switch has the largest long-range links, which make it the most area consuming structure.

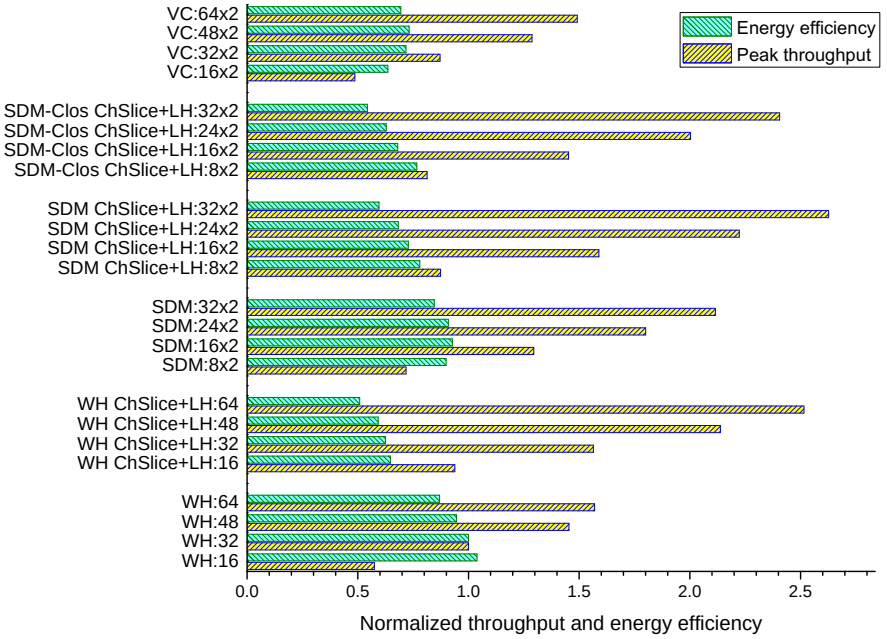
The throughput increase is significant in all router structures but the increase is not linear with data width. The routers using synchronized pipelines show worse throughput increase than the routers using channel slicing and lookahead pipelines. Specifically, increasing the data width of the wormhole (WH) router from 48 bits to 64 bits brings $73.8\ \mathrm{MByte/s}$ extra throughput. It is only 25.5% of the extra throughput incurred by increasing the data width from 32 bits to 48 bits. The same ratios are 65.8%, 62.4%, 63.6%, 73.1% and 49.2% for WH ChSlice+LH, SDM, SDM ChSlice+LH, SDM-Clos ChSlice+LH and VC respectively. This degraded increase is because of two reasons: (1) The routers using synchronized pipelines have deep C-element trees in their completion detection circuits which prolong the period. (2) The payload size is fixed to 64 bytes. Increasing data width leads to frequent switch reallocation which causes throughput loss.

The energy efficiency of nearly all router structures drops with data width. The SDM routers using synchronized pipelines (SDM) and VC routers are the only exceptions. They show a small efficiency increase from the 16-bit cases to the 32-bit cases.

[2]It can be proved that in a packet drop SDM router, the contention rate is reduced by 6.7% by increasing the number of virtual circuits from two to four. In routers using input queued switches as presented in this thesis, the blocked requests request again on the next available occasion. The contention reduction is less than the rate in packet drop routers. The effective contention reduction can be calculated using the G/M/m queueing theory [67] depending on the traffic flow and the switch arbitration algorithm.



(a) Area consumption



(b) Throughput and power

Figure 8.3: Router performance with various data widths

In these test cases, using the 2-stage Clos switch in the SDM routers with only two virtual circuits (SDM-Clos ChSlice+LH:Wx2) shows no area reduction or improvement in energy efficiency. Using a rearrangeable Clos switch when the switch radix is small leads to throughput degradation with no significant area reduction. Figure 8.4

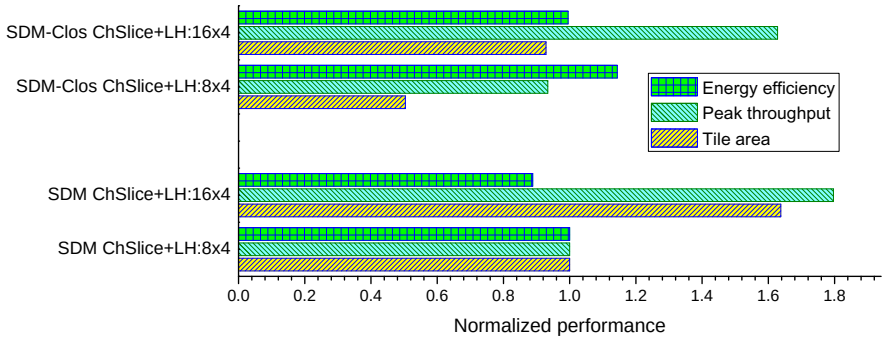


Figure 8.4: Performance of SDM routers with four virtual circuits

reveals the performance of using the 2-stage Clos switch in the SDM routers with four virtual circuits, where the radix of the central switch is enlarged to 20×20. All results are normalized using the performance of the SDM ChSlice+LH:8x4 router as the baseline. It is shown that using the 2-stage Clos switch in these cases significantly reduces tile area and improves energy efficiency but causes a small throughput degradation.

8.2 Network performance

In the network evaluation, tiles are connected into mesh networks and their performance in these networks is examined. Two mesh networks will be tested: an 8×8 mesh network is produced to test the performance with random uniform traffic patterns and a 4×3 network is built to reveal the performance of implementing an MPEG-4 system using asynchronous on-chip networks.

8.2.1 Mesh network with uniform traffic

One of the objectives in network evaluation is to examine the performance in large scale networks. Since all the simulations in this chapter use post-layout netlists, the simulation speed is slow and the simulations are memory consuming. For the largest router implementation, the 32-bit VC router with four VCs (VC:32x4), the 8×8 network simulation needs more than 2 GB memory for compilation and nearly 1.5 GB memory for simulation. The network scale is constrained by the available memory space. On the other hand, networks up to 16×16 can be easily simulated using the behavioural models presented in Chapter 5.

The frame generator and the frame sink used in Section 8.1.2 are combined into a processing element (PE). The local port of every router is connected to a PE and other

Table 8.3: Network performance

	Tile Area μm^2	Min Frame Latency ns	Saturation Throughput MByte/Node/s	Tile Power mW	Energy Efficiency MByte/mJ
VC	243,940	155.3	262.8	17.0	15.46
WH	47,167	81.0	207.8	9.1	22.74
WH ChSlice+LH	71,873	58.4	320.1	22.2	14.42
SDM	182,644	252.0	366.9	19.7	18.63
SDM ChSlice+LH	208,300	205.4	446.8	31.5	14.18
SDM-Clos ChSlice+LH	104,987	211.3	408.9	24.5	16.69

ports are connected to adjacent routers. All the frames generated and consumed by PEs are recorded by a global bookkeeping database in the same way as in Section 8.1.2. The switching activities of router(3,3) is recorded to represent the power of the whole network.

Table 8.3 reveals the network performance of different router implementations. Note that the *tile power* and the *energy efficiency* are obtained using the power consumption of router(3,3) instead of the whole network.

The minimum frame latency in the network evaluation is longer than that in the single router evaluation due to the extra time consumed for reserving a path between the sender and the receiver. The latency values of all routers show extra delay around 15 to 30 ns except for the VC router case which needs extra 61.6 ns. The VC router is configured with only one full buffer stage in every VC buffer. It thus suffers from the credit loop latency which is 6.44 ns, 1.22 times of the period (see the description of Equation 5.31).

SDM and VC routers show better throughput increase compared with the increase in the single router evaluation. The head-of-line (HOL) problem in the network evaluation compromises throughput. Virtual circuits and VCs in SDM and VC routers alleviate the HOL problem. In detail, the SDM router using channel slicing and lookahead pipelines (SDM ChSlice+LH) provides the best throughput of 446.8 MByte/Node/s, which is 2.15 times of the throughput of the wormhole router using synchronized pipelines (WH).

Energy efficiency has no significant change from the single router evaluation. The energy efficiency of all routers drops because a frame needs to traverse multiple routers before reaching the target node. The wormhole router using synchronized pipelines (WH) still shows the best energy efficiency and the SDM router using channel slicing

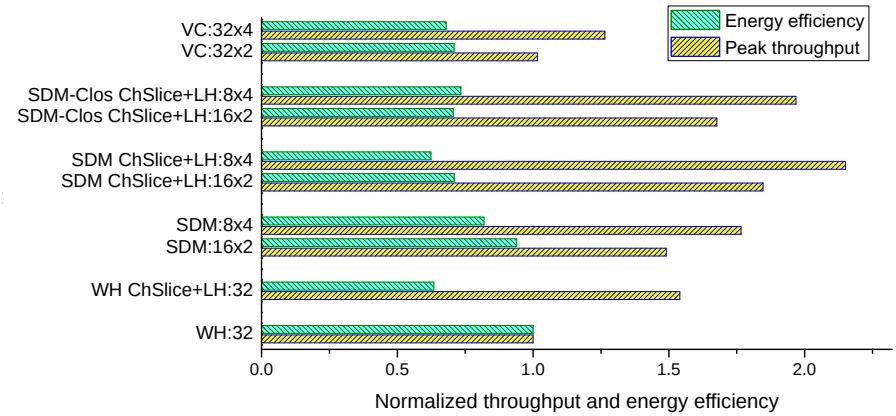


Figure 8.5: Network performance with various number of virtual circuits/VCs

is the worst. Using 2-stage Clos switches in SDM routers improves energy efficiency in the same way as in the single router evaluation.

Figure 8.5 shows the network performance with various numbers of virtual circuits or VCs. Router area is not presented as it can be found in Figure 8.2a. Energy efficiency shows no significant difference compared with the efficiency in the single router evaluation as shown in Figure 8.2b but the throughput results of SDM and VC routers are much improved due to their capability of alleviating the HOL problem. It is shown that increasing the number of virtual circuits or VCs achieves significant throughput increase in the network evaluation although the throughput increase in the single router evaluation is marginal.

Similar throughput improvement is found in the network performance of using various data widths as shown in Figure 8.6. The network throughput of the SDM ChSlice+LH:32x2 router is $620.37\ \mathrm{MByte/Node/s}$, which is 2.99 times of the network throughput of the WH:32 router. In the single router performance shown in Figure 8.3b, this figure is only 2.63. The extra 0.36 improvement is due to the HOL alleviation. Similarly, the energy efficiency does not change much. The 2-stage Clos switch cannot save power or significantly reduce area in the SDM routers with fewer than four virtual circuits.

8.2.2 An MPEG-4 system

A 4×3 mesh network has been built to test the network performance with a traffic pattern extracted from an MPEG-4 system [10, 65]. The task mapping and the bandwidth requirement between nodes are illustrated in Figure 8.7a [65]. The values between nodes are the bidirectional bandwidth requirements in unit of $\mathrm{MByte/s}$. SDRAM

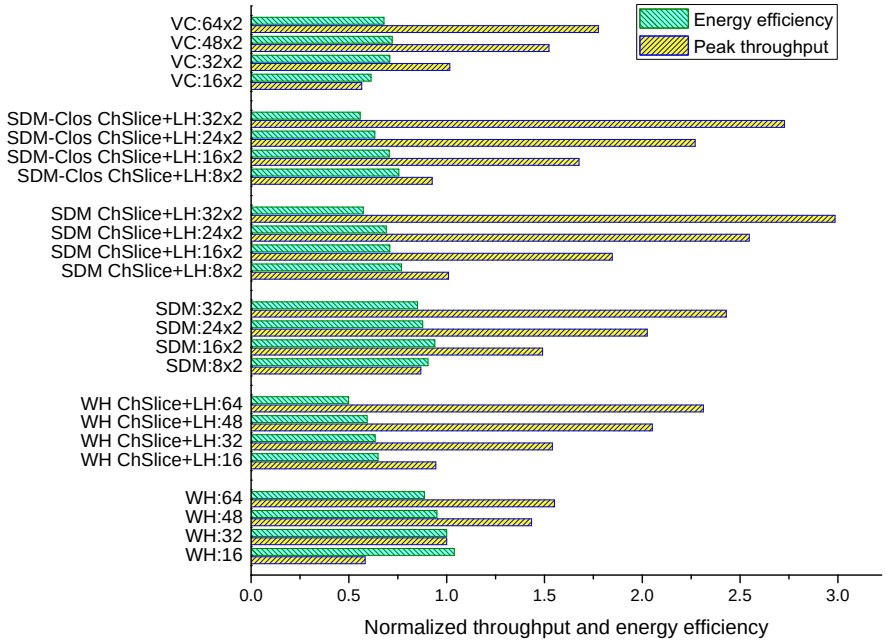
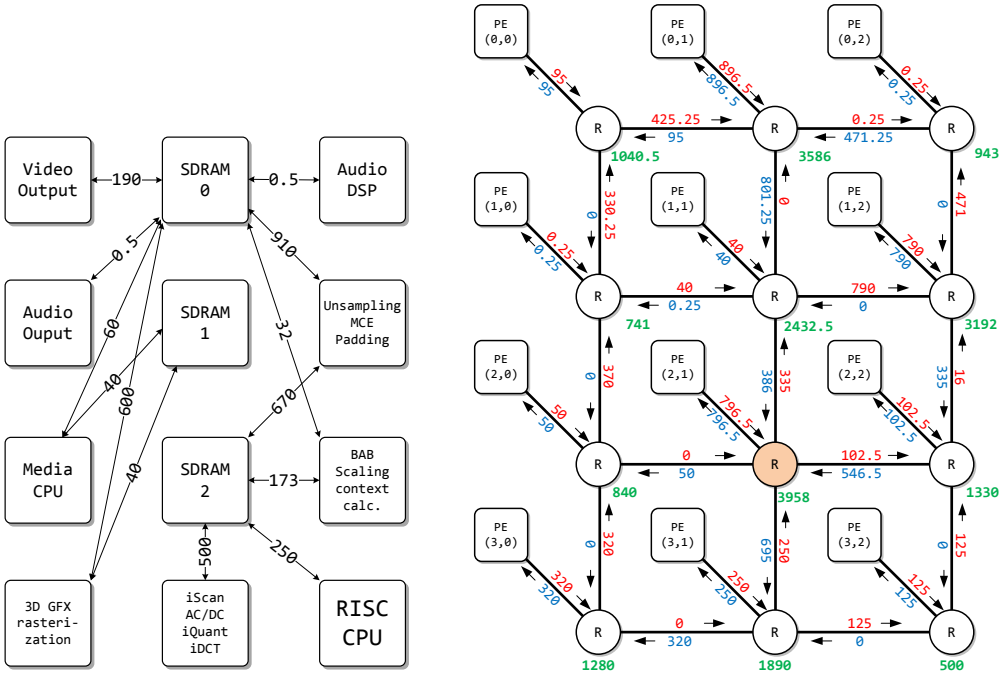


Figure 8.6: Network performance with various data widths

0 and SDRAM 2 are the two hot spots of the NoC. The bandwidth requirements of these two nodes are $1.793\ \mathrm{GByte/s}$ and $1.593\ \mathrm{GByte/s}$ respectively. Assuming the bandwidth requirements are equally divided in forward and backward directions [65], the detailed bandwidth requirements of all links and routers using the XY routing algorithm is depicted in Figure 8.7b. The total bandwidth requirement of a router is labelled beside it in green. The router connected to PE(2,1) is the busiest one. 57.1% of the network traffic ($3.446\ \mathrm{GByte/s}$) goes through it.

In the simulation, the expected data generated by each processing element complies with the task mapping shown in Figure 8.7a. The target node addresses of frames are randomly chosen from the possible addresses in Figure 8.7a and the address distribution complies with the bandwidth ratio. The payload size of each frame varies from eight to 64 bytes. The communication with a small bandwidth requirement uses short frames and the communications with requirements more than $64\ \mathrm{MByte/s}$ are served using with the largest payload size of 64 bytes.

Figure 8.8 reveals the overall throughput of different router structures. As identified by a dash line in the figure, the overall throughput requirement of the MPEG-4 system is $3,466\ \mathrm{GByte/s}$. It is shown that the wormhole router using synchronized pipelines (WH) can satisfy the requirement when the data width is equal to 48 bits or larger. The data width can be reduced to 32 bits if an SDM router (SDM:16x2) or channel slicing (Wh ChSlice+LH:32) is utilized. The VC router can satisfy the throughput



(a) Task mapping (b) Detailed bandwidth requirement

Figure 8.7: Task mapping and bandwidth requirement of MPEG-4

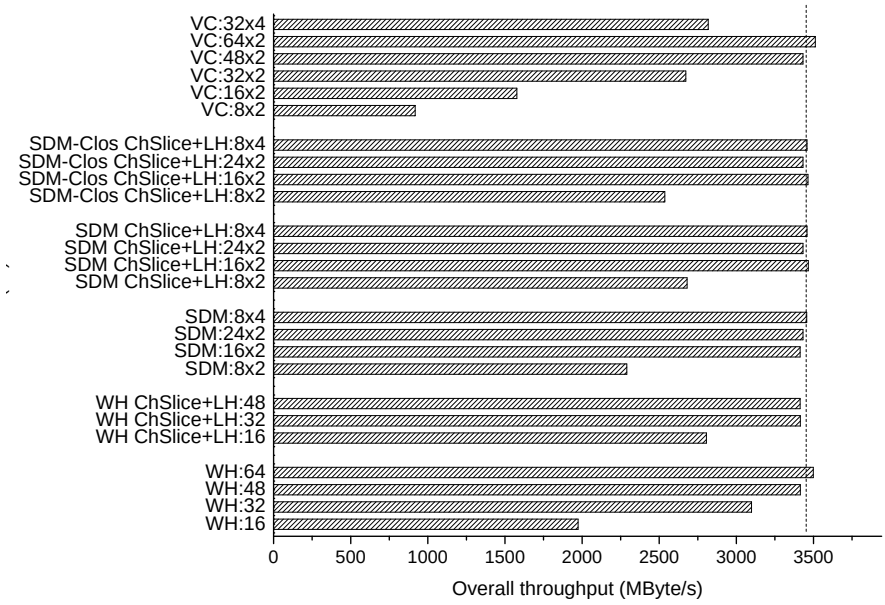
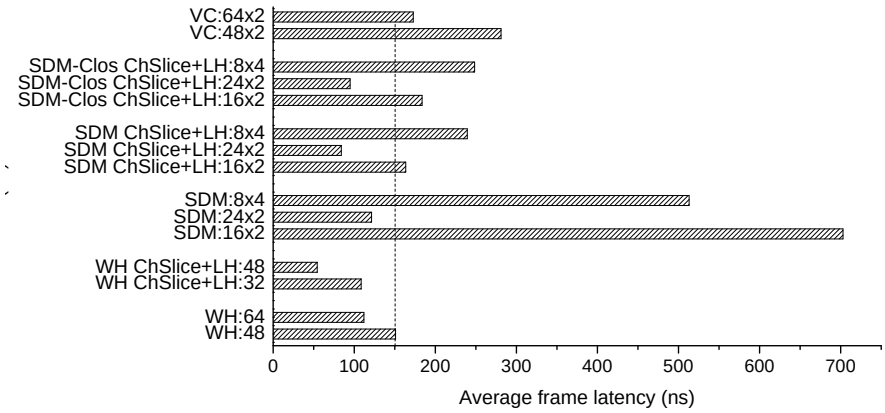


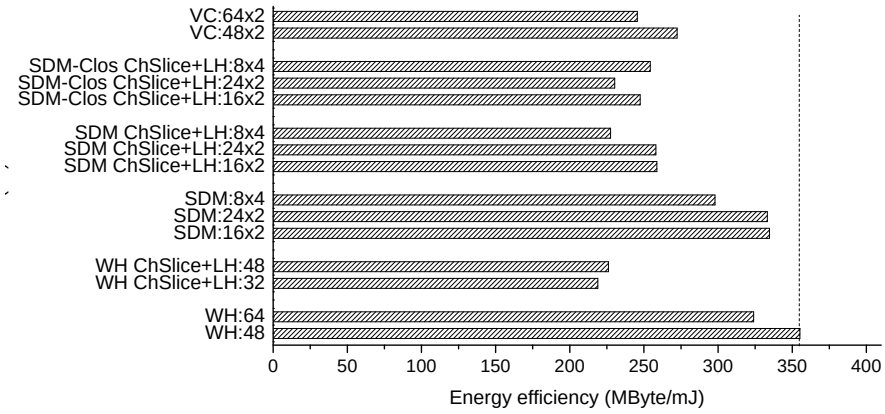
Figure 8.8: Overall throughput of the MPEG-4 NoCs

requirement only with the 48-bit data width (VC:48x2).

Considering only those routers satisfying the throughput requirements, Figure 8.9 demonstrates the average frame latency and the energy efficiency. SDM routers suffer



(a) Latency



(b) Power

Figure 8.9: Latency and power consumption of the MPEG-4 NoCs

from long frame latency as frames are transmitted in a serialized manner. Nevertheless, SDM routers still demonstrate shorter frame latency than the 48-bit wormhole router (WH:48) when channel slicing and lookahead pipelines are used (SDM ChSlice+LH:24x2), or when its data width is equal to 48 bits or larger (SDM:24x2). The wormhole routers using channel slicing and lookahead pipelines (WH ChSlice+LH) always outperform the wormhole routers using synchronized pipelines (WH) in frame latency.

It is known that most techniques proposed in this thesis compromise energy efficiency. The energy efficiency of all router structures is revealed in Figure 8.9b. It is shown that the 48-bit wormhole router using synchronized pipelines (WH:48) provides the best energy efficiency of $355\ \mathrm{MByte/mJ}$. It is also shown that the 32-bit and 48-bit SDM routers using synchronized pipelines (SDM:16x2 and SDM:24x2) outperform the 64-bit wormhole router (WH:64) in this particular traffic pattern because increasing data width compromises energy efficiency.

The throughput requirement of the MPEG-4 system is comparatively small. The basic wormhole router using synchronized pipelines satisfies the requirement when the data width is more than 48 bits. It is also shown that SDM and channel slicing can be used to reduce the data width, and the average frame latency. When extra bandwidth is required, SDM routers can be used for both shorter frame latency and higher energy efficiency than wormhole routers.

8.3 Summary

The performance of different router structures has been compared in the single router evaluation and in the network evaluation. The single router evaluation reveals: SDM, channel slicing and the lookahead pipeline style introduce area overhead but it is smaller than the overhead of using virtual channels. The 2-stage Clos switch significantly reduces the area overhead of SDM routers when the number of virtual circuits is four or larger. Wormhole routers have the shortest frame latency thanks to their simple router structure. SDM routers introduce latency overhead as frames are serialized in transmission. The throughput in the single router evaluation provides the maximum throughput of a router without considering the HOL problem. It is shown that channel slicing and lookahead pipelines improve throughput significantly. However, these techniques compromise the energy efficiency. Compared with them, SDM is more energy efficient and the 2-stage Clos switch further improves energy efficiency. VC routers consumes the largest area and provide the worst throughput.

The network evaluation examines routers in the presence of the HOL problem. All routers are simulated in an 8×8 mesh network using random uniform traffic. SDM and VC demonstrate their capability of alleviating the HOL problem. The SDM routers using channel slicing and lookahead pipelines provide the best throughput. The minimum frame latency and the energy efficiency of all router structures have no apparent differences compared with the results presented in the single router evaluation because these performances are not affected by the HOL problem.

In the MPEG-4 system, different routers are utilized to support a practical traffic pattern with two hot spots. The wormhole router using synchronized pipelines satisfies the throughput requirements only when the data width is 48 bits or larger. Using channel slicing or SDM reduces the data width to 32 bits. If the data width is 48 bits, the wormhole router using channel slicing and lookahead pipelines, and all SDM routers

provide shorter average frame latency than the wormhole router using synchronized pipelines. In addition, the SDM router using synchronized pipelines demonstrates better energy efficiency than the wormhole router when larger data widths are used.

The raw data derived from the simulations in this Chapter are listed in Appendix C for further reference.

Chapter 9

Conclusions and Future Work

9.1 Summary of the thesis

Network-on-chips are the state-of-the-art communication fabric for current and future multiprocessor systems-on-chip. Most on-chip networks are built with synchronous circuits which are fast, area efficient and fully supported by EDA tools. However, synchronous on-chip networks are facing several design challenges: firstly, large area and power overhead is introduced by global clock trees in high-speed circuits. Secondly, technology scaling introduces worse delay variation due to the process variation, the run-time fluctuations of the die temperature and the difference of power density among in-die regions. Finally, IP cores requiring different clock frequencies lead to cross-clock domain issues. Asynchronous on-chip networks built with asynchronous routers, on the other hand, have no clock tree, show natural tolerance to delay variation and provide unified interfaces to all clock domains.

The majority of current asynchronous on-chip networks resemble the router structures and the flow control methods explored thoroughly in synchronous on-chip networks. Specifically speaking, low-level asynchronous pipelines are synchronized to implement wide pipelines and virtual channels using time division multiplexing are widely adopted. The synchronization issues introduced by these techniques have been shown to significantly compromise the throughput of asynchronous on-chip networks. This thesis explores the possibilities of using spatial parallelism rather than timing division methods to improve throughput. The work concentrates on the lowest two network layers: the physical layer and the switching layer.

9.1.1 Channel slicing and lookahead pipelines

Channel slicing and lookahead pipelines are introduced in Chapter 4. They are two physical layer techniques that can be used to reduce the period of asynchronous pipelines.

Channel slicing allows low-level asynchronous pipelines to transmit data independently whenever the data do not need to be analysed. The wide data links in current asynchronous on-chip routers synchronize the data transmission in all low-level pipelines. In wormhole routers, only a small portion (the header flit) of a frame is analysed while other parts are delivered directly. The synchronization of low-level pipelines prolongs the period and it is not necessary when data analysis is not required. Channel slicing removes the synchronization circuits and allows low-level pipelines to run independently. Extra control logic is inserted in the stages where data analysis is occasionally required. As a result, most data transmission proceeds at the highest speed while re-synchronization is still provided to occasionally pause the pipeline for data analyses.

The lookahead pipeline style is a self-timed pipeline technique using early evaluated acknowledgement. A lookahead pipeline stage is simply implemented by adding an AND gate on the ack line. It can be seamlessly connected to normal QDI pipelines. The pipeline stages in a router have different periods. The throughput of a router is determined by the slowest pipeline stage — the critical cycle. Because this critical cycle locates inside the router, utilizing the lookahead pipeline on this critical cycle reduces period without introducing any timing issues outside the router.

Channel slicing and lookahead pipelines can be used in any routers complying with the basic wormhole flow control method. A router implementation in Section 4.4.2 has demonstrated that using these two techniques in a 32-bit wormhole router achieves a 70% throughput boost with 28% area overhead.

9.1.2 SDM

Spatial division multiplexing (SDM) is a flow control method proposed in the switching layer for high network throughput.

Virtual channel (VC) is the most utilized flow control method in asynchronous routers for QoS support. It is well-known that VC improves network throughput by alleviating the head-of-line (HOL) problem. The blocked frames are temporarily stored in VC buffers allowing other frames to use the otherwise occupied switches and links.

The area overhead of utilizing VCs includes duplicated buffers, extra VC allocators and enlarged switches. VCs prolong the period as the switch is reallocated once per flit and the allocation latency cannot be hidden in asynchronous circuits.

Instead of duplicating buffers and letting multiple frames share the same resources in a time divided manner, SDM divides links and switches into multiple virtual circuits. Multiple frames can share the same links and switches concurrently by exclusively occupying a virtual circuit — a portion of the total bandwidth. Since both SDM and VC share links among multiple frames, they both alleviate the HOL problem and support QoS. The area overhead of SDM is shown to be smaller than VC. More importantly, since a virtual circuit is exclusively occupied by a frame, switches are reallocated once per frame instead of once per flit as in VC routers. No extra switch allocation latency is introduced by SDM and channel slicing can be used for further throughput increase.

Behavioural level simulation models have been built in Section 5.4 to compare the performance of different flow control methods. It is shown that SDM routers always outperform VC routers in area and throughput performance. Compared with wormhole routers, SDM routers show prominent throughput increase with moderate area overhead. Nevertheless, when the port data width is large enough, SDM routers outperform wormhole routers in area to throughput efficiency. In other words, when the port data width is large, SDM is the best flow control method for high throughput and low area consumption.

9.1.3 Clos

The major area overhead introduced by SDM is the enlarged central switch. The Clos network is theoretically the most area efficient structure for high-radix switches but it has not yet been implemented or scheduled by asynchronous circuits. Chapter 6 presents the first asynchronous Clos scheduler for asynchronous 3-stage Clos networks using a novel asynchronous dispatching algorithm. The asynchronous Clos scheduler has been implemented and compared with a synchronous Clos scheduler. It is shown that the asynchronous scheduler configures a 3-stage Clos network in a faster and more energy efficient way than the synchronous Clos scheduler.

A 2-stage Clos switch, which can be used in SDM routers, has been introduced in Chapter 6. When the number of switching modules in the first and third stages of a 3-stage Clos switch is equal with the number of directions in an SDM router, the third stage can be removed because all virtual circuits for the same direction are equivalent.

The 2-stage Clos switch incurs the smallest area overhead when the number of virtual circuits per direction is less than 18.

9.1.4 Overall remarks

An SDM router has been presented in Chapter 7 to demonstrate the way of combining all techniques together into a router design. Several routers using different technique combinations are implemented and compared in Chapter 7 and Chapter 8.

As a summary for the benefits and overheads of different techniques, the wormhole router using synchronized pipelines consumes the smallest area and provides the best energy efficiency when the overall bandwidth requirement is small. Channel slicing and lookahead pipelines significantly increase throughput by reducing the period but they compromise energy efficiency. Thanks to the capability of alleviating the HOL problem, SDM achieves a prominent throughput increase with moderate reduction in energy efficiency. Utilizing the 2-stage Clos switch in SDM routers reduces area overhead and saves power; however, the throughput is slightly compromised due to the blocking problem inside the Clos switch. The SDM routers using channel slicing and lookahead pipelines always provide the best throughput performance.

The VC router has demonstrated its ability of alleviating the HOL problem in the network evaluation. However, its large area overhead and long period make it the worst router structure for area efficiency and high throughput.

9.1.5 Discussion of the performance of sync/async NoCs

The performance comparison between synchronous and asynchronous NoCs is an interesting research issue. However, this thesis has not compared the asynchronous SDM routers against synchronous routers for the following reasons:

- *It is not one of the research objectives of this thesis.*

  Based on the findings that the synchronization utilized in state-of-the-art asynchronous on-chip routers introduces long period and compromises throughput, this thesis introduces several techniques which improve throughput without introducing extra synchronization if it is not reduced. Synchronization is a design overhead only in asynchronous circuits as synchronous circuits are naturally synchronized by the clock. Since the new techniques are proposed to cope with the

issues occurring only in asynchronous circuits, the performance comparison between asynchronous and synchronous implementations is irrelevant to revealing any advantages or limitations of the new techniques.

- *General conclusions have been provided.*

  Several asynchronous NoCs have been compared against synchronous NoCs. The MANGO router [12] was compared with the routers of the synchronous ÆTHEREAL NoC [56]. Both routers achieved around 500 MHz port speed with similar area overhead. The asynchronous SPIN (ASPIN) router [107] was compared with the synchronous SPIN (DSPIN) router. It was reported that ASPIN consumed 10% less area than DSPIN and the throughput of both routers was similar. The ANOC router was compared with DSPIN in [79] where ANOC consumed less power but DSPIN provided better throughput.

  General conclusions of the performance comparison between asynchronous and synchronous NoCs have already been provided in [79]: asynchronous routers are good candidates for low latency and low power applications while synchronous routers are more suited to low area and high throughput applications.

  Strong evidence supporting these conclusions has been provided in the pipeline analyses presented in [120]. For the data paths in routers, which can be recognized as simple pipelines, asynchronous multi-rail pipelines consume significantly larger area than synchronous pipelines. The latency of multi-rail pipelines is shorter than synchronous pipelines, since the forward latency of a 4-phase multi-rail pipeline stage is around 25% of the period while synchronous pipelines require a whole clock cycle. It is also shown that asynchronous multi-rail pipelines are less energy efficient than synchronous pipelines for data transmission. This provides a precondition of the low-power consumption claim of asynchronous NoCs: they are low power only when the throughput is low, in which case they rather benefit from the zero clock tree energy than suffer from the inefficient data transmission.

9.2 Future work

The research of asynchronous SDM routers is in its early stage. Many design areas have not been thoroughly investigated. In addition, the techniques proposed for asynchronous SDM routers can be utilized in other research directions. Some research

issues, which can be done in the near future, are listed as follows:

- The network interface for SDM routers is still an open question. It can be an advantage to expose all virtual circuits to the local processing element if it handles multiple communications concurrently. In this case, frames are naturally serialized in network interfaces. However, if the processing element does not need to handle concurrent communications, moving the frame serialization into the local port of the SDM router reduces area overhead and provides a unified interface compatible with wormhole routers.

- Channel slicing imposes a design challenge for the network interface design. The period of the critical cycle in wormhole routers has been reduced significantly. The critical cycle of network interfaces must be shorter than that of routers for the full speed benefit, otherwise network interfaces become the new throughput bottleneck. Lookahead pipelines or bundled-data pipelines may be utilized in network interfaces for this purpose.

- This thesis concentrates on best-effort throughput performance leaving the QoS support in SDM routers an open issue. It is known that SDM can support QoS by assigning virtual circuits with different priorities. In fact, the switch allocator is smaller rather than larger to support priorities because input virtual circuits with low priorities cannot compete for the output virtual circuits with high priorities. SDM routers provide guaranteed data transmission latency for all priority levels rather than only the highest one in asynchronous VC routers. It would be interesting to compare the performance of SDM and VC for supporting QoS.

- Similar with VC routers, SDM routers can be used to support adaptive routing algorithms. One virtual circuit can be reserved as the escaping channel in the same way as in a VC router.

- The current Clos scheduler is not QDI yet. Although it shows strong tolerance to delay variation, a QDI scheduler is proved tolerant to delay variation in theory, which is more reliable. However, a QDI scheduler is expected to be slower than the current one as all actions are explicitly indicated or detected.

- The current Clos scheduler supports only space-space-space Clos networks. The Clos networks utilized in IP networks are memory-space-memory networks where buffers are inserted in input modules and output modules. It is shown

that an asynchronous Clos scheduler is faster than a synchronous Clos scheduler. Expanding the scheduler to support memory-space-memory Clos networks may open a new research direction for asynchronous IP routers.

- It is also possible to replace the synchronous Clos scheduler in current IP routers with an asynchronous one for better power consumption. However, it is still required to expand the current asynchronous Clos scheduler to support memory-space-memory Clos networks.

- The iteration based switch allocation can be utilized in synchronous VC routers for better switch throughput and energy efficiency. As described by the parallel iterative matching algorithm, more requests are served by introducing extra iterations. The state-of-the-art VC router reallocates the central switch in every cycle. Utilizing an iterative switch allocation may achieve higher switch throughput. Because the switch is reallocated once per several iterations, the reallocation frequency is dropped and power consumption is therefore reduced.

The implementations of the routers in Chapter 7, including the wormhole, the SDM and the VC routers, have been made available at http://opencores.org/project,async_sdm_noc.

Bibliography

[1] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino. SPIN: A scalable, packet switched, on-chip micro-network. In *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, pages 70–73 suppl., 2003.

[2] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems*, 11(4):319–352, 1993.

[3] J. Bainbridge and S. Furber. Chain: a delay-insensitive chip area interconnect. *IEEE Micro*, 22:16–23, 2002.

[4] J. Bainbridge, W. Toms, D. Edwards, and S. Furber. Delay-insensitive, point-to-point interconnect using m-of-n codes. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 132–140, May 2003.

[5] A. Banerjee and S. W. Moore. Flow-aware allocation for on-chip networks. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 183–192, May 2009.

[6] E. Beigné, F. Clermidy, S. Miermont, and P. Vivet. Dynamic voltage and frequency scaling architecture for units integration within a GALS NoC. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 129–138, april 2008.

[7] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin. An asynchronous NOC architecture providing low latency service and its multi-level design framework. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 54–63, March 2005.

[8] V. E. Beneš. On rearrangeable three-stage connecting networks. *Bell System Technical Journal*, 41(5):1481–1492, September 1962.

[9] L. Benini and G. D. Micheli. Networks on chips: a new SoC paradigm. *IEEE Computer*, 35(1):70–78, 2002.

[10] D. Bertozzi and L. Benini. Xpipes: a network-on-chip architecture for gigascale systems-on-chip. *IEEE Circuits and Systems Magazine*, 4(2):18–31, 2004.

[11] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø. An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip. In *Proc. of International Symposium on System-on-Chip*, pages 171–174, 2005.

[12] T. Bjerregaard and J. Sparsø. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, pages 1226–1231, 2005.

[13] T. Bjerregaard and J. Sparsø. A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proc. of IEEE International Symposium on Asynchronous Circuits and Systems*, pages 34–43, 2005.

[14] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: from basic principles to state of the art. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):589 –607, April 2008.

[15] P. Bogdan and R. Marculescu. A theoretical framework for on-chip stochastic communication analysis. In *Proc. of International Conference on Nano-Networks and Workshops*, 2006.

[16] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. Routing table minimization for irregular mesh NoCs. In *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, pages 942–947, April 2007.

[17] A. Bystrov, D. Kinniment, and A. Yakovlev. Priority arbiters. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 128–137, 2000.

[18] T. J. Chaney and C. E. Molnar. Anomalous behavior of synchronizer and arbiter circuits. *IEEE Transactions on Computers*, 22(4):421–422, April 1973.

[19] H. J. Chao, K.-L. Deng, and Z. Jing. Petastar: a petabit photonic packet switch. *IEEE Journal on Selected Areas in Communications*, 21(7):1096–1112, September 2003.

[20] H. J. Chao, Z. Jing, and S. Y. Liew. Matching algorithms for three-stage bufferless Clos network switches. *IEEE Communications Magazine*, 41(10):46–54, October 2003.

[21] H. J. Chao, C. H. Lam, and E. Oki. *Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers*. John Wiley & Sons, Inc., 2001.

[22] J. Cheyns, C. Develder, E. V. Breusegem, D. Colle, F. D. Turck, P. Lagasse, M. Pickavet, and P. Demeester. Clos lives on in optical packet switching. *IEEE Communications Magazine*, 42(2):114–121, 2004.

[23] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729–738, July 2000.

[24] F. M. Chiussi, J. G. Kneuer, and V. P. Kumar. Low-cost scalable switching solutions for broadband networking: the ATLANTA architecture and chipset. *IEEE Communications Magazine*, 35(12):44–53, December 1997.

[25] T.-A. Chu. *Synthesis of self-timed VLSI circuits from graph-theoretic specifications*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1987. http://dspace.mit.edu/handle/1721.1/14794 [Online; accessed 2/12/2010].

[26] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. Wehn. A 477mW NoC-based digital baseband for MIMO 4G SDR. In *Proc. of IEEE International Solid-State Circuits Conference*, pages 278–279, 2010.

[27] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(5):406–424, March 1953.

[28] N. Concer, L. Bononi, M. Soulié, and R. Locatelli. CTC: an end-to-end flow control protocol for multi-core systems-on-chip. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 193–202, May 2009.

[29] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, 1997.

[30] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[31] W. J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):466–475, 1993.

[32] W. J. Dally and C. L. Seitz. The torus routing chip. *Distributed Computing*, 1(4):187–196, December 1986.

[33] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5):547–553, May 1987.

[34] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Proc. of Design Automation Conference*, pages 684–689, 2001.

[35] W. J. Dally and B. Towles. *Principles and Practices of interconnection networks*. Morgan Kaufmann Publishers, San Francisco, CA, 2004.

[36] R. Dobkin, R. Ginosar, and C. P. Sotiriou. Data synchronization issues in GALS SoCs. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 170–180, 2004.

[37] R. R. Dobkin and R. Ginosar. Two-phase synchronization with sub-cycle latency. *Integration, the VLSI Journal*, 42(3):367–375, June 2009.

[38] R. R. Dobkin, R. Ginosar, and A. Kolodny. QNoC asynchronous router. *Integration, the VLSI Journal*, 42(2):103–115, March 2009.

[39] R. R. Dobkin, Y. Perelman, T. Liran, R. Ginosar, and A. Kolodny. High rate wave-pipelined asynchronous on-chip bit-serial data link. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 3–14, 2007.

[40] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4:1320–1331, December 1993.

[41] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection networks: an engineering approach*. Morgan Kaufmann Publishers, 2003.

[42] T. Dumitras and R. Marculescu. On-chip stochastic communication. In *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, 2003.

[43] D. Edwards and A. Bardsley. Balsa: an asynchronous hardware synthesis language. *The Computer Journal*, 45(1):12–18, 2002.

[44] *FSC0H_D, UMC* $0.13\ \mu\mathrm{m}$ *Logic HS(FSG) process high density core cell library*, 2006. http://www.faraday-tech.com/dowloadDoc/techDocument/FSC0H_D_GENERIC_CORE%20Product%20Brief_v1.2.pdf [Online; accessed 19/01/2011].

[45] T. Felicijan. *Quality-of-service (QoS) for asynchronous on-chip networks*. PhD thesis, the Faculty of Science and Engineering, the University of Manchester, 2004. http://apt.cs.man.ac.uk/apt/publications/thesis/felicijan04_phd.php [Online; accessed 13/12/2010].

[46] T. Felicijan, J. Bainbridge, and S. Furber. An asynchronous low latency arbiter for quality of service (QoS) applications. In *Proc. of International Conference on Microelectronics*, pages 123–126, December 2003.

[47] T. Felicijan and S. B. Furber. An asynchronous on-chip network router with quality-of-service (QoS) support. In *Proc. of IEEE International SOC Conference*, pages 274–277, September 2004.

[48] S. Furber and J. Bainbridge. Future trends in SoC interconnect. In *Proc. of International Symposium on System-on-Chip*, pages 183–186, November 2005.

[49] S. Furber and P. Day. Four-phase micropipeline latch control circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(2):247–253, June 1996.

[50] R. Ginosar. Fourteen ways to fool your synchronizer. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 89–96, 2003.

[51] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *Journal of the ACM*, 41(5):874–902, September 1994.

[52] S. Golubcovs, D. Shang, F. Xia, A. Mokhov, and A. Yakovlev. Modular approach to multi-resource arbiter design. In *Proc. of International Symposium Asynchronous Circuits and Systems*, pages 107–116, May 2009.

[53] S. Golubcovs, D. Shang, F. Xia, A. Mokhov, and A. Yakovlev. Multi-resource arbiter decomposition. Technical report, School of Electrical, Electronic & Computer Engineering, Newcastle University, February 2009. http://async.org.uk/tech-reports/NCL-EECE-MSD-TR-2009-143.pdf [Online; accessed 17/12/2010].

[54] C. Gómez, M. E. Gómez, P. López, and J. Duato. Exploiting wiring resources on interconnection network: increasing path diversity. In *Proc. of Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 20–29, February 2008.

[55] M. E. Gomez, N. A. Nordbotten, J. Flich, P. Lopez, A. Robles, J. Duato, T. Skeie, and O. Lysne. A routing methodology for achieving fault tolerance in direct networks. *IEEE Transactions on Computers*, 55(4):400–415, April 2006.

[56] K. Goossens, J. Dielissen, and A. Radulescu. Æthereal network on chip: concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22:414–421, September 2005.

[57] J. Henkel, W. Wolf, and S. Chakradhar. On-chip networks: a scalable, communication-centric embedded system design paradigm. In *Proc. of International Conference on VLSI Design*, pages 845 – 851, 2004.

[58] J. Hu and R. Marculescu. DyAD: smart routing for networks-on-chip. In *Proc. of Design Automation Conference*, 2004.

[59] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam. Compact thermal modeling for temperature-aware design. In *Proc. of Design Automation Conference*, pages 878–883, 2004.

[60] *International Technology Roadmap for Semiconductors*, chapter Design, pages 12–13. 2009. http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_Design.pdf [Online; accessed 11/11/2010].

[61] M. B. Josephs and J. T. Yantchev. CMOS design of the tree arbiter element. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(4):472–476, December 1996.

[62] M. J. Karol, M. G. Hluchyj, and S. P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, 35(12):1347–1356, December 1987.

[63] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3(4):257–286, 1979.

[64] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta. Microarchitecture of a high radix router. In *Proc. of ACM/IEEE International Symposium on Computer Architecture*, pages 420–431, June 2005.

[65] M. Kim, D. Kim, and G. E. Sobelman. MPEG-4 performance analysis for a CDMA network-on-chip. In *Proc. of International Conference on Communications, Circuits and Systems*, pages 493–496, May 2005.

[66] D. J. Kinniment. *Synchronization and Arbitration in Digital Systems*. John Wiley & Sons Inc., 2007.

[67] L. Kleinrock. *Queueing System - Volume I: Theory*. John Wiley & Sons Inc., 1975.

[68] M. Krstić, E. Grass, F. K. Gürkaynak, and P. Vivet. Globally asynchronous, locally synchronous circuits: overview and outlook. *IEEE Design and Test of Computers*, 24(5):430–441, 2007.

[69] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N. Jha. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *Proc. of International Conference on Computer Design*, October 2007.

[70] A. Lankes, T. Wild, A. Herkersdorf, S. Sonntag, and H. Reinig. Comparison of deadlock recovery and avoidance mechanisms to approach message dependent deadlocks in on-chip networks. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 17–24, 2010.

[71] A. Leroy, D. Milojevic, D. Verkest, F. Robert, and F. Catthoor. Concepts and implementation of spatial division multiplexing for guaranteed throughput in networks-on-chip. *IEEE Transactions on Computers*, 57(9):1182–1195, September 2008.

[72] B. Li, L.-S. Peh, and P. Patra. Impact of process and temperature variations on network-on-chip design exploration. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 117–126, April 2008.

[73] K. S. Low and A. Yakovlev. Token ring arbiters: an exercise in asynchronous logic design with Petri nets. Technical Report 537, Department of Computer Science, University of Newcastle upon Tyne, November 1995. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.9060 [Online; accessed 13/12/2010].

[74] S. Majzoub, R. Saleh, and R. Ward. PVT variation impact on voltage island formation in MPSoC design. In *Proc. International Symposium on Quality of Electronic Design*, pages 814–819, March 2009.

[75] A. J. Martin. The design of a self-timed circuit for distributed mutual exclusion. Technical Report CaltechCSTR:1983.5097-tr-83, California Institute of Technology, 1983. http://resolver.caltech.edu/CaltechCSTR:1983.5097-tr-83 [Online; accessed 13/12/2010].

[76] A. J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In *Proc. of MIT conference on Advanced research in VLSI*, pages 263–278, Cambridge, MA, USA, 1990. MIT Press.

[77] A. J. Martin, A. M. Lines, and U. V. Cummings. Asynchronous circuits with pipelined completion process, 2002. US patent 6,502,180.

[78] P. Maurine, J. Rigaud, F. Bouesse, G. Sicard, and M. Renaudin. Static implementation of QDI asynchronous primitives. In *Proc. of International Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 181–191, 2003.

[79] I. Miro-Panades, F. Clermidy, P. Vivet, and A. Greiner. Physical implementation of the DSPIN network-on-chip in the FAUST architecture. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 139–148, April 2008.

[80] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost. HERMES: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38(1):69–93, 2004.

[81] D. E. Muller and W. S. Bartky. A theory of asynchronous circuits. In *Proc. of Annals of Computing Laboratory of Harvard University*, pages 204–243, 1959.

[82] R. Mullins and S. Moore. Demystifying data-driven and pausible clocking schemes. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 175–185, 2007.

[83] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. *Proc. of International Symposium on Computer Architecture*, 0:188–, 2004.

[84] R. Mullins, A. West, and S. Moore. The design and implementation of a low-latency on-chip network. In *Proc. of Asia and South Pacific Design Automation Conference*, pages 164–169, January 2006.

[85] J. Navaridas, M. Luján, J. Miguel-Alonso, L. A. Plana, and S. Furber. Understanding the interconnection network of SpiNNaker. In *Proc. of international conference on Supercomputing*, pages 286–295, New York, NY, USA, 2009. ACM.

[86] T. Nesson and S. L. Johnsson. ROMM routing on mesh and torus networks. In *Proc. of ACM symposium on Parallel algorithms and architectures*, pages 275–287, 1995.

[87] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das. ViChaR: a dynamic virtual channel regulator for network-on-chip routers. In *Proc. of Annual IEEE/ACM International Symposium on Microarchitecture*, pages 333–346, December 2006.

[88] J. L. Nunez-Yanez, D. Edwards, and A. M. Coppola. Adaptive routing strategies for fault-tolerant on-chip networks in dynamically reconfigurable systems. *IET Computers & Digital Techniques*, 2(3):184–198, May 2008.

[89] U. Y. Ogras, J. Hu, and R. Marculescu. Key research problems in NoC design: a holistic perspective. In *Proc. of IEEE/ACM/IFIP international conference on Hardware/Software Codesign and System Synthesis*, 2005.

[90] E. Oki, Z. Jing, R. Rojas-Cessa, and H. J. Chao. Concurrent round-robin-based dispatching schemes for Clos-network switches. *IEEE/ACM Transactions on Networking*, 10(6):830–844, 2002.

[91] E. Oki, N. Kitsuwan, and R. Rojas-Cessa. Analysis of space-space-space Clos-network packet switch. In *Proc. of Internatonal Conference on Computer Communications and Networks*, pages 1–6, August 2009.

[92] Open SystemC Initiative. *IEEE 1666: SystemC Language Reference Manual*, 2005. http://standards.ieee.org/getieee/1666/download/1666-2005.pdf [Online; accessed 25/01/2011].

[93] R. O. Ozdag and P. A. Beerel. High-speed QDI asynchronous pipelines. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 13–22, April 2002.

[94] G. Passas, M. Katevenis, and D. Pnevmatikatos. A 128 x 128 x 24Gb/s crossbar interconnecting 128 tiles in a single hop and occupying 6% of their area. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 87–95, May 2010.

[95] S. S. Patil. Forward acting n x m arbiter. Technical Report 67, Computation Structures Group, Massachusetts Institute of Technology, June 1972. http://csg.csail.mit.edu/CSGArchives/memos/Memo-67.pdf [Online; accessed 22/03/2011].

[96] L.-S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *Proc. of International Symposium on High-Performance Computer Architecture*, pages 255–266, January 2001.

[97] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. Fault tolerant algorithms for network-on-chip interconnect. In *Proc. of IEEE Computer society Annual Symposium on VLSI*, 2004.

[98] L. A. Plana, J. Bainbridge, S. Furber, S. Salisbury, Y. Shi, and J. Wu. An on-chip and inter-chip communications network for the SpiNNaker massively-parallel neural net simulator. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 215–216, 2008.

[99] L. A. Plana, S. B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, and S. Yang. A globally asynchronous, locally synchronous infrastructure for a massively-parallel multiprocessor. *IEEE Design and Test of Computers*, 24(5):454–463, 2007.

[100] S. Rodrigo, J. Flich, A. Roca, A. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato. Addressing manufacturing challenges with cost-efficient fault tolerant routing. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 25–32, 2010.

[101] R. Rojas-Cessa and C.-B. Lin. Scalable two-stage Clos-network switch and module-first matching. In *Proc. of Workshop on High Performance Switching and Routing*, pages 303–308, 2006.

[102] S. Scott, D. Abts, J. Kim, and W. J. Dally. The BlackWidow high-radix Clos network. In *Proc. of ACM/IEEE International Symposium on Computer Architecture*, pages 16–28, 2006.

[103] N. Seifert and N. Tam. Timing vulnerability factors of sequentials. *IEEE Transactions on Device and Materials Reliability*, 4(3):516–522, September 2004.

[104] D. Shang, F. Xia, S. Golubcovs, and A. Yakovlev. The magic rule of tiles: virtual delay insensitivity. In *Proc. of International Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 286–296, 2009.

[105] A. Sheibanyrad. *Asynchronous Implementation of a Distributed Network-on-Chip*. PhD thesis, University of Pierre et Marie Curie, 2008. ftp://asim.lip6.fr/pub/reports/2008/th.lip6.2008.sheibanyrad.1.pdf [Online; accessed 11/11/2010].

[106] A. Sheibanyrad and A. Greiner. Two efficient synchronous – asynchronous converters well-suited for networks-on-chip in GALS architectures. *Integration, the VLSI Journal*, 4(1):17–26, January 2008.

[107] A. Sheibanyrad, A. Greiner, and I. Miro-Panades. Multisynchronous and fully asynchronous NoCs for GALS architectures. *IEEE Design Test of Computers*, 25(6):572–580, 2008.

[108] Y. Shi, S. B. Furber, J. Garside, and L. A. Plana. Fault tolerant delay insensitive inter-chip communication. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 77–84, May 2009.

[109] K. S. Shim, M. H. Cho, M. Kinsy, T. Wen, M. Lis, G. E. Suh, and S. Devadas. Static virtual channel allocation in oblivious routing. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 38–43, 2009.

[110] M. Singh and S. M. Nowick. MOUSETRAP: ultra-high-speed transition-signaling asynchronouspipelines. In *Proc. of International Conference on Computer Design*, pages 9–17, September 2001.

[111] M. Singh and S. M. Nowick. The design of high-performance dynamic asynchronous pipelines: lookahead style. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(11):1256–1269, November 2007.

[112] W. Song and D. Edwards. Building asynchronous routers with independent sub-channels. In *Proc. of International Symposium on System-on-Chip*, pages 48–51, October 2009.

[113] W. Song and D. Edwards. An asynchronous routing algorithm for Clos networks. In *Proc. of International Conference on Application of Concurrency to System Design*, pages 67–76, 2010.

[114] W. Song and D. Edwards. A low latency wormhole router for asynchronous on-chip networks. In *Proc. of Asia and South Pacific Design Automation Conference*, pages 437–443, 2010.

[115] W. Song and D. Edwards. Asynchronous spatial division multiplexing router. *Microprocessors and Microsystems*, 35(2):85–97, 2011.

[116] W. Song, D. Edwards, J. L. Nunez-Yanez, and S. Dasgupta. Adaptive stochastic routing in fault-tolerant on-chip networks. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 32–37, 2009.

[117] J. Sparsø and S. Furber. *Principles of Asynchronous Circuit Design — A Systems Perspective*. Kluwer Academic Publishers, Boston, U.S.A, 2001.

[118] *SpiNNaker — A Universal Spiking Neural Network Architecture*. http://apt.cs.man.ac.uk/projects/SpiNNaker/ [Online; accessed 05/01/2011].

[119] M. B. Stensgaard and J. Sparsø. ReNoC: A network-on-chip architecture with reconfigurable topology. In *Proc. of ACM/IEEE International Symposium on Networks-on-Chip*, pages 55–64, 2008.

[120] K. S. Stevens, P. Golani, and P. A. Beerel. Energy and performance models for synchronous and asynchronous communication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(3):369–382, march 2011.

[121] I. Sutherland and S. Fairbanks. GasP: a minimal FIFO control. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 46 –53, 2001.

[122] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, 1989.

[123] Synopsys, Inc. *DC Ultra: Best-in-class Quality of Results that Correlate to Layout*, 2010. http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Documents/dc_ultra_ds.pdf [Online; accessed 25/01/2011].

[124] Y. Thonnart, E. Beigné, and P. Vivet. Design and implementation of a GALS adapter for ANoC based architectures. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 13–22, May 2009.

[125] D. N. Truong, W. H. Cheng, T. Mohsenin, Z. Yu, A. T. Jacobson, G. Landge, M. J. Meeuwsen, C. Watnik, A. T. Tran, Z. Xiao, E. W. Work, J. W. Webb, P. V. Mejia, and B. M. Baas. A 167-processor computational platform in 65 nm CMOS. *IEEE Journal of Solid-State Circuits*, 44(4):1130–1144, April 2009.

[126] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proc. of ACM symposium on Theory of computing*, pages 263–277, 1981.

[127] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, 43(1):29–41, january 2008.

[128] D. Wiklund and D. Liu. SoCBUS: switched network on chip for hard real time embedded systems. In *Proc. of International Parallel and Distributed Processing Symposium*, April 2003.

[129] P. T. Wolkotte, G. J. Smit, G. K. Rauwerda, and L. T. Smit. An energy-efficient reconfigurable circuit-switched network-on-chip. In *Proc. of IEEE International Parallel and Distributed Processing Symposium*, April 2005.

[130] J. Wu and S. Furber. A multicast routing scheme for a universal spiking neural network architecture. *The Computer Journal*, 53:280–288, 2010.

[131] K. Y. Yun, P. A. Beerel, and J. Arceo. High-performance asynchronous pipeline circuits. In *Proc. of International Symposium on Asynchronous Circuits and Systems*, pages 17–28, Los Alamitos, CA, USA, 1996.

[132] Z. Zhang, A. Greiner, and S. Taktak. A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip. In *Proc. of ACM/IEEE Design Automation Conference*, pages 441–446, June 2008.

Appendix

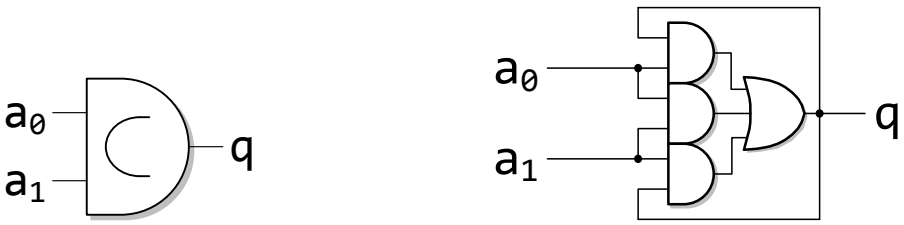
Appendix A

Basic Elements of Asynchronous Circuits

A.1 C-elements

A.1.1 2-input symmetric C-element

Symbol and schematic



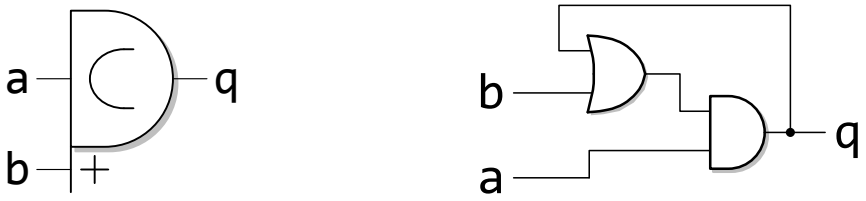
Verilog implementation

```
module c2 (a0, a1, q);
input  a0, a1;
output q;

  AO222EHD U1 ( .A1(q), .A2(a0), .B1(q), .B2(a1), .C1(a0), .C2(a1), .O(q) );

endmodule
```

The gate "AO222EHD" is a standard cell in the Faraday $0.13\ \mu\mathrm{m}$ standard cell library, as well as other gates in the Verilog implementations all over this appendix.

A.1.2 2-input asymmetric C-element with a plus input

Symbol and schematic

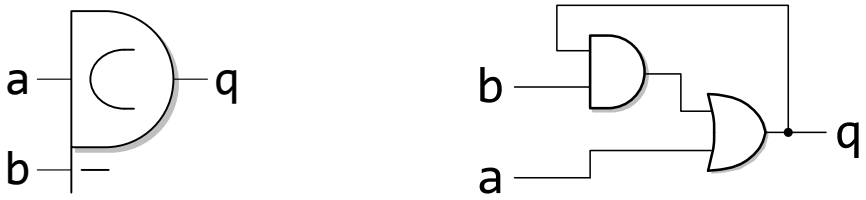


Verilog implementation

```
module c2p (a, b, q);

input  a, b;
output q;

  OA12EHD U1 ( .B1(b), .B2(q), .A1(a), .O(q) );

endmodule
```

A.1.3 2-input asymmetric C-element with a minus input

Symbol and schematic

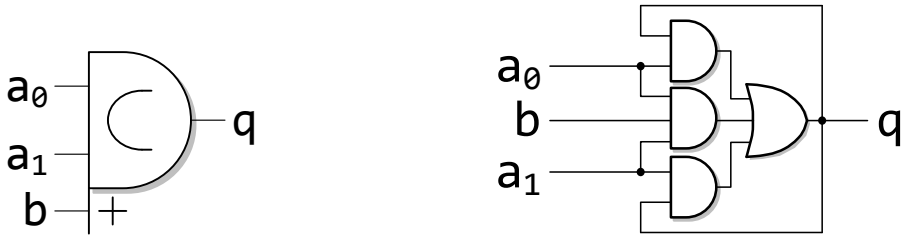


Verilog implementation

```
module c2n (a, b, q);

input  a, b;
output q;

  AO12EHD U1 (.B1(b), .B2(q), .A1(a), .O(q));

endmodule
```

A.1.4 3-input asymmetric C-element with a plus input

Symbol and schematic



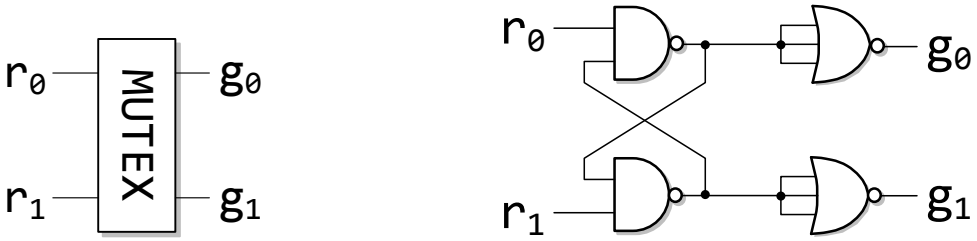
Verilog implementation

```
module c3p (a0, a1, b, q);

input  a0, a1, b;
output q;
wire   n1;

  OA12EHD U2 ( .B1(a1), .B2(a0), .A1(q), .O(n1));
  AO13EHD U1 ( .B1(a1), .B2(a0), .B3(b), .A1(n1), .O(q));

endmodule
```

A.2 Other cells

A.2.1 MUTEX

Symbol and schematic



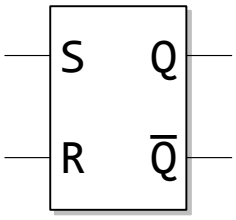
Verilog implementation

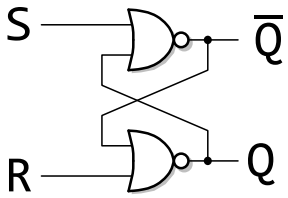
```
module mutex ( r0, r1, g0, g1 );
input  r0, r1;
output g0, g1;
wire   g0n, g1n;
```

```
  ND2HHD U1 ( .I1(r0), .I2(g1n), .O(g0n) );
  NR3HHD U2 ( .I1(g1n), .I2(g1n), .I3(g1n), .O(g1) );  // don't touch
  NR3HHD U3 ( .I1(g0n), .I2(g0n), .I3(g0n), .O(g0) );  // don't touch
  ND2KHD U4 ( .I1(r1), .I2(g0n), .O(g1n) );

endmodule
```

A.2.2 RS latch

Symbol and schematic





Verilog implementation

```
module rs_latch ( r, s, q, qn );
input  r, s;
output q, qn;

 NR2HHD U1 ( .I1(r), .I2(qn), .O(q));
 NR2HHD U2 ( .I1(s), .I2(q), .O(qn));

endmodule
```

Appendix B

Reproduction of the QoS NoC

The asynchronous VC router is a reproduction of the VC router in the QoS NoC presented in [45, 47]. The original router supports four VCs with different priorities. The reproduction has strictly followed all the design details presented in Chapter 8 in the thesis of Dr. Tomaz Felicijan [45] but with necessary modifications to support best-effort traffic instead of QoS traffic. Since the thesis is accessible on-line from http://apt.cs.man.ac.uk/publications/thesis/felicijan04_phd.php, only the modifications are explained in this appendix. Please refer to the original thesis for other design details.

B.1 Router structure

The VC router in QoS NoC is an asynchronous VC router using exactly the same router structure as synchronous VC routers. Most asynchronous VC routers, such as MANGO [12], ANOC [7] and QNoC [38], use extended central switches allowing all VCs in one direction to concurrently communicate with other VCs without arbitration. With this arbitration, the VC router in QoS NoC has the smallest central switch. However, the asynchronous behaviour of VC buffers compromises the throughput of QoS NoC, which makes it unfair to be compared with wormhole routers or SDM routers. The central switch in the reproduced VC routers is extended in the same way as in ANOC. The router structure, which was shown in Figure 8.9 in [45], has been modified and illustrated in Figure B.1. In this way, the contention problem as shown in Figure 8.10 in [45] is eliminated automatically.

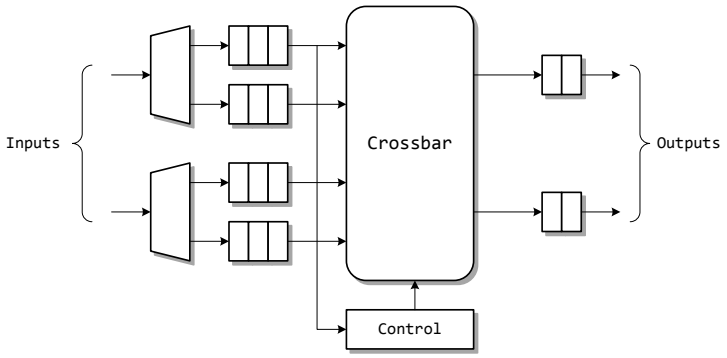


Figure B.1: Router structure

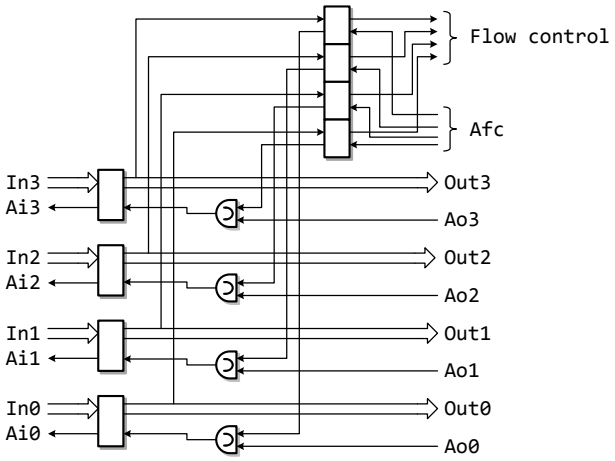


Figure B.2: Input buffer and crossbar interface

B.2 Connection of input buffers and the crossbar

The original QoS NoC router puts a multiplexer controlled by a multi-way MUTEX arbiter at the end of each input buffer. The arbiter randomly chooses an active VC and connects the VC to the central crossbar through the multiplexer. The new reproduction uses the extended crossbar as shown in Figure B.1. No multiplexers or arbiters are added at the end of input buffers. All VC buffers are connected independently to the central crossbar. The modified interface is shown in Figure B.2.

Similar with the original structure as depicted in Figure 8.13 in [45], a credit is sent back to the previous router when a flit is delivered by the last stage of a VC buffer. The C-element generating the ack for each VC ensures that a flit is only released after corresponding credit is received. The original design use a common ack line for the credit output since the arbiter actually constrains the maximum throughput to one flit at a time. The new structure lets all VCs run concurrently. As a result, multiple flits can

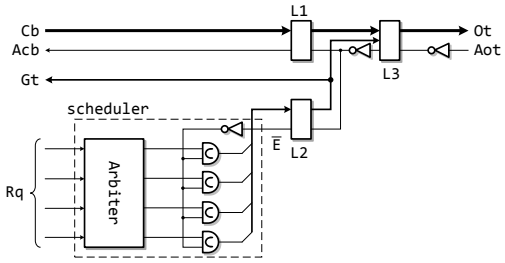


Figure B.3: Scheduler in an output buffer

be transmitted at the same time, which breaks the prerequisite of utilizing a common ack line for the credit output. Instead, every VC is connected to a separated credit output port with a separated ack line.

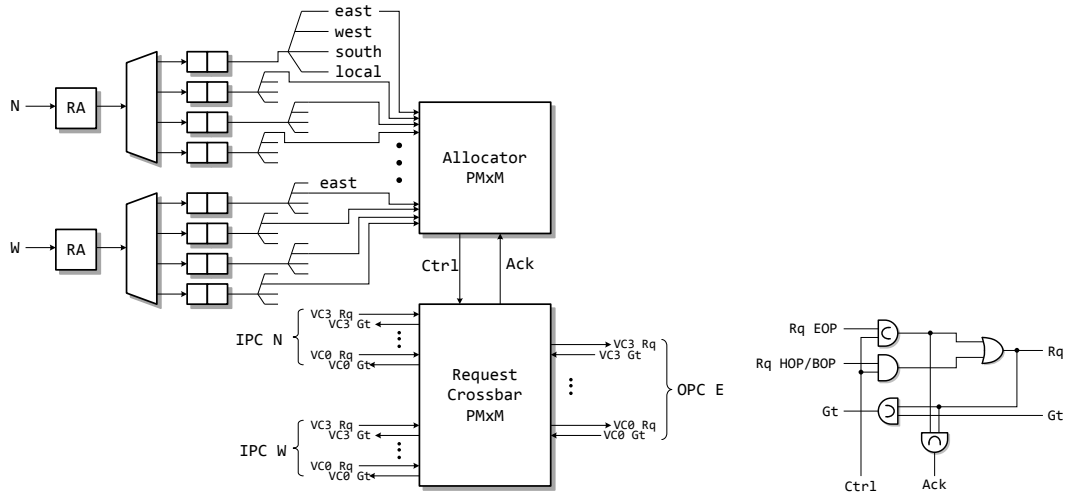
B.3 Scheduler in the output port

Figure 8.16 in [45] presented a block diagram of the scheduler in output ports without a detailed implementation of the scheduler. According to the STG illustrated in Figure 8.17 in [45], Figure B.3 demonstrates the implementation in the reproduction.

Multi-way MUTEX arbiters are used for the arbiter shown in Figure B.3 but other arbiter structures can be used, such as tree or ring arbiters. A column of C-elements are inserted between the arbiter and the L_2 buffer stage for VC numbers. These C-elements are controlled by the ack line from L_2; therefore, a new VC arbitration is accepted only when the flit corresponding to the previous VC arbitration is received by the next router. The captured VC arbitration (*Gt*) is sent back to the route manage unit (Section 8.6 of [45] and Section B.4) as an acknowledge for switch allocation requests (*Rq*).

B.4 Route management unit

The original QoS NoC router supports only QoS traffic. An input VC buffer requests only to the output VC buffers with the same priority. This is clearly illustrated in Figure 8.19 of [45] where the requests from the *VC3* buffers (the VC with the highest priority) in north and west input ports are connected only to the *VC3* in the east output port. The reproduction supports best-effort traffic. A VC in an input port can request any output VCs. The new route management unit is demonstrated in Figure B.4.



(a) Route management unit (b) Steering logic

Figure B.4: The operation of the route management unit

Figure B.4a shows the overall structure of the route manage unit for the east output port. The east requests of all input VCs are connected to an allocator to compete for the four output VCs (*VC0* to *VC3*) in the east output port. The M-N match allocator described in Section 6.3.1 is used. The allocation result (*Ctrl*) drives a crossbar which connects the switch requests from all input VCs to the scheduler in the east output port. Every cross-point in the crossbar is a steering logic shown in Figure B.4b.

The steering logic has been modified from the original design (Figure 8.19 in [45]). Instead of driving *Ack* by the detected end-of-packet request (*Rq EOP*) exclusively, the new *Ack* also depends on the value of the grant *Gt* from the scheduler in the east output port. The C-element on *Ack* defers the withdrawal of *Ack* until the end of data transmission while the original design drops *Ack* when *Rq EOP* is delivered. Since the structure of the arbiter in the original router design is unknown, it is uncertain whether the same problem exists in the original design. As for the allocator in the new router reproduction, dropping *Ack* before the release of *Gt* allows the allocator to assign the same output VC to another input VC before the end of current data transmission cycle. This leads to the rare but possible error of clashing between two flits.

Appendix C

Detailed implementation results

C.1 Single router evaluation

	Tile Area μm^2	Router Area μm^2	Frame Latency ns	Saturation Throughput MByte/s	Tile Power mW	Router Power mW
WH:16	25,283	15,728	116.2	366.1	8.3	3.4
WH:32	47,167	28,451	64.0	636.6	14.9	6.3
WH:48	71,883	41,640	44.2	925.4	22.9	9.0
WH:64	92,345	54,041	40.2	999.3	26.9	11.4
WH ChSlice+LH:16	38,721	25,552	70.2	597.3	21.6	9.1
WH ChSlice+LH:32	71,873	48,170	40.3	995.8	37.3	17.6
WH ChSlice+LH:48	108,659	73,643	28.1	1,361.1	53.8	26.9
WH ChSlice+LH:64	148,890	101,994	24.5	1,601.3	73.7	38.2
SDM:8x2	43,433	33,411	198.9	457.0	11.9	5.2
SDM:16x2	71,219	51,849	109.8	824.3	20.8	9.5
SDM:24x2	101,977	72,415	78.9	1,146.0	29.5	13.7
SDM:32x2	131,310	92,227	65.7	1,346.6	37.3	17.8
SDM:8x4	182,644	163,620	223.5	877.2	25.0	13.3
SDM:16x4	271,081	233,172	126.0	1,507.4	44.8	25.5
SDM ChSlice+LH:8x2	49,091	38,703	162.0	556.4	16.7	7.6
SDM ChSlice+LH:16x2	86,856	65,383	88.2	1,011.6	32.5	15.3
SDM ChSlice+LH:24x2	123,508	90,243	61.6	1,415.3	48.5	22.5
SDM ChSlice+LH:32x2	162,819	122,601	51.5	1,672.1	65.7	34.3
SDM ChSlice+LH:8x4	208,300	187,872	179.8	1,032.5	39.8	22.6
SDM ChSlice+LH:16x4	341,084	299,556	100.0	1,855.1	80.5	49.7

	Tile Area μm^2	Router Area μm^2	Frame Latency ns	Saturation Throughput MByte/s	Tile Power mW	Router Power mW
SDM-Clos ChSlice+LH:8x2	42,695	32,011	170.8	517.6	15.8	7.3
SDM-Clos ChSlice+LH:16x2	79,060	57,254	90.5	924.2	31.8	15.2
SDM-Clos ChSlice+LH:24x2	122,129	89,246	65.1	1,274.9	47.5	24.5
SDM-Clos ChSlice+LH:32x2	172,300	128,490	53.6	1,531.3	65.9	36.1
SDM-Clos ChSlice+LH:8x4	104,987	84,054	187.4	964.3	32.5	15.8
SDM-Clos ChSlice+LH:16x4	193,407	151,151	102.0	1,680.6	65.1	35.4
VC:8x2	47,152	40,152	321.9	183.5	8.7	4.1
VC:16x2	64,229	52,091	171.0	309.8	11.4	5.5
VC:32x2	99,609	78,332	90.3	544.7	18.1	9.3
VC:48x2	139,965	108,336	60.6	819.2	26.2	13.8
VC:64x2	172,000	133,156	52.0	949.3	32.0	17.8
VC:8x4	158,694	150,034	324.6	174.9	8.9	4.8
VC:16x4	184,997	171,990	186.2	320.9	13.5	7.3
VC:32x4	243,940	222,277	93.7	579.8	21.5	12.4
VC:64x4	355,135	315,974	58.8	933.1	32.9	19.3

C.2 Network evaluation

	Tile Area μm^2	Router Area μm^2	Min Frame Latency ns	Saturation Throughput MByte/Node/s	Tile Power mW	Router Power mW
WH:16	25,283	15,728	133.8	121.2	5.1	2.1
WH:32	47,167	28,451	81.0	207.8	9.1	3.9
WH:48	71,883	41,640	62.0	298.0	13.8	5.4
WH:64	92,345	54,041	58.4	322.3	16.0	6.8
WH ChSlice+LH:16	38,721	25,552	87.5	196.3	13.3	5.6
WH ChSlice+LH:32	71,873	48,170	58.4	320.1	22.2	10.6
WH ChSlice+LH:48	108,659	73,643	47.1	426.3	31.6	15.9
WH ChSlice+LH:64	148,890	101,994	43.4	480.6	42.4	22.2
SDM:8x2	43,433	33,411	220.5	180.3	8.8	3.8
SDM:16x2	71,219	51,849	131.7	309.7	14.5	6.5
SDM:24x2	101,977	72,415	100.4	420.9	21.1	9.7
SDM:32x2	131,310	92,227	87.0	504.8	26.1	12.3
SDM:8x4	182,644	163,620	252.0	366.9	19.7	10.5
SDM:16x4	271,081	233,172	153.8	637.2	35.5	20.1

	Tile Area μm^2	Router Area μm^2	Min Frame Latency ns	Saturation Throughput MByte/Node/s	Tile Power mW	Router Power mW
SDM ChSlice+LH:8x2	49,091	38,703	184.5	209.7	12.0	5.5
SDM ChSlice+LH:16x2	86,856	65,383	109.5	383.7	23.8	11.2
SDM ChSlice+LH:24x2	123,508	90,243	83.1	529.3	33.7	15.6
SDM ChSlice+LH:32x2	162,819	122,601	73.5	620.4	47.6	25.0
SDM ChSlice+LH:8x4	208,300	187,872	205.4	446.8	31.5	17.9
SDM ChSlice+LH:16x4	341,084	299,556	132.2	753.8	59.9	37.1
SDM-Clos ChSlice+LH:8x2	42,695	32,011	196.4	192.4	11.2	5.1
SDM-Clos ChSlice+LH:16x2	79,060	57,254	113.8	348.4	21.7	10.4
SDM-Clos ChSlice+LH:24x2	122,129	89,246	90.9	471.7	32.8	17.0
SDM-Clos ChSlice+LH:32x2	172,300	128,490	77.3	566.6	44.7	24.6
SDM-Clos ChSlice+LH:8x4	104,987	84,054	211.3	408.9	24.5	12.0
SDM-Clos ChSlice+LH:16x4	193,407	151,151	127.3	689.6	49.8	27.2
VC:8x2	47,152	40,152	409.5	66.0	6.0	2.9
VC:16x2	64,229	52,091	227.6	117.6	8.4	4.1
VC:32x2	996,09	78,332	138.4	211.1	13.1	6.7
VC:48x2	139,965	108,336	102.4	316.5	19.3	10.1
VC:64x2	172,000	133,156	91.4	369.0	23.9	13.2
VC:32x4	243,940	222,277	155.3	262.8	17.0	9.7

C.3 MPEG-4 evaluation

	Tile Area μm^2	Router Area μm^2	Avg. Frame Latency ns	Overall Throughput MByte/s	Tile Power mW
WH:16	25,283	15,728	10,258.0	1,974.4	5.4
WH:32	47,167	28,451	3,730.6	3,098.4	8.7
WH:48	71,883	41,640	150.9	3,415.0	9.6
WH:64	92,345	54,041	111.9	3,499.0	10.8
WH ChSlice+LH:16	38,721	25,552	5,335.4	2,805.9	12.1
WH ChSlice+LH:32	71,873	48,170	108.6	3,414.6	15.6
WH ChSlice+LH:48	108,659	73,643	54.4	3,413.6	15.1
WH ChSlice+LH:64	148,890	101,994	45.4	3,395.2	18.7

	Tile Area μm^2	Router Area μm^2	Avg. Frame Latency ns	Overall Throughput MByte/s	Tile Power mW
SDM:8x2	43,433	33,411	9,379.6	2,290.1	7.1
SDM:16x2	71,219	51,849	702.9	3,413.3	10.2
SDM:24x2	101,977	72,415	121.4	3,432.0	10.3
SDM:32x2	131,310	92,227	88.4	3,414.9	11.4
SDM:8x4	182,644	163,620	513.1	3,455.9	11.6
SDM ChSlice+LH:8x2	49,091	38,703	6,459.8	2,680.7	9.7
SDM ChSlice+LH:16x2	86,856	65,383	163.5	3,466.7	13.4
SDM ChSlice+LH:24x2	123,508	90,243	84.1	3,432.1	13.3
SDM ChSlice+LH:32x2	162,819	122,601	67.1	3,410.5	16.3
SDM ChSlice+LH:8x4	208,300	187,872	239.4	3,458.1	15.2
SDM ChSlice+LH:16x4	341,084	299,556	115.8	3,461.6	17.6
SDM-Clos ChSlice+LH:8x2	42,695	32,011	7,450.4	2,536.5	9.3
SDM-Clos ChSlice+LH:16x2	79,060	57,254	183.7	3,464.3	14.0
SDM-Clos ChSlice+LH:24x2	122,129	89,246	95.0	3,431.9	14.9
SDM-Clos ChSlice+LH:32x2	172,300	128,490	70.5	3,410.7	17.7
SDM-Clos ChSlice+LH:8x4	104,987	84,054	248.5	3,458.1	13.6
SDM-Clos ChSlice+LH:16x4	193,407	151,151	116.5	3,458.0	15.8
VC:8x2	47,152	40,152	17,291.0	918.2	5.5
VC:16x2	64,229	52,091	14,455.0	1,577.2	7.2
VC:32x2	996,09	78,332	6,940.8	2,672.4	10.8
VC:48x2	139,965	108,336	281.1	3,432.0	12.6
VC:64x2	172,000	133,156	172.9	3,511.7	14.3
VC:32x4	243,940	222,277	5,921.7	2,817.6	12.7