Design and Implementation of Networks-on-Chip: A Cost-Efficient Framework

ZHANG, Min

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

in

Electronic Engineering

The Chinese University of Hong Kong

March 2010

Abstracts of thesis entitled:

Design and Implementation of Networks-on-Chip:

A Cost-Efficient Framework

Submitted by **Min ZHANG**

for the degree of **Doctor of Philosophy**

in **Electronic Engineering**

at **The Chinese University of Hong Kong**

in March 2010

Integrating many processing elements (PE) in a single chip is inevitable as silicon technology allows more than one billion of transistors in a single piece of silicon. Networks-on-Chip (NoCs) has been proposed as a scalable solution to both increasing bandwidth requirements and physical design problems for multi-PE chips. However, as multi-PE chips drive the design focus to shift from the computation-centric to communication-centric, area and power costs consumed by communication has become comparable to what computation consumes.

This thesis tackles design and implementation of cost-efficient NoCs along two orthogonal directions. The first direction is to reduce area and power costs of a single virtual channel router. Through ASIC implementations, we find that allocator logic, including both virtual channel allocator (VA) and switch allocator (SA), consumes a large amount of costs. Based on RTL simulations for the entire NoCs, we identify great opportunities to reduce design costs of VA and then propose two low-complexity allocators: look-ahead VA and combined switch-VC allocator (SVA). Evaluations are performed for a wide range of traffic patterns and router parameters.

Results show that both proposed architectures significantly reduce area and power costs of allocators without penalties on network performances.

The second direction is to reduce hop counts of packets when they travel from sources to destinations, and thus to reduce power consumption of NoCs. The reduction of hop counts is realized by using a recently proposed express virtual channel (EVC) technique to virtually bypass intermediate routers. We study the EVC technique in two domains. The first domain is to present a high-level, application-specific methodology to improve power efficiency of EVC paths early in the design stage. The methodology includes three steps. Firstly, aggregate communication loads between routers are calculated. Secondly, an energy reduction model and an energy overhead model are developed. Finally, energy savings of all possible EVCs path are calculated and a greedy algorithm is applied to insert EVCs paths in an iterative way.

The second domain is to exploit the EVC flow control in design and implementation of low-power NoCs. We firstly present cost-efficient hardware components for both EVC source and EVC bypass routers, then propose a statistical approach to customize buffer architectures for EVC networks, then describe creative use of low-power circuit techniques such as clock gating and operand isolation for EVC routers, and finally evaluate EVC NoCs through detailed ASIC implementations. Results show that EVC NoCs can save up to 34.26% of power compared to baseline NoCs.

摘要

随着单块芯片上集成的晶体管达到十亿级，在单芯片上进行多核处理成为必然。多核芯片的通信结构需要巨大的带宽，物理实现上也非常困难。因此，研究者提出了片上网络来处理芯片内多个内核间的通信。然而，多核芯片使芯片设计从传统的以计算为中心向以通信为中心转变，从而使通信结构的成本上升到与计算结构的成本相提并论的地步。

本论文在两个相互正交的方向上研究低成本的片上网络的设计与实现。其一，减少单个包交换路由器的实现成本。通过 ASIC 实验，我们发现虚通道分配器和交换带宽分配器的成本很高。通过 RTL 仿真，我们发现虚通道分配器可以大大简化。因此，我们提出了两种低成本的分配器结构：look-ahead 虚通道分配器和组合的交换带宽－虚通道分配器。大量实验表明这两种结构能有效地减少分配器的成本，但并不会降低网络性能。

其二，通过减少包在传输过程中所经过的路由器的数量来减少功耗。快速虚通道技术是一种新近提出的流控技术，它能在逻辑上旁路包所经过的路由器。我们在两个方面对快速虚拟通道技术进行了研究。第一个方面，提出了一种上层的，基于应用的方法。该方法能在设计初期快速提高快速虚拟通道网络的能效，主要分为三个步骤。首先，计算路由器间的累计通信量。其次，对快速虚拟通道的节能模型进行建模。最后，计算所有可能的快速虚拟通道所能节约的能量，然后采用贪婪算法通过迭代来确定加入到网络中的快速虚拟通道。

第二个方面，为低功耗片上网络实现快速虚拟通道技术。我们为快速虚拟通路由器设计了低成本的功能模块，为快速虚拟通道网络提出了一种基于统计的存储单元优化方法，讨论了传统低功耗技术（如门控时钟和门控电路）在快速虚拟通道路由器中的应用，并通过 ASIC 实现对快速虚拟通道网络和基准网络进行了比较。结果表明快速虚拟通道网络最大能减少 34.26% 的功耗。

ACKNOWLEDGEMENTS

I would like to express my thanks to all those who have supported me in finishing this thesis. Foremost, I feel deep sense of gratitude to my advisor, Professor CHOY Chiu Sing, for his invaluable guidance throughout the course of my research. Without his vision, patience, inspiration, stimulating suggestions and encouragement, this thesis would have never been possible.

I am grateful to my wonderful committee members Professor KUMAR Shashi, Professor LEUNG Ka Nang, and Professor PUN Kong Pang for their insightful suggestions and comments on my research.

I am pleased to express my thanks to laboratory technician, Mr. YEUNG Wing Yee, who has helped me a lot for CAD tools. I am also grateful to other members in the VLSI and ASIC Laboratory for discussions and friendship. Special thanks to Dr. XU Ke who has helped me much in general ASIC designs. I am also thankful to AI Yan Qing for his help during chip physical implementations.

I am truly indebted to Dr. GRATZ Paul in the TRIPS team of the University of Texas at Austin for providing me the TRIPS OCN traffic traces of Minne-SPEC benchmarks and answering me many questions on how to process them.

Thanks to my friends who have made my time at CUHK enjoyable. I have had many memorable moments besides my research, such as badminton, soccer, and computer games.

Last but most importantly, my deepest thanks go to my parents ZHANG Yun Hai and TANG Yin Xiang, my brother ZHANG Liang, and my wife MA Li for their endless love, continuous encouragement, and unselfish support. This dissertation is dedicated to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBRIEVATIONS

AS-EVC	application-specific express virtual channel
ASIC	application specific integrated circuit
BW	buffer write
CG	communication graph
CMP	chip multi-processor
DSM	deep sub-micron
EVC	express virtual channel
FIFO	first in first out
FPGA	field programmable gate array
HoL	head-of-line
LT	link traversal
NoCs	networks-on-chip
NVC	normal virtual channel
OSI	open system interconnect
PC	physical channel
PE	processing element
PTP	point to point
QoS	quality-of-service
RAG	router aggregate communication graph

RC	routing computation
RCG	router communication graph
SA	switch allocation/allocator
SAF	store-and-forward
SoC	system-on-chip
ST	switch traversal
SVA	switch-virtual channel allocation/allocator
TG	topology graph
VA	virtual channel allocation/allocator
VC	virtual channel
VCT	virtual cut-through
WH	wormhole

CHAPTER 1. INTRODUCTION

Driven by DSM technologies, on chip interconnection structure is becoming the bottleneck for future SoCs and CMPs. NoCs, which is adapted from traditional off-chip networks, has been proposed as a promising solution to this problem [1-5]. However, design costs of NoCs are clearly a gap between today's technologies and those needed by future systems. This thesis aims to address cost-efficient design and implementation of NoCs. The introduction provides a brief overview of NoCs and the scope of this thesis.

1.1 THE EMERGENCE OF NOCS

Scaling down of silicon technology will allow chip complexities of more than one billion transistors on a single piece of silicon [6]. In order to efficiently utilize the exploding number of transistors, integrating multi PEs (or IP cores) in a single chip becomes inevitable. According to [7], the number of PEs in a SoC will increase to about 80 in 2010, 270 in 2015, and 880 in 2020. The communication infrastructure for such a SoC has to meet the following requirements:

- High throughput.
- Low latency.
- Low area and power costs.
- Scalable.
- Reusable.

Figure 1.1 shows examples of communication architectures: bus, PTP links, and NoCs. Let's analyse the pros and cons for them.

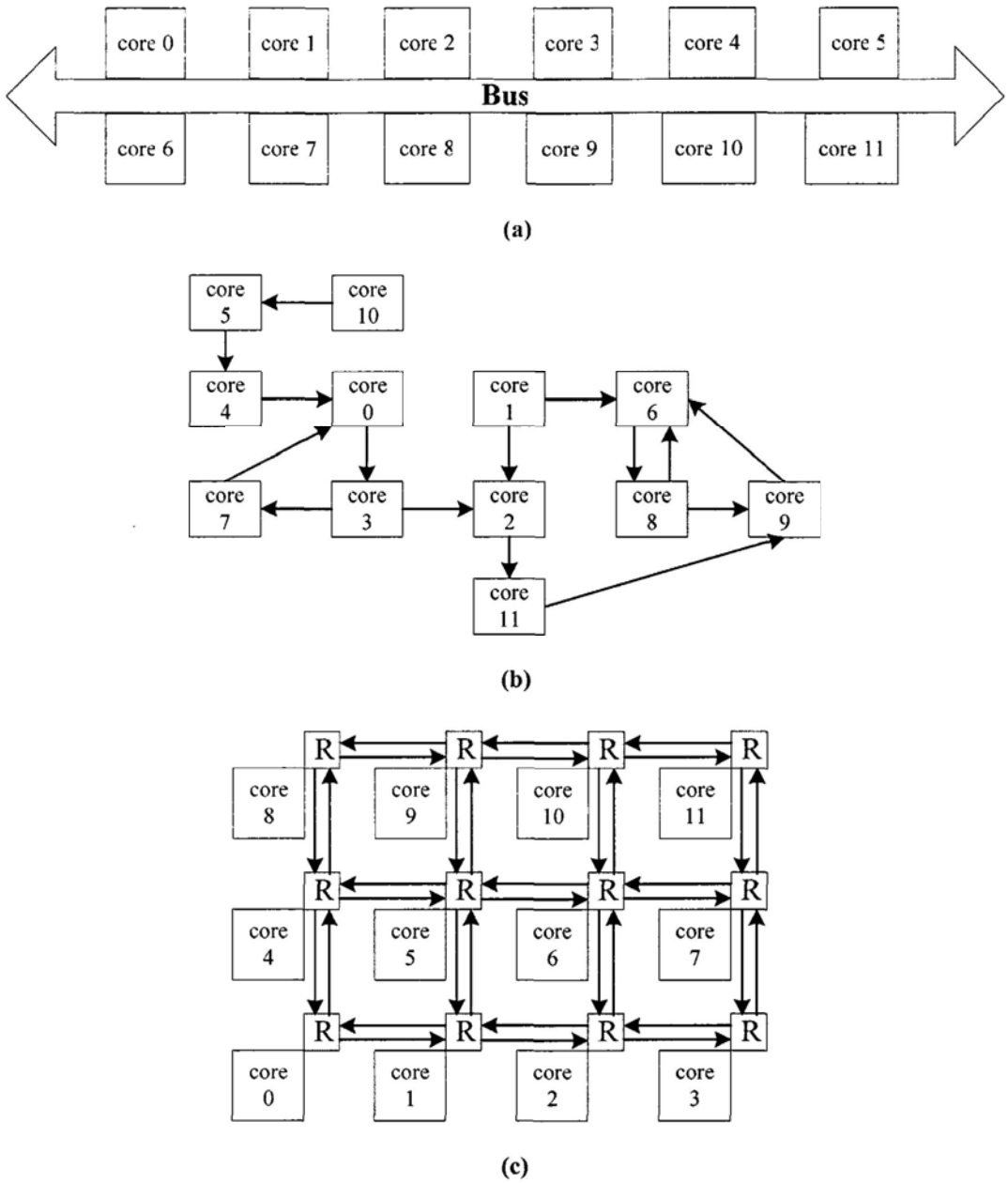


Figure 1.1. Examples of communication architectures. (a) Bus. (b) PTP links. (C) NoCs.

Busses can not provide high throughput for two reasons. First, there can be at most one transaction over busses at any point of time. Second, there are many global control signals for global arbitrations, which have long delay and then make the maximum operating frequency of busses low. Those using standard lightly pipelined interconnects are usually in the range of 80 to 150MHz, while the highest reported frequencies for pipelined interconnects are inching towards 250MHz [8]. Meanwhile, it is hard for busses to achieve low latency because of low operating frequencies. In

addition, busses fan out their wires to all targets because every data transfer is broadcast. As a result, power usage per data transfer is large due to the large capacitive load. What is more, the above bottlenecks will become more and more critical as more cores are attached to busses. Thus, busses are poor at scalability. Finally, busses are reusable since there are many well-developed bus standards.

Dedicated PTP links are optimal in terms of throughput, latency, and costs as they are designed specially for a given purpose. Nevertheless, they are bad in scalability because the number of wires, which becomes more and more costly, increases sharply ($O(n^2\sqrt{n})$) as the number (n) of cores increases [9]. Additionally, they are not reusable because they are fully customized for a given application. Designers have to completely change PTP architectures when applications are different.

NoCs can easily supply high throughput. First, many packets are allowed to traverse concurrently in a network. Second, the design frequency can be very high because of local processing and regular, short wires. In addition, NoCs is inherently scalable since it is a distributed communication architecture, which uses distributed routers, network interfaces, and structured wires. NoCs is reusable because it is based on the OSI protocol stack that decouples computation (cores) from communication (network). It makes the network transparent from the point of computing resources. Meanwhile, standard components library can be built for routers and network interfaces to reduce design efforts. However, although frequency is high, latency is a challenge because a packet has to pass multi hops from a source to a sink. In addition, power and area costs are critical challenges due to high design complexity of packet-switching routers.

In summary, share busses and PTP links can not meet requirements of future interconnection infrastructure to interconnect hundreds of cores. NoCs is a promising solution but can not be widely used unless some critical challenges are resolved [10].

1.2 NOCS BASICS

Figure 1.2 shows the layered architecture of a NoCs. The *system* includes a lot of processing elements. The work at this level is similar to that in a general large-scale SoC design, including mapping, task scheduling, modelling etc. At this level, messages or transactions are the basic datagram and design details of the network are not considered. The *network interface* decouples computation (the system) from communication (the network) and makes the network transparent from the point of the system. It handles the end-to-end flow control and break messages or transactions to packets that are delivered in packet-switching networks or streams that are delivered in circuit-switching networks. The *network* consists of routers and links. It sends packets from source routers to sink routers. Packets are further divided into flits or phits that are transferred along links. Flits are the flow control units whereas phits are the physical units that are the minimum size of datagram that can be transmitted in one link transaction. A flit could be made up of a series of phits. However, most commonly flits are equivalent to phits (We assume this throughout the thesis). The *link* level deals with the encoding/decoding, reliability and synchronization issues.

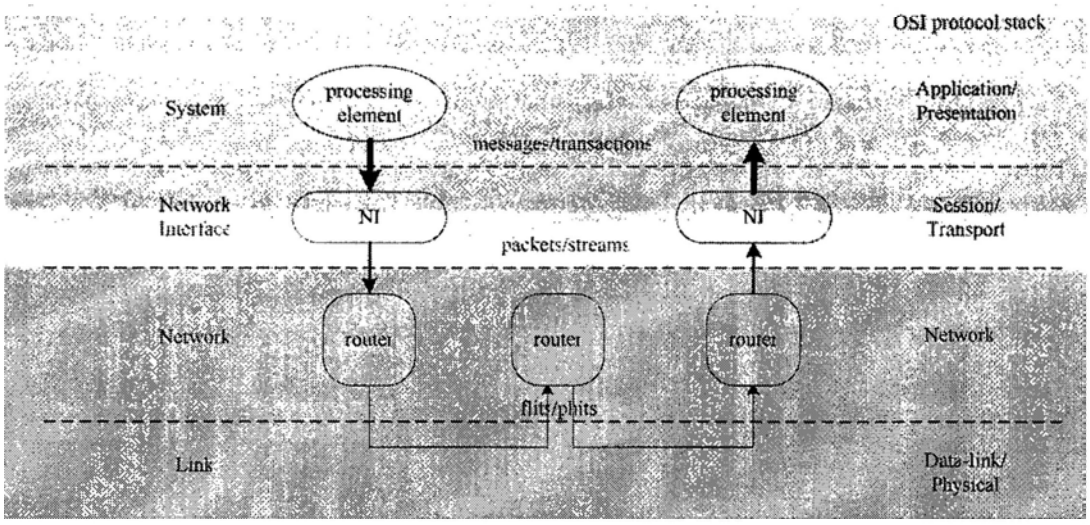


Figure 1.2. The layered architecture of a NoCs [11]

This thesis focuses on issues at the network level. A network is defined by the topology, routing, and flow control. The connection patterns of routers and links define the network's topology. Once a topology has been chosen, there can be many possible paths that a message could take through the network to reach its destination. Routing determines which of these possible paths a message actually takes. Once a path has been selected, flow control dictates which messages get access to particular network resources (channels and buffers) over time. The network is analogous to the traffic network in reality. The topology determines the roadmap, the routing method steers the car, and the flow control controls the traffic lights [12].

1.2.1 Network Topology

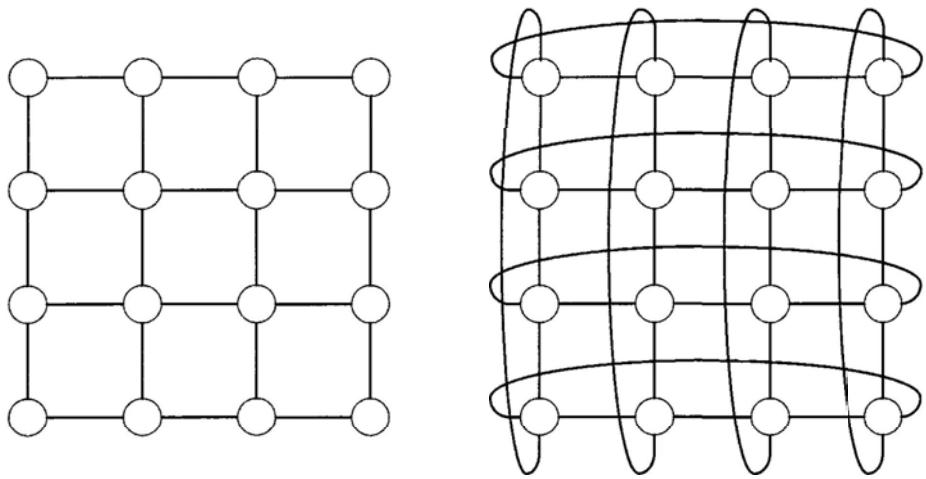
A NoCs is composed of a set of routers and links, and the topology of a network refers to the arrangement of these routers and links. In general, there are three classes of topologies: regular, semi-customized, and customized.

The mesh is the most simple and popular among regular topologies. Both the size of routers (except those on edges) and the length of links are regular. The torus is another popular regular topology. The difference between a mesh and a torus is that

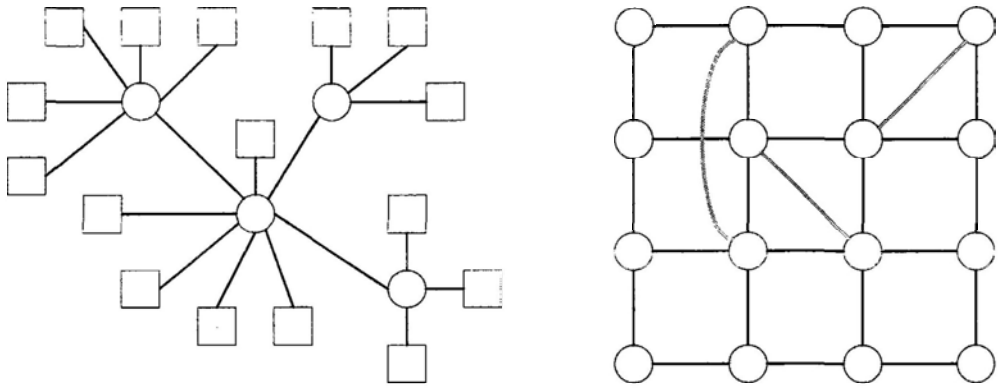
in a torus network, there is a wrap-around channel that connects the two edge nodes at each dimension, so that the hop count of the two edge nodes reduces to one. Regular topologies provide structured global interconnects that ensure well-controlled electrical parameters. However, they may become less attractive for application-specific designs.

On the other hand, customized topologies are specially designed for specific applications [13-16]. They improve network performances at the cost of altering the regularity of routers and channels. As shown in Figure 1.3 (c), there are two 4-port routers, one 6-port router, and one 8-port router. In addition, the length of channels varies largely.

These two extreme classes do not cover the whole design space of interconnection networks. In reality, many networks are neither completely regular nor completely customized. Thus, semi-customized topologies were proposed to explore the potential of using regular topologies in conjunction with a few customized long-range links, to improve performances with moderate overheads [17, 18].



(a) **(b)**



(c) (d)

Figure 1.3. Examples of network topologies. (a). A 4x4 mesh. (b). A 4x4 torus. (c). A customized topology. (d). A semi-customized topology. Circles, lines, and boxes respectively denote routers, channels, and PEs. For simplicity, a pair of channels, one in each direction, is represented by one line. The bold, red lines are express physical channels. For clarity, PEs are only shown for the customized topology.

1.2.2 Routing schemes

The routing scheme employed by a network determines the path taken by a packet from a source to a destination. In general, routing schemes can be classified into two categories: deterministic and adaptive. In a *deterministic* routing scheme, the traversal path of a packet is determined by its source and destination alone. Dimension-order routing is a popular deterministic routing, in which the packet follows one dimension first, then moves along another dimension toward the destination. In an *adaptive* routing scheme, the routing path of a packet not only depends on the source and destination of that packet, but also depends on the dynamic network status like link congestion. Deterministic routing has small design cost and traffic flows can be predicted well whereas adaptive routing can deal with network congestion by dynamically using alternative paths but has large design complexity.

1.2.3 Flow control

Flow control determines how resources of a network, like buffer space and channel bandwidth, are allocated to messages traversing the network. There are generally two categories of flow control strategies: circuit switching and buffered flow control. Furthermore, the buffered flow control includes three popular techniques: store-and-forward, virtual cut-through, and wormhole [19].

In *circuit switching*, a physical path from a source to a destination is reserved prior to the transmission of a message. This is accomplished by injecting the header flit into the network. This header flit contains the destination address and some additional control information. The header flit progresses toward the destination, reserving physical links as it traverses intermediate routers. When the header flit reaches the destination, a complete path has been set up and an acknowledgment is sent back to the source. The message contents may now be transmitted at the full bandwidth of the physical path. The circuit may be released by the destination or by the last few bits of the message. Circuit switching is well suitable for transferring infrequent and long messages that have much longer transmission time than the path setup time.

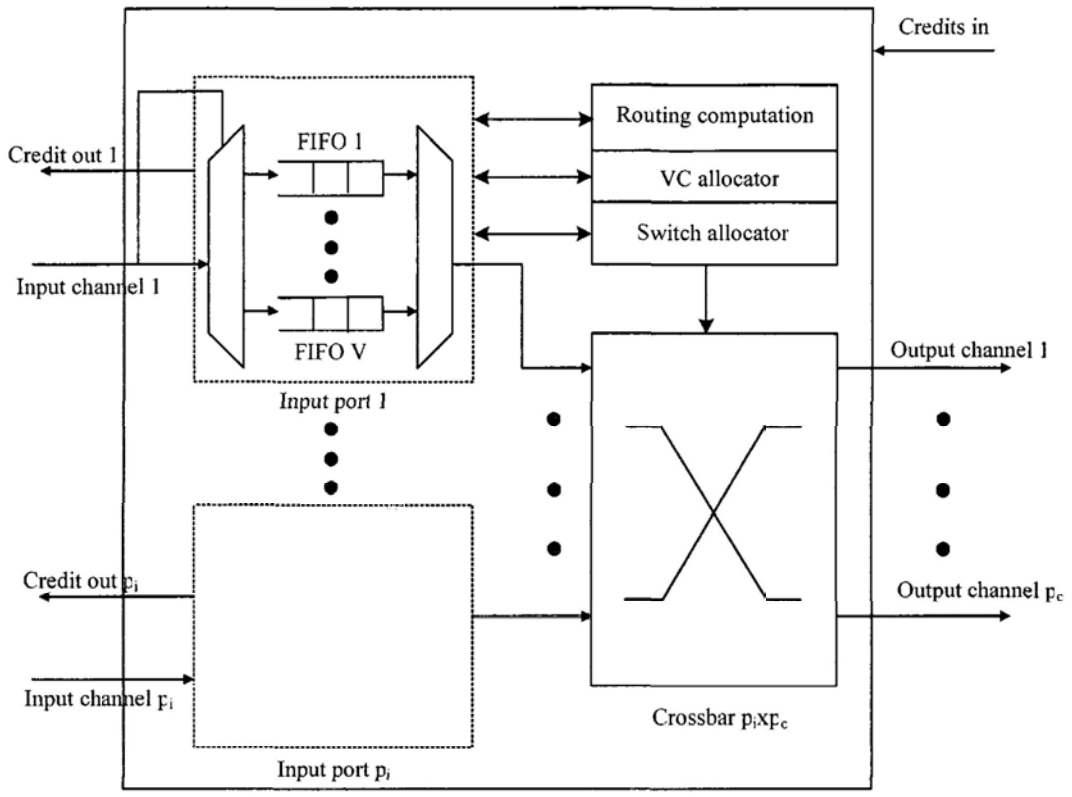
Alternatively, a message can be partitioned and transmitted as packets. Each packet is individually transferred from the source to the destination. In *store-and-forward* flow control, a packet is completely buffered at each intermediate router before it is forwarded to the next router. In order to buffer complete packets, large buffers have to be used. In addition, the transfer of a packet across the physical channel often takes multiple cycles. Although the routing information is typically available after the first few cycles, the routing decision can not be made before the entire packet is received. Thus, the latency of SAF flow control is high. In *virtual*

cut-through flow control, as soon as the next router has enough buffers for an entire packet, the current router can make routing decision and forward the header and following data bytes of the packet to the next router before the entire packet has been received at the current router. As a result, packets are pipelined through routers so that latency is small. But it also needs large buffers to save complete packets. In *wormhole* flow control, packets are also pipelined through routers. However, buffer requirements within routers are substantially reduced over the requirements for VCT flow control. A packet is partitioned into a set of flits and buffers within a router are required to store a few flits instead of complete packets.

Virtual channel flow control [20], which associates multi virtual channels with a single physical channel have many advantages. Firstly, it can avoid deadlocks. Since VCs are not mutually dependent to each other, one may break cycles in the resource dependency graph by adding VCs. Secondly, it can remove HoL blocking and increase utilization of physical channels. HoL means that a packet at the head of a VC whose designated output channel is busy will block subsequent packets in that VC from being transmitted even if their own designated output channels are free. As a result, VC flow control reduces packet latency and improves network throughput. In addition, VC flow control can be used to and provide QoS by allowing high priority message streams overtake those of lower priority. Therefore, although some researchers don't use VCs in their NoCs [1, 4], VC flow control is the prevailing scheme for NoCs [21-25]. However, implementing VC flow control results in area and possibly power and latency overhead because it requires more buffers and larger control logics to manage the VCs.

1.2.4 Router microarchitecture

Figure 1.4 (a) and (b) demonstrate the microarchitecture and the pipeline of a VC router. The router has p_i input and p_o output physical channels/ports, supporting V VCs per port. The FIFOs in input ports buffer arriving flits (BW stage). The routing computation directs the head flit of an incoming packet to the appropriate output physical channel (RC stage). The VC allocator arbitrates among all *input VCs* (VCs of input ports) which request the same *output VCs* (VCs of output PCs. In fact, VCs of an output PC are VCs of the connected input port at the downstream router.) and assigns available output VCs to successful input VCs (VA stage). The switch allocator distributes output PCs and the crossbar to input VCs (SA stage). The crossbar passes flits to appropriate output PCs (ST stage). Finally, flits traverse output PCs to the next router (LT stage). Each head flit passes five pipeline stages whereas each body/tail flit passes four pipeline stages. There is no RC stage in the pipeline because the look-ahead routing computation [26] is applied, where the route of a packet is computed one hop in advance.



(a)

Head flit	BW	VA	SA	ST	LT	
		Body/tail flit	BW	SA	ST	LT

(b)

Figure 1.4. A virtual channel router. (a) Microarchitecture. (b) Pipeline.

1.2.5 Buffer organization

Buffers are one of the most important structures in a router [24]. On the one hand, buffers have large impacts on network performances. On the other hand, in by far the most NoCs architectures, buffers account for the main parts of area and power costs in the router [2, 27-29]. There are generally two types of buffer organizations (Figure 1.5): a central buffer and a separate buffer.

In a central buffer organization, a large memory pool is shared across all virtual channels of a physical channel. In a separate buffer organization, a small buffer memory is provided for each virtual channel. The central buffer always uses the dynamical allocation mechanism that realizes sharing of memory spaces and leads to good memory utilization. However, control logic for dynamic allocation is very complex, generating high latency and power costs. On the contrary, the separate buffer organization requires very simple control logic because memory spaces are allocated statically but it has poor buffer utilization since memory spaces of idle virtual channels cannot be allocated to busy virtual channels. In general, if a NoCs aims to support a range of applications and traffic characteristics of these applications are unknown, the dynamically allocated central buffer structure is preferred in order to provide flexible, efficient use of memory spaces. However, if traffic characteristics of a given application can be well predicted, the statically-allocated separate buffer structure is better because memory spaces for each virtual channel can be customized to significantly improve buffer utilization.

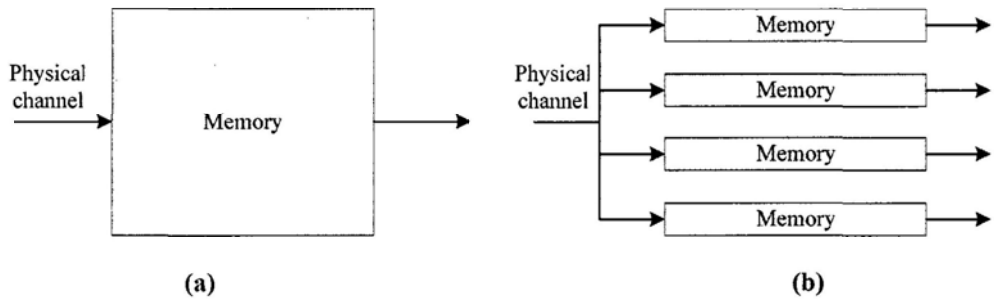


Figure 1.5 Buffer organization. (a) A central buffer. (b). A separate buffer.

1.2.6 Network performance metrics

Latency for a packet is the time required for it to traverse the network from source to destination, including the time that the packet buffers in the source queue before it enters the network. Generally, graphs of average latency vs. offered traffic are used.

The average latency is the mean value of the latencies of all measured packets. The offered traffic is the rate at which packets are generated by packet sources, which is also known as applied load, generation rate, or injection rate. We use injection rate throughout this thesis and calculate it in two steps. Firstly, injection rate for each node is computed by counting the number of flits (a packet has a number of flits) entering the network from this node and dividing it by the time interval (cycles). Then, injection rate for the network is calculated as the mean value of injection rates of all nodes. Thus, its unit is flits/(node*cycle) (or flits/node/cycle).

Throughput is the rate at which packets are delivered by the network. It is also called accepted traffic that is contrasted with the offered traffic. Generally, graphs of average throughput vs. offered traffic are used. Like the injection rate, we calculate average throughput for the network by firstly calculating throughput for each node and then computing the mean value of all throughputs. Thus, its unit is also flits/(node*cycle).

1.3 COST-EFFICIENT DESIGN FRAMEWORK

1.3.1 Motivations

Interconnection networks have been used in off-chip domain (chip-to-chip, board-to-board, etc.) for many years, where the only goal is to achieve the highest possible performances (packet latency, network throughput, etc.). However, as interconnection networks shift to on-chip domain, a critical challenge is to keep their design costs (area and power) small, especially when they are applied for portable SoCs or embedded devices.

Costs of NoCs have become comparable to on-chip computation costs. On the one hand, NoCs consumes a large portion of the chip area. For example, TeraFLOPS has

3 mm^2 (in 65nm technology) tiles and 53-kilogates routers account about 17% of the transistors [30]. In general, the 3 mm^2 tiles are large. Thus, on-chip routers will account more percentage of the chip area when PEs are modest-sized [31]. On the other hand, NoCs consumes a large amount of the chip power. For instance, in the MIT Raw microprocessor, the on-chip networks consume 36% of the total chip power, only 9% lower than what the main processor consumes [32]. These examples demonstrate the importance to reduce area and power costs of NoCs.

Virtual channel flow control is widely applied in NoCs to obtain high throughput by efficiently sharing network physical channels. However, it comes with high area and power costs because of the hardware complexity of VC routers. As shown in Figure 1.4 (a), in a VC router, a number of VCs are associated with a single physical channel. Since each VC is implemented as a separate memory, a large number of buffers are needed that are the major part of the router area. Meanwhile, complicated control blocks such as the virtual channel allocator and the switch allocator are required to handle sharing of buffers and crossbar bandwidth. Both the buffers and the control blocks include many registers. These registers and their associated clock tree consume a certain amount of power even when the network is idle and no packets are travelling in it, which is referred to *standby power* [33, 34]. In addition, as a packet travels from a source to a sink, it dissipates additional *stream power* for buffer access (write and read), arbitration (routing computation, VC allocation, switch allocation, etc.), and crossbar traverse. Therefore, routers dominate NoCs power. Implementation results show that routers consume two to three times more power than physical links in deep submicron technologies [35].

In summary, we concentrate on reducing area and power costs of routers in this thesis because they are much more costly than links. In addition, as mentioned before, we do not address network interfaces.

1.3.2 Contributions

This thesis aims to achieve cost-efficient design and implementation of NoCs in two levels: router microarchitecture and network architecture.

In the router microarchitecture level, low-cost hardware implementations for the VC allocator and the switch allocator, which are the largest two components in the control path of a VC router, are proposed. By running simulations for the entire NoCs and investigating utilization statistics of arbiters in a generic VA, we find big opportunities those cannot be identified by analysing the generic VA in isolation, to simplify design complexity of the generic VA. Then, we propose two low-cost allocators: a look-ahead VA and a combined switch-VC allocator (SVA). However, arbiters sharing in the SVA leads to deadlock problem. Thus, we study deadlock problem for the SVA. Finally, we present the effects of VA simplifications on the critical paths of the VA and the SA.

In the network architecture level, we exploit a new flow control mechanism, express virtual channel [36], to reduce power. We propose a novel methodology to insert EVC paths in an-application specific manner by exploiting communication characteristics of various applications, with the main objective to reduce stream power as much as possible. The AS-EVC method consists of three steps. First, calculate the aggregate communication volumes between any pair of routers. Second, calculate power savings for all possible EVC paths based on analytical energy models. Third, insert EVCs using a greedy algorithm subject to several insertion

rules. Because all the calculations are based on analytical models, the AS-EVC method can help designers quickly insert power-efficient EVC paths for applications in the early design stage.

Furthermore, we study design and implementation issues for NoCs with the EVC flow control in three aspects. First, we design special hardware components for EVC source and bypass routers. Second, we propose a simulation-based, statistical approach to customize buffer organization after EVCs insertion. Third, we explore low-power circuit techniques like clock gating and operand isolation to save power as much as possible when an EVC flit travels an EVC bypass router.

The rest of this thesis is organized as follows. Chapter 2 presents two cost-efficient allocator implementations for NoCs routers. Chapter 3 proposes an application-specific EVC insertion methodology for power-efficient NoCs. Chapter 4 presents design and implementation of cost-efficient NoCs with the EVC flow control. Next, chapter 5 concludes this thesis. Finally, Appendix A describes the application-specific EVC insertion tool, Appendix B shows the application-specific buffer customization tool, and Appendix C presents the fully-synthesizable parameterized NoCs library.

CHAPTER 2. COST-EFFICIENT ALLOCATOR IMPLEMENTATIONS

2.1 INTRODUCTION

Virtual channel flow control is the prevailing scheme for NoCs because it provides high throughput through multiplexing of physical channels and it has smaller costs than store-and-forward and virtual cut-though flow control schemes. However, costs of VC routers are still too large to be used in NoCs for practical CMPs, especially for cost-sensitive portable SoCs and embedded devices.

Many researchers have made efforts to reduce area and power costs of on-chip routers. However, they only focused on components in the data path of a router such as buffers and a crossbar. Some power-efficient components like a segmented-crossbar, a cut-through crossbar, and a write-through input buffer were proposed in [37]. Buffers for input ports were customized to reduce large buffer costs in [29, 38]. Although it is indeed reasonable to preferentially study components in the data path of a router because they consume the largest parts of costs, costs of components in the control path are not negligible. Nevertheless, few researchers address reducing costs for components in the control path.

Design costs of a generic virtual channel allocator and a generic switch allocator are comparable to buffers and a crossbar in a VC router. This chapter focuses on the reduction of their design costs. The contributions include the following aspects.

- We study virtual channel and switch allocators in the context of the entire NoCs. This is different from previous studies where they were addressed in isolation. By running simulations for the complete NoCs and investigating utilization statistics of arbiters in the allocators, we find great optimization

opportunities to reduce design costs of the allocators. These opportunities would not be found if we just investigate the allocators themselves.

- We propose three methods to simplify the generic VC allocator gradually. First, the p_iV *V:1* arbiters in the first stage are totally removed and the number of $p_iV:1$ arbiters in the second stage is decreased from p_oV to p_o. Second, the number of *V:1* arbiters at each input port is reduced from p_o to 1 through logic sharing. Third, the simplified VA and the generic SA share a *V:1* arbiter at each input port and a $p_i:1$ arbiter at each output port.

- Sharing arbiters by the generic VA and the generic SA make VA requests and SA requests dependent on each other. This dependency may lead to deadlock problem. We study the deadlock problem and two kinds of solutions to it.

- We present effects of the three simplification methods on the critical paths of the VA and SA pipeline stages.

- We evaluate performances, delay, area, and power costs of the generic architecture, the look-ahead architecture, the combined architecture through RTL-level simulations and VLSI implementations for a wide range of design parameters and traffic patterns.

The structure of this chapter is as follows. Section 2.2 reviews the generic VA and SA architectures and describes the motivations. Following, Section 2.3 illustrates how to simplify the generic VA in a step-by-step way. Next, Section 2.4 handles deadlock problem for the SVA. After that, Section 2.5 presents effects of simplification methods on the VA and SA pipeline stages. Then, Section 2.6

demonstrates evaluations in terms of both network performances and implementation costs. Finally, Section 2.7 concludes this thesis chapter.

2.2 RELATED WORK

2.2.1 Basics of allocators

An allocator performs a matching between a group of resources and a group of requesters, each of which may request one or more of the resources. The allocator can be considered as accepting a $n \times m$ request matrix R containing the individual requests, r_{ij} and generating a grant matrix G containing the individual grants, g_{ij}. R is an arbitrary binary-valued matrix. G is also a binary-valued matrix that only consists ones in entries corresponding to non-zero entries in R (This ensures that a grant can be asserted only if the corresponding request is asserted), has at most one one in each row (This ensures that at most one grant for each requester may be asserted), and at most one one in each column (This ensures that at most one grant for each resource can be asserted.) [12]. Examples of request and grant matrices for a 4×3 allocator are as below.

$$R = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad G = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

An allocator can be implemented in an exact or separable way. Figure 2.1 (a) shows a 4×3 exact allocator, in which a maximum matching can always be found through iteratively augmenting a sub-maximum matching [12]. However, it is too slow and its design complexity is too high. Figure 2.1 (b) demonstrates a 4×3 separable allocator, in which allocation is performed as two sets of arbitration: one across the requesters and one across the resources. The separable allocator admits a

much simple implementation while sacrificing a small amount of matching efficiency compared to the exact allocator. Therefore, separable allocators are generally applied in routers where allocators must make allocations with low latency and low design costs.

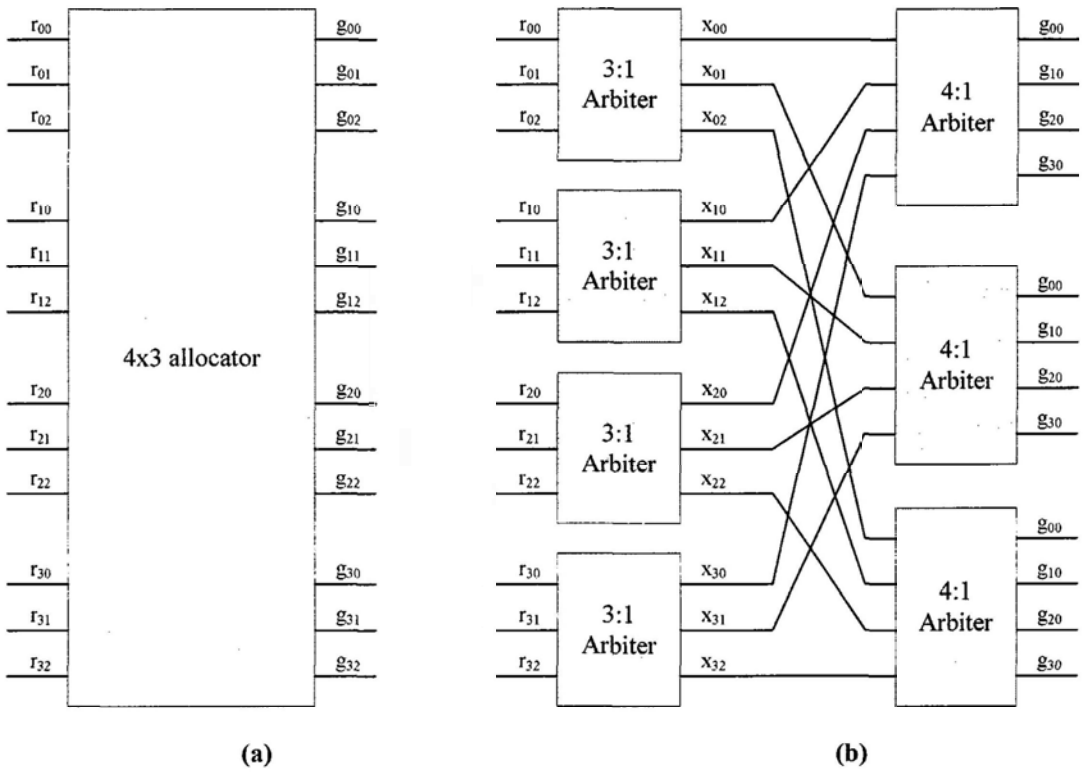


Figure 2.1. (a) A 4×3 exact allocator. (b). A 4×3 requester-first separable allocator. [12]

2.2.2 The generic virtual channel allocator

The generic architectures of a virtual channel allocator and a switch allocator were presented in [39]. The range of the routing function determines the complexity of a VA. If the routing function returns at most one[1] VC of a single output PC, the VA needs only arbitrate among input VCs that are competing for the same output VC. If

[1] If there are idle output VCs at the output PC, one of them will be returned. If there are no idle output VCs at the output PC, no output VC will be returned.

the routing function is more general and returns any candidate VCs of a single output PC, the VA needs additionally arbitrate among V output VCs for each input VC. If the routing function is the most general and returns all possible candidate VCs of all output PCs, the VA needs additionally arbitrate among p_oV output VCs for each input VC. *The routing function that returns any candidate VCs of a single output PC is the most general possible in a router with deterministic routing* [39]. Thus, the allocator architecture shown in Figure 2.2 (the generic VA in this chapter) is widely used in VC routers [21, 24].

The generic VA performs arbitration in two stages. In the first stage, each input VC selects one available VC from returned output VCs. Since there are at most V available VCs in an output PC, a *V:1* arbiter is needed for each input VC. In the second stage, each output VC grants one from the winning requests of the first stage allocation. The number of requests to an output VC is p_iV in the worst case, so each output VC needs a $p_iV:1$ arbiter. As shown in Figure 2.3, a large *piV:1* arbiter is generally simplified by organizing it as a tree of smaller arbiters [21]. The *V:1* arbiters arbitrate between requests from the same input ports and the $p_i:1$ arbiter determine the winning input port. The tree architecture much reduces design costs with some penalty of matching efficiency.

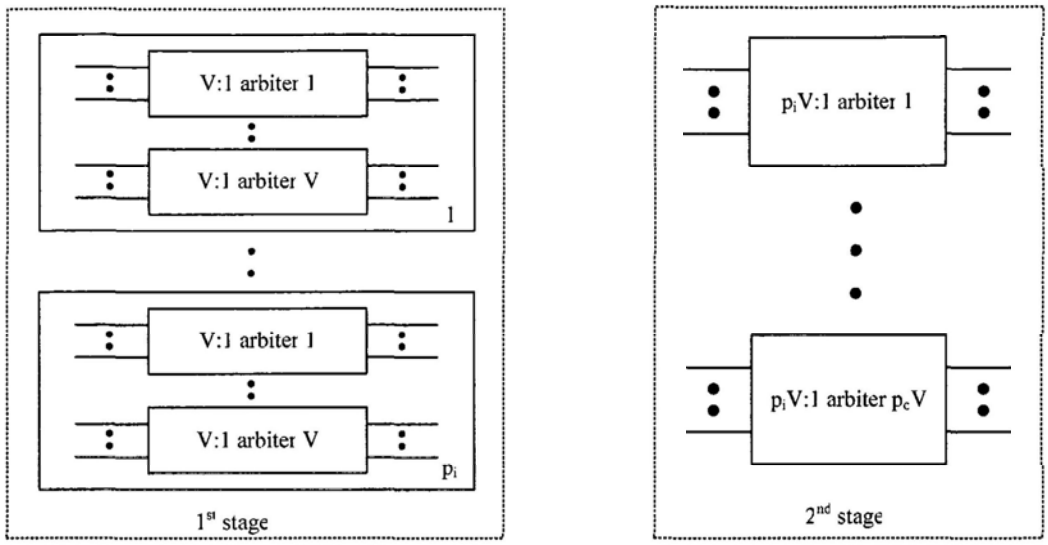


Figure 2.2. Complexity of a VA given a routing function returns any candidate VCs of a single output PC.

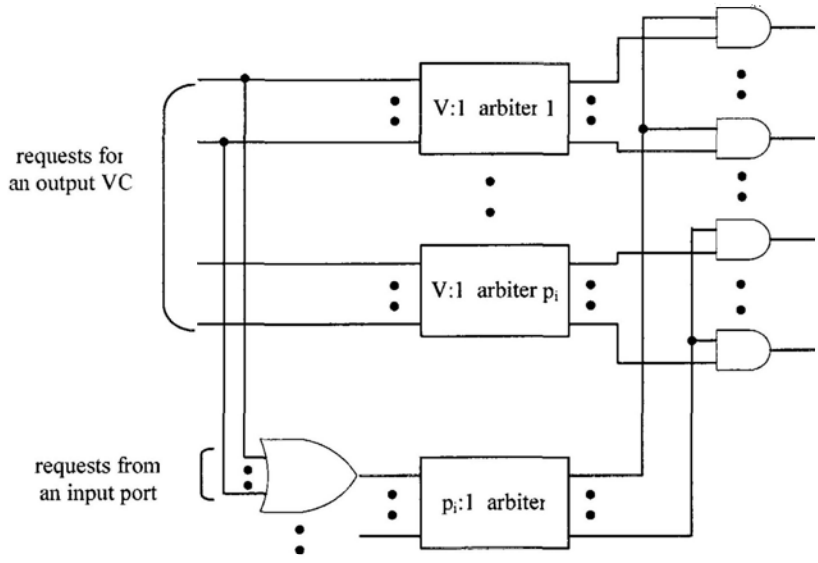


Figure 2.3. Tree architecture of a large $p_iV:1$ arbiter.

2.2.3 The generic switch allocator

A SA allocates crossbar bandwidth to input VCs. The generic architecture of a SA is designed in two stages as well (Figure 2.4). The first stage reflects sharing of a single crossbar input port by V input VCs. This requires a *V:1* arbiter for each input port. The second stage arbitrates among winning requests from p_i crossbar input ports for a crossbar output port. It needs a p_i*:1* arbiter at each output port.

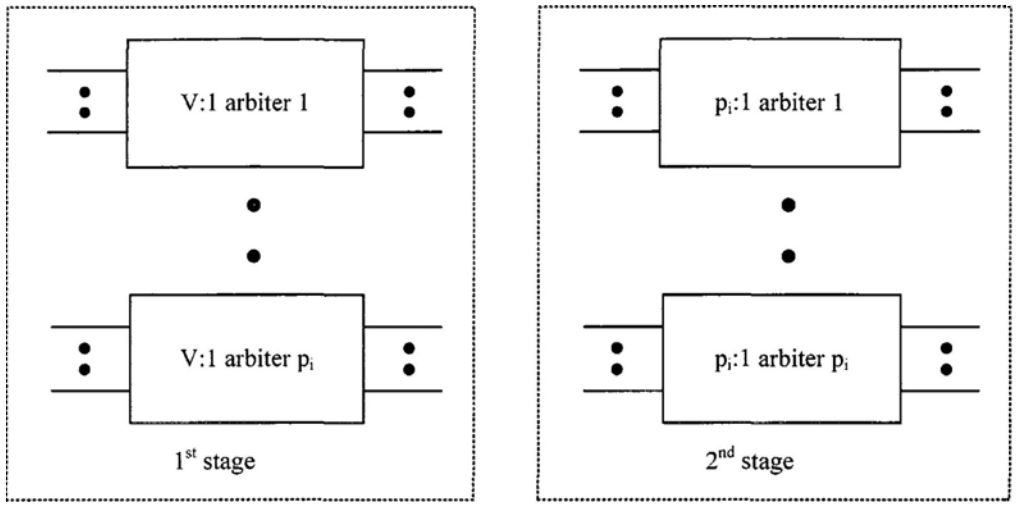


Figure 2.4. Complexity of the generic SA.

2.2.4 Motivations

Table 2.1 shows the proportions of area and power[2] demanded for the allocation logic, including the generic VA, the generic SA and associated logics for different numbers of ports and VCs in a router. Other router parameters are stated in Table 2.7. Area was taken from Synopsys [40] DC synthesis report under worse case conditions. Power was obtained from Synopsys PrimeTime PX with UMC 130nm library files, post-synthesis netlist, wire load model, and post-synthesis switching activities as inputs. The switching activities of $router_{30}$ (a 3-port router. We define the left-bottom router as $router_{00}$ for the 4×4 mesh throughout the paper.), $router_{31}$ (a 4-port router), and $router_{21}$ (a 5-port router) were derived from simulations of an complete NoCs assuming uniform traffic running at saturation points[3]. It can be seen that the allocation logic consumes significant amounts of area and power for all cases.

[2] Unless otherwise stated, power results in the thesis include both leakage power and dynamic power.

[3] Unless otherwise stated, results in the chapter are obtained when router radix is 5, the number of VCs is 4, packet length is 4, and network runs at the saturation point of uniform traffic at 250MHz.

Furthermore, the allocation logic cost proportionally increases with the number of ports or VCs. Therefore, it is important to reduce this cost.

Obviously, a VA is much more costly than a SA because it includes a large number of big ($p_iV:1$) arbiters. To our credit, it is possible to totally remove all arbiters in a VA and to make a VA and a SA share the same arbiters, the one shown in Figure 2.4. The sharing is identified after a careful study of the working principle of a VA and utilization statistics of VA arbiters. This will be explained in detail in the next section.

Table 2.1. COSTS PERCENTAGES OF ALLOCATION LOGIC IN VC ROUTERS

Router parameters	Area	Power
p_i =3, V =4	26.91%	28.06%
p_i =4, V =4	34.64%	30.79%
p_i =5, V =2	21.62%	20.81%
p_i =5, V =4	41.36%	32.65%
p_i =5, V =6	57.27%	37.28%

2.3 SIMPLIFICATION OF A GEINECRIC VA

2.3.1 Representations

Many parameters are used in this chapter. They are defined in Table 2.2 for reference.

Table 2.2. PARAMETER LIST

Parameter	Description
r_{mn}^{kl}	The request from the n^{th} VC at the m^{th} input port to the l^{th} VC at the k^{th} output port. The value is 1 if the request is valid. Otherwise, the value is 0.
r_{mn}^{k}	The request from the n^{th} input VC at the m^{th} input port to any VC at the k^{th} output port. The value is 1 if the request is valid. Otherwise, the value is 0.
r^{kl}	Whether there are valid requests to the l^{th} VC of the k^{th} output port.

	It is 0 when $(\sum_{m=1}^{p_i}\sum_{n=1}^{V} r_{mn}^{kl}) = 0$. Otherwise, it is 1.
r^k	The number of simultaneously requested VCs at the k^{th} output port $(\sum_{l=1}^{V} r^{kl})$.
r_m^k	Whether there are valid requests from any VCs at the m^{th} input port to any VC at the k^{th} output port. It is 0 when $\sum_{n=1}^{V} r_{mn}^{k} = 0$. Otherwise, it is 1.
r_m	The number of output ports where a VC is requested by any VCs in the m^{th} input port $(\sum_{k=1}^{p_o} r_m^k)$.
$pb1^k$	The probability that one VC at the k^{th} output port is requested.
pb^k	The probability that multi VCs at the k^{th} output port are concurrently requested.
$pb1_m$	The probability that VCs in the m^{th} input port request VCs at one output ports.
pb_m	The probability that VCs in the m^{th} input port request VCs at multi output ports.

Let us start simplification of a generic VA from its second stage that is the most costly. The second stage of a generic VA allocates p_oV output VCs to p_iV input VCs. Design complexity of the second stage can be represented as a $p_iV \times p_oV$ request matrix:

$$R = \begin{bmatrix} r_{11}^{11} & \cdots & r_{11}^{1V} & \cdots & r_{11}^{p_o 1} & \cdots & r_{11}^{p_o V} \\ \vdots & & & & & & \vdots \\ r_{1V}^{11} & \cdots & r_{1V}^{1V} & \cdots & r_{1V}^{p_o 1} & \cdots & r_{1V}^{p_o V} \\ \vdots & & \vdots & & & & \vdots \\ \vdots & & & \ddots & & & \vdots \\ r_{p_i 1}^{11} & \cdots & r_{p_i 1}^{1V} & \cdots & r_{p_i 1}^{p_o 1} & \cdots & r_{p_i 1}^{p_o V} \\ \vdots & & & & & \ddots & \vdots \\ r_{p_i V}^{11} & \cdots & r_{p_i V}^{1V} & \cdots & r_{p_i V}^{p_o 1} & \cdots & r_{p_i V}^{p_o V} \end{bmatrix}$$

Where, each row represents requests from an input VC while each column represents requests to an output VC. *Since an input VC is not allowed to simultaneously request*

multi output VCs, there is at most one valid request in each row. As a result, a row does not require an arbiter. However, there are at most p_iV requests to an output VC, so a column requires a $p_iV:1$ arbiter.

2.3.2 Changing the output-VC-selection function

When an output port returns any output VCs in the output-VC-selection function[4], at most V output VCs in this output port may be requested simultaneously. Thus, V *piV:1* arbiters are assigned to the output port with each one handling requests to one output VC. *However, if we change the output-VC-selection function so it returns at most one output VC of a single output port, there will be at most one requested output VC at the output port. As a result, only one piV:1 arbiter is needed for the output port*[5]*. Furthermore, since at most one output VC is returned to each input VC, the first stage of the generic VA is unnecessary and can be completely removed.* Nevertheless, returning only one output VC of a single output port decreases matching efficiency of the VA, which is illustrated by the following example.

Figure 2.6 shows the case that the output-VC-selection function returns any candidate output VCs of a single output port (We assume p_i and p_o are five, and V is four in all examples of this chapter unless otherwise stated.). We take the output port

---

[4] The term "routing function" is used in [39] since the function to return output VCs is performed in the RC pipeline stage. We also use this term in Section 2.2 in order to be consistent to the reference paper. However, it has a drawback that once an idle output VC is returned to an input VC, the input VC can not change the output VC even when the output VC is allocated to another input VC in the VA pipeline stage. Hence, the input VC has to wait for the allocated output VC to be released. Therefore, we change to run this function in the VA pipeline stage and thus use the new term "output-VC-selection function" from here on to avoid confusion.

[5] It is possible that multi output VCs of a single output port are requested simultaneously if the function to return at most one output VC of the output port is done in the RC stage. This is why there are still V $p_iV:1$ arbiters for each output port in [39].

0 for example. Figure 2.6 (a) shows the VA requests. There are only two valid requests, one to output VC0 and the other to output VC1. In the generic VA, a *20:1* arbiter is assigned to each output VC. Thus, the request to output VC 0 and the request to output VC 1 enter the corresponding *20:1* arbiters respectively. Both requests will succeed in the arbitrations because all the other requests to the arbiters are invalid. As a result, *two input VCs will be successfully allocated output VCs* and can enter the SA pipeline stage in the next clock cycle. Figure 2.6 demonstrates the case that the output-VC-selection function returns at most one output VC of a single output port. In this case, at most one output VC of the output port 0 is requested at each clock cycle (Figure 2.6 (a)). *Thus, at most one input VC can be successfully allocated an output VC. Therefore, returning at most one output VC of a single output port sacrifices VA efficiency, and decrease of the efficiency depends on the probability (pb^k) that multi VCs at the output port are simultaneously requested.*

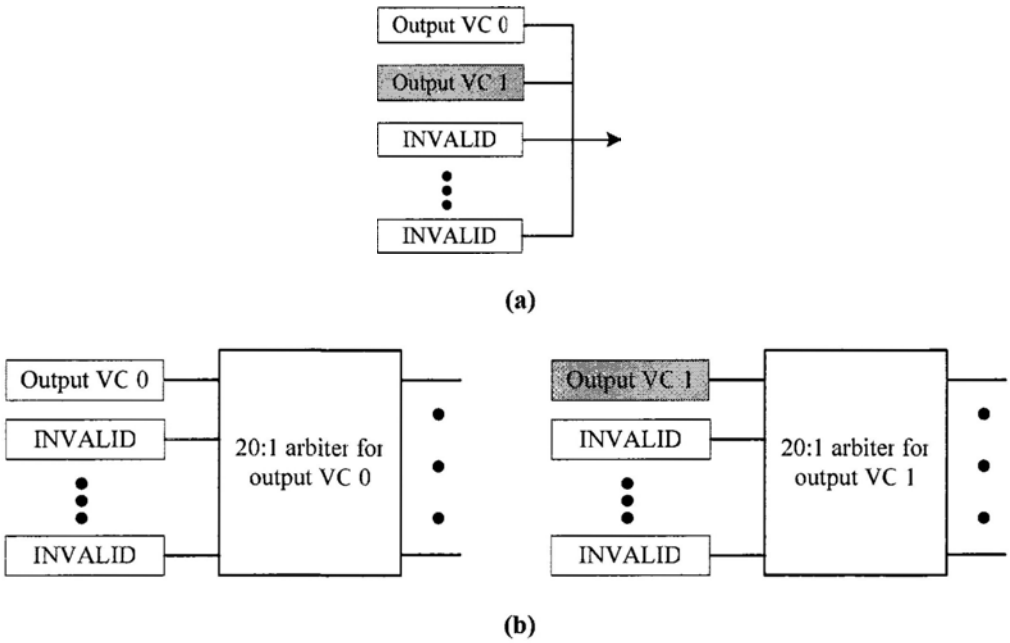


Figure 2.5. An example of the second stage of VA when any candidate output VCs of a single output port are returned. (a). VA requests to output port 0. (b). Assign a *20:1* arbiter to each output VC at output port 0. For clarity, arbiters for the other two output VCs are omitted.

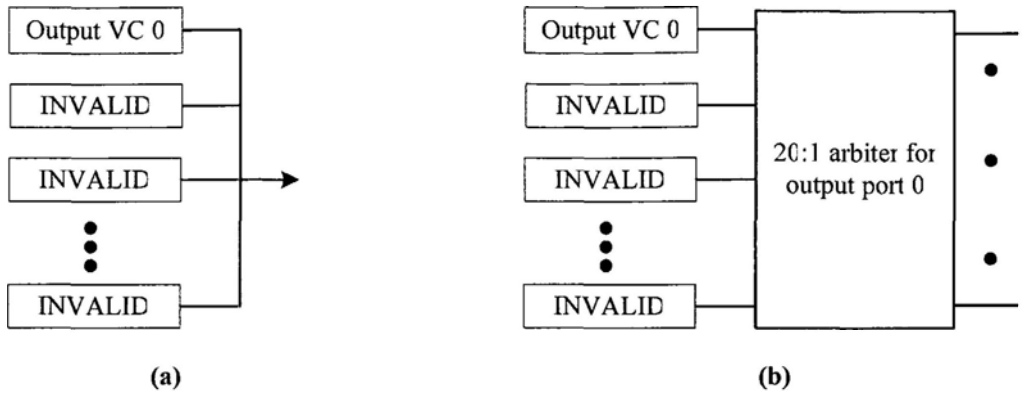


Figure 2.6. An example of the second stage of VA when at most one output VC of a single output port is returned. (a). VA requests to output port 0. (b). Assign a *20:1* arbiter to the output port 0.

We ran simulations for a 4×4 mesh NoCs under uniform, hotspot, and transpose traffic patterns and calculated pb^k by counting the number of cycles when r^k is 2 or above and then dividing by the number of simulation cycles. The results show that pb^k remains small for all routers for all injection rates. Hence, for each traffic pattern, only the pb^k for output PCs of the $router_{21}$ (we define the left-bottom router as $router_{00}$ for the 4×4 mesh throughout this chapter.) at saturation point (the saturation points are at 0.652, 0.603, and 0.248 flits/(node*cycle) for uniform, hotspot, and transpose traffic patterns respectively.) is shown (Table 2.3). It can be seen that all pb^k are smaller than 0.50% for all tested traffics. In other words, in each output PC, at most one VC is requested in more than 99.50% of all cases. The key reason for small pb^k results is that pb^k represents the probability that two or more VCs in an output port are simultaneously requested. To validate this claim, we also calculated $pb1^k$ for the three traffic patterns (Table 2.4). We can see that results of $pb1^k$ are much larger than results of pb^k. Thus, changing the output-VC-selection function from returning any candidate output VCs to retuning at most one output VC sacrifices the VA efficiency very tiny. After that, the second stage is simplified to a $p_iV \times p_o$ matrix. In addition, p_iV *V:1* arbiters of the first stage are removed and p_o output-VC-selection

blocks are added. Design of the output-VC-selection block will be described in Section 2.6.

$$R = \begin{bmatrix} r_{11}^{1} & \cdots & r_{11}^{p_o} \\ \vdots & & \vdots \\ r_{1V}^{1} & & r_{1V}^{p_o} \\ \vdots & & \vdots \\ r_{p_i1}^{1} & & r_{p_i1}^{p_o} \\ \vdots & & \vdots \\ r_{p_iV}^{1} & \cdots & r_{p_iV}^{p_o} \end{bmatrix}$$

Table 2.3. RESTULS OF pb^k

Traffic	Output PC				
	West	North	East	South	Local
Uniform	0.21%	0.23%	0.32%	0.33%	0.16%
Hotspot	0.23%	0.10%	0.30%	0.48%	0.30%
Transpose	0	0	0	0.01%	0

Table 2.4. RESTULS OF $pb1^k$

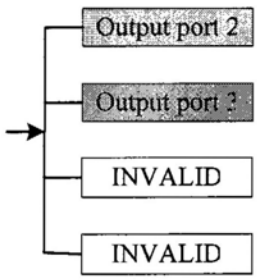
Traffic	Output PC				
	West	North	East	South	Local
Uniform	13.78%	12.43%	16.28%	17.53%	15.43%
Hotspot	12.68%	11.26%	16.78%	13.88%	13.41%
Transpose	0	7.90%	0	16.48%	0

2.3.3 Sharing of *V:1* arbiters at each input port

The VA after the above simplification now requires p_o $p_iV:1$ arbiters. This can be similarly organized as a tree architecture which has a set of p_o $V:1$ arbiters for every input port and one $p_i:1$ arbiter for each output port. This is also too generous because there are totally V VCs in an input port and many of them do not have VA requests most of the time. *Design costs will be much saved if we assign only one V:1 arbiter*

to an input port. However, it will sacrifice the VA efficiency as well, which is illustrated as below.

We assume there are two valid VA requests at input port 0 (Figure 2.7 (a)): one requests an output VC of output port 2 while the other requests an output VC of output port 3. The requests to different output ports are sent to different *4:1* arbiters in separated architecture (Figure 2.7 (b)) whereas they are sent to the same *4:1* arbiter in the shared architecture (Figure 2.7 (c)). In the separated architecture, both VA requests are certainly successful in the arbitrations. However, in the shared architecture, only one of them succeeds. If there are no VA requests to output port 3 from other input ports, an idle VC at output port 3 will be certainly assigned to the input VC (blue colour) at input port 0 in the separated architecture. However, this idle VC at output port 3 will be wasted at the current clock cycle if the request to output port 3 fails in the shared arbiter at input port 0. *In summary, decrease of the VA efficiency by sharing one arbiter across VA requests to various output ports is determined by the probability (pb_m) that input VCs at an input port request output VCs at different output ports.*



(a)

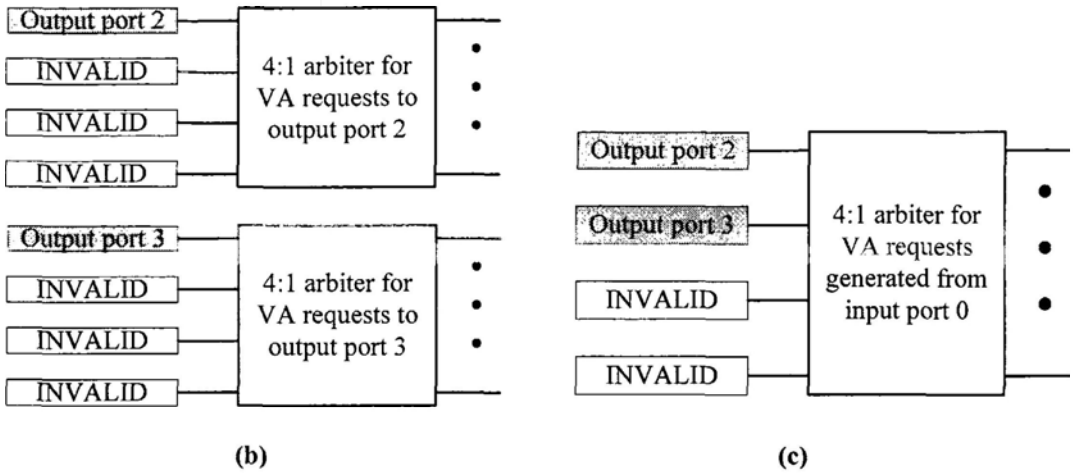


Figure 2.7. An example of sharing *V:1* arbiters at an input port. (a). VA requests generated at input port 0. (b). Assign five *4:1* arbiters for VA requests generated at input port 0. Each arbiter handles VA requests going to an output port. For brevity, arbiters serving requests to the other output ports are omitted. (c). Assign one *4:1* arbiter for all VA requests generated at input port 0.

From simulations as described previously, we calculated pb_m by counting the number of cycles when r_m is 2 or above and then dividing by the number of simulation cycles. Table 2.5 shows pb_m for input PCs of the router$_{21}$ at saturation points. The largest pb_m is 0.75% for uniform traffic and 0.28% for hotspot traffic. All the pb_m are zeros for transpose traffic (no packets enter the north, south, and local input PCs. Packets entering the west (east) input PC only go to the north (south) output PC.). Similar to pb^k, the key reason for small pb_m is that pb_m represents the probability that VCs in an input port request VCs at two or more output ports simultaneously. We can see that results of $pb1_m$ (Table 2.6) are significantly larger than results of pb_m. *Therefore, making all VCs of an input port to share one V:1 arbiter decreases VA efficiency by only a small amount. As a result, it is possible with negligible performance cost to reduce the number of V:1 arbiters at a single input port from p_o to 1.*

Table 2.5. RESTULS OF pb_m

Traffic	Input PC				
	West	North	East	South	Local

Uniform	0.20%	0.23%	0.75%	0.33%	0.15%
Hotspot	0.20%	0.15%	0.13%	0.15%	0.28%
Transpose	0	0	0	0	0

Table 2.6. RESTULS OF $pb1_m$

Traffic	Input PC				
	West	North	East	South	Local
Uniform	13.18%	15.91%	17.76%	16.91%	17.06%
Hotspot	11.01%	12.36%	15.31%	15.53%	20.91%
Transpose	7.99%	0	16.75%	0	0

2.3.4 Combining VA and SA arbiters

After the above two simplifications, a VA will consist of one *V:1* arbiter at each input port and one p_i*:1* arbiter at each output PC, which is obviously the same as the SA shown in Figure 2.4. Moreover, VA arbiters have the same functions as the corresponding SA arbiters: a *V:1* arbiter handles requests from VCs at an input port while a p_i*:1* arbiter deals with requests from various input ports to an output PC. The only difference is the type of requests (VA or SA requests). *Thus, a VA and a SA can share their arbiters if VA and SA requests are processed concurrently. This leads to a further 50% reduction of arbiters.*

Note that the concurrent processing means to process VA requests from some input VCs and to process SA requests from other input VCs in the same clock cycle. It is because an input VC can only be at one of three states, namely, making no request, making a VA request, and making a SA request. This is explained in Figure 2.8.

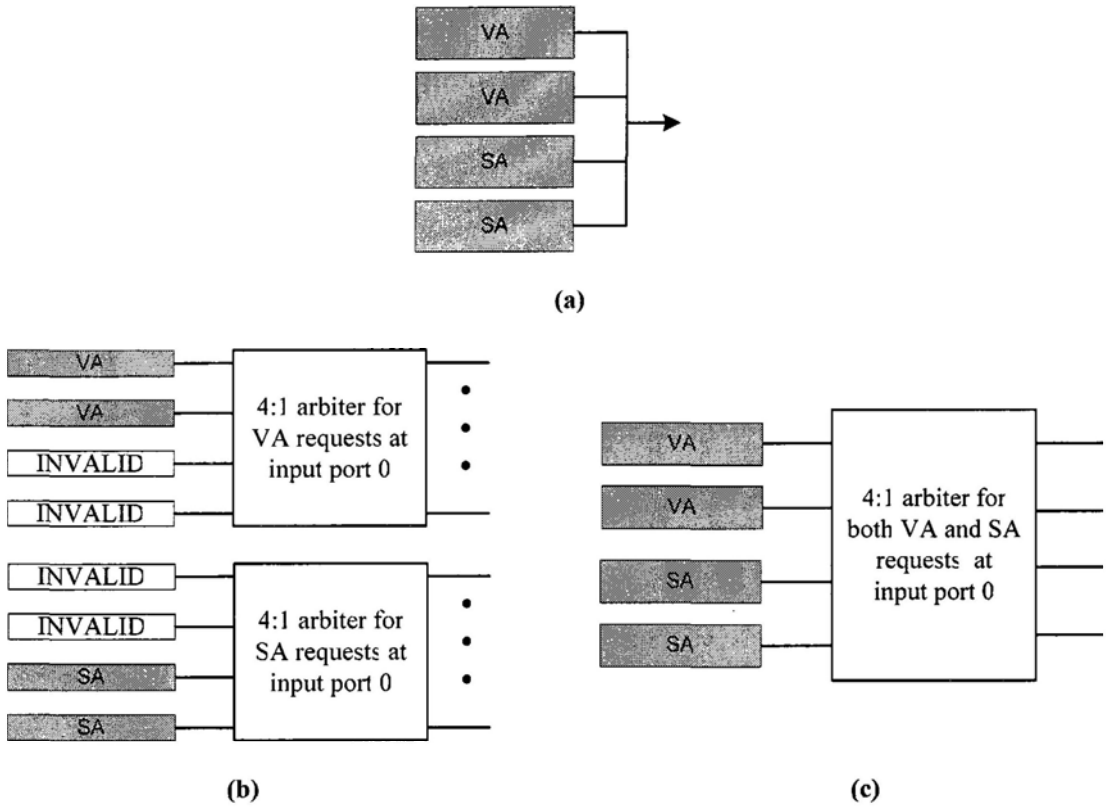


Figure 2.8. An example of sharing VA and SA arbiters at an input port. (a). VA and SA requests generated at input port 0. (b). Assign separated *4:1* arbiters for VA requests and SA requests, with one arbiter handling VA requests whereas the other arbiter handling SA requests. (c). Assign one *4:1* arbiter for both VA and SA requests.

More importantly, as shown in Figure 2.9, combining a VA and a SA and processing VA and SA requests simultaneously removes the VA pipeline stage for head flits, and thus reduces packet latency. Nevertheless, processing VA and SA requests concurrently may lead to deadlock and will be discussed in the next section.

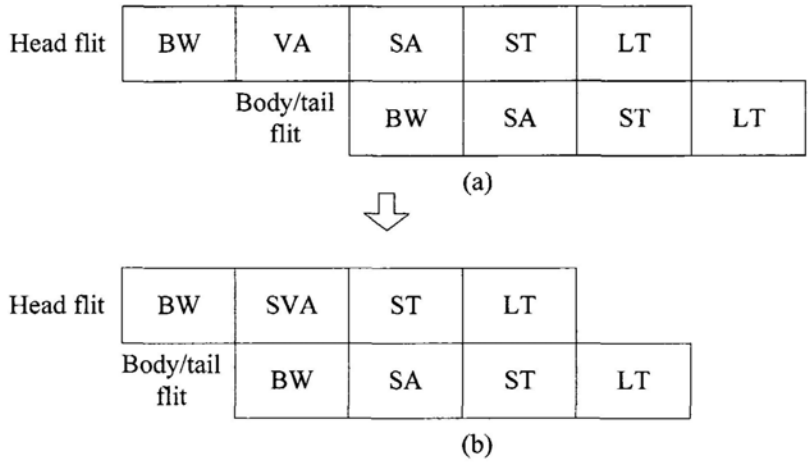


Figure 2.9. (a). Router pipeline when using separated VA and SA. (b). Router pipeline when using combined VA and SA.

2.4 DEADLOCK

2.4.1 Free output VC check

An input VC is successfully allocated an output VC when two requirements are met. One is that VA request of the input VC wins in the SVA, and the other is that there is a free output VC in the destined output PC. Free output VC check may be done in a speculative (Figure 2.10 (a)) or non-speculative (Figure 2.10 (b)) way. In the speculative architecture, we speculate that a VA request will successfully find a free output VC in the destined output port. Thus, the free output VC check is done in parallel with the SVA. In the non-speculative architecture, we must ensure that there is a free output VC for a VA request before it enters the SVA. Therefore, the free output VC check is done in series with the SVA. The speculative architecture removes the free output VC check from the critical path and is often selected when a generic VA is used. However, if the SVA is used, we found that the speculative architecture is not deadlock free and we have to use the non-speculative architecture.

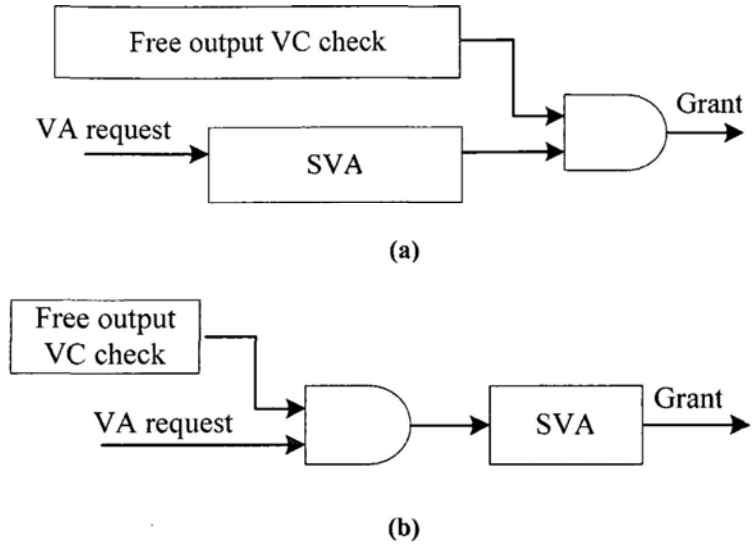


Figure 2.10. (a). A speculative architecture. (b). A non-speculative architecture.

2.4.2 Allocation and release of an output VC

If the head flit of a packet succeeds to be allocated an output VC in the VA stage, the output VC is held for the whole packet (or the input VC that buffers the packet) until the tail flit of this packet is successful in the SA stage and then releases the output VC. In other words, a logic connection between an input VC and an output VC is established by the VA operation for the head flit and released by the SA operation for the tail flit. The output VC can not be allocated to any other head flits until it is released. Figure 2.11 shows the timing diagram of output VC reallocation through a two-flit packet *A* followed by the head flit of packet *B* (assume the packet *B* requests the same output VC as the packet *A*.). At cycle 2, the head flit of the packet *A* succeeds in the VA pipeline stage and is allocated the output VC. At cycle 4, the output VC is released when the tail flit of the packet *A* succeeds in the SA stage. Although the BW stage for the head flit of the packet *B* is finished at cycle 3 and the VA stage for it can be performed at cycle 4, this head flit has to stall for one cycle, waiting for the packet *A* to release the output VC. Then, at cycle 5, the released output VC is reallocated to the head flit of the packet *B*. If the tail flit of the packet *A*

would never succeed in the SA, the output VC would be always held by the packet *A* and thus never be reallocated to the packet *B*.

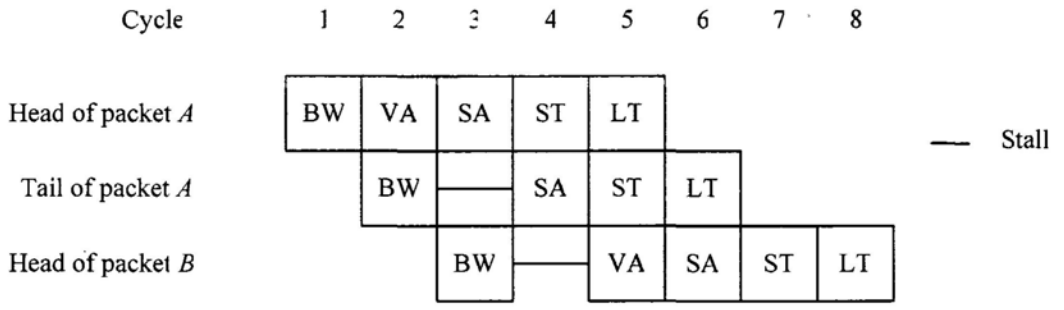


Figure 2.11. Timing diagram of reallocating an output VC.

2.4.3 Deadlock problem

After a VA and a SA is combined, there will be two groups of requests: the VA group and the SA group. Also, there will be two groups of resources: the output VC group and the priority group (In general, the round-robin algorithm is applied in both VA and SA arbiters to achieve fairness. A priority table is used to accomplish the round-robin algorithm by setting the priority of the request that is just served the lowest.). Requests and resources are related by hold and wait-for relations. As shown in Figure 2.12 (a), the VA group holds the priority group and waits for the output VC group. Similarly, the SA group holds the output VC group and waits for the priority group. If a request group holds a resource group, then that resource group is waiting on the request group to release it. Thus, each hold relation induces a wait-for relation in the opposite direction [12]. Redrawing the hold edges as wait-for edges in the opposite direction gives the graph of Figure 2.12 (b). The cycle in this graph shows that the architecture is deadlocked.

Let us illustrate the deadlock by a router in a mesh network (Figure 2.13). In the west input port, the VC0 has a VA request for any VCs in the east output PC while

the other three VCs have already held the VC1, VC2, and VC3 in the east output PC respectively. Similarly, in the north input port, the VC0 has a VA request for any VCs in the east output PC, the VC1 holds the VC0 in the east output PC, and the VC2 and VC3 occupy the VCs in other output PCs. In both the west and the north input ports, the VC0 has the highest priority. Therefore, in the speculative architecture, the SA requests at the two input ports always fail in the SVA, causing that four held VCs in the east output PC are no longer released. On the other hand, although the two VA requests always succeed in the SVA, they always fail to find a free output VC because all four VCs in the east output PC are always being occupied. As a result, they always keep the highest priorities in the corresponding input ports.

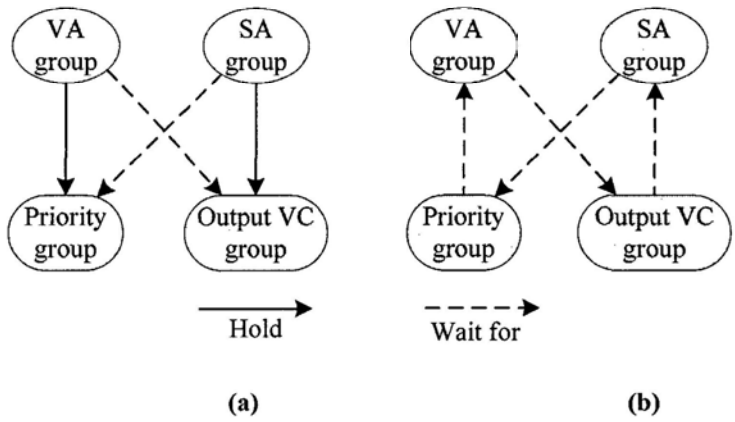


Figure 2.12. Hold and wait-for relationships.

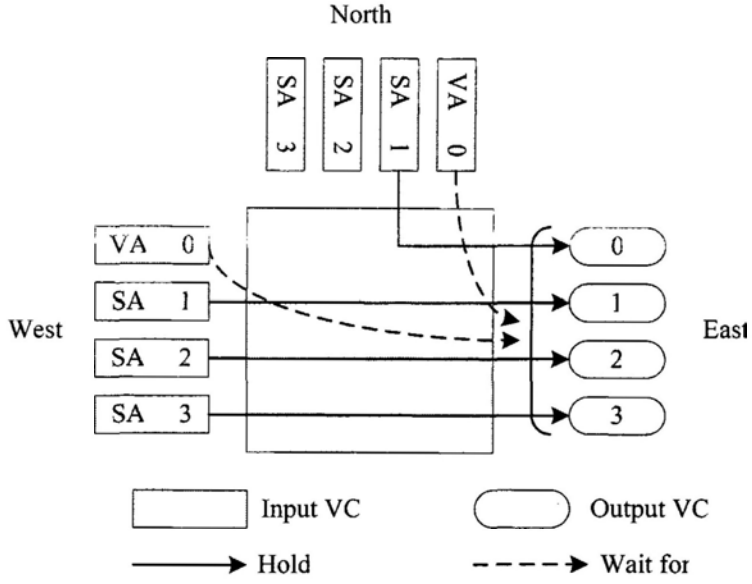


Figure 2.13. An example of the deadlock.

2.4.4 Solutions to deadlock

There are two approaches to deal with deadlock: deadlock recovery and deadlock avoidance. For the deadlock recovery (Figure 2.14), a counter is used to counting the number of consecutive cycles when a VA request succeeds in the arbitration but fails to find a free output VC. Once the deadlock is detected (The counter hits the pre-set threshold), the priority of this VA request will be set to the lowest to recover from the deadlock. The recovery approach works at low and moderate loads but leads to serious starvation problem at high loads.

Figure 2.15 illustrates the starvation problem at high loads. On the one hand, it is often that no free VC is found when the VA request has the highest priority (normally 3 or 4 cycles). On the other hand, it is usual that an output VC is allocated to some input VC as soon as the output VC is released. As a result, it is possible that the two requirements (highest priority and free output VC) can not be simultaneously met for the VA request, leading to the starvation problem.

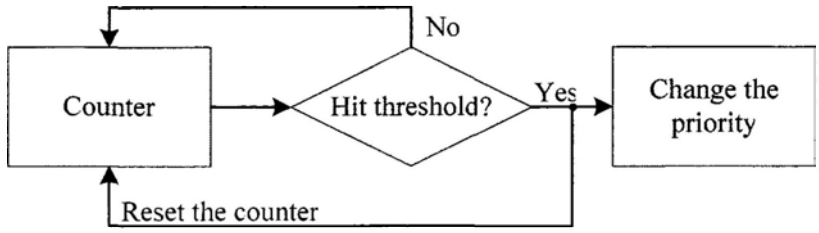


Figure 2.14. Deadlock recovery.

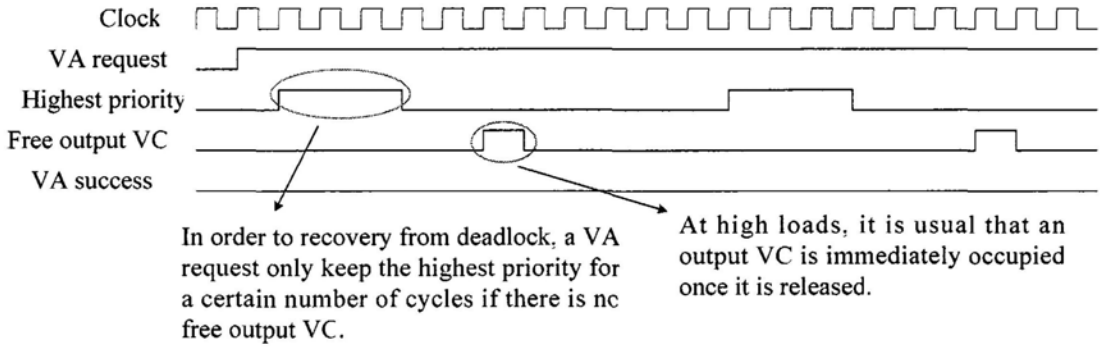


Figure 2.15. Starvation problem caused by deadlock recovery.

Since the deadlock recovery approach may result in the starvation problem, we use the deadlock avoidance method by breaking the cyclic dependence. In the non-speculative architecture, the two VA requests (Figure 2.13) are considered as invalid because there is no free output VC for them. As a result, they do not win in the SVA arbitration although they have the highest priorities. Instead, the four SA requests for the east output PC win the SVA in a round-robin way until one of them sends a tail flit and releases the output VC held by it. Then, at the next cycle, the two VA requests are all valid and one of them is allocated the released output VC. In summary, the VA group does not hold the priority group when there is no free output VC and thus breaks the cyclic dependence.

2.5 CRITICAL PATH ANALYSIS

2.5.1 Critical paths for the generic VA/SA

Figure 2.16 (a) shows the critical path for the generic VA. Firstly, an input VC checks whether there are free output VCs in the destined output PC. Then, it selects one from all free output VCs using the arbiter in the 1st stage. After that, a VA request is generated and sent to the 2nd stage arbiter for the selected output VC. Once the input VC is successfully allocated the selected output VC, status of this output VC is updated to be busy. Figure 2.16 (b) demonstrates the critical path for the generic SA. In the beginning, all 1st stage SA requests generated in the previous clock cycle enter the 1st stage arbiters for arbitration. Then, requests for the 2nd stage arbiters are generated and arbitrated. After that, the 1st stage requests for the next cycle are produced. The max frequency for a router with generic VA/SA architectures is determined by the critical path of the generic VA because it is longer than that of the generic SA.

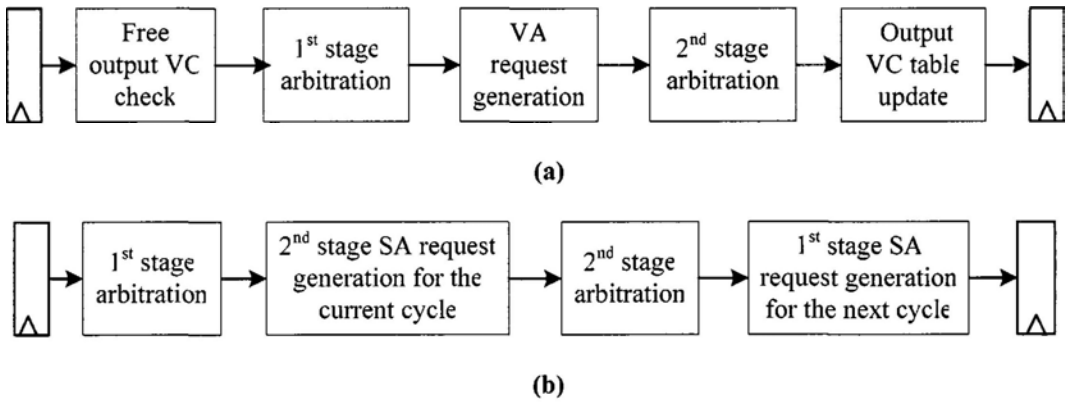


Figure 2.16. Critical path for the generic VA (a) and the generic SA (b).

2.5.2 VA simplification effects on critical paths

As described in Section 2.3, there are three methods for VA simplification: changing the output-VC-selection function, sharing of *V:1* arbiters, and combining

VA and SA arbiters. In the following, we explain their effects on the VA/SA critical paths one by one.

Firstly, reducing the number of $p_iV:1$ arbiters from V to 1 in an output PC does not produce additional delay for the VA because no logic is needed to detect conflicts. After changing the output-VC-selection function to return a single free VC of an output PC, at most one VC in an output PC will be requested at each cycle. Thus, one $p_iV:1$ arbiter is enough and no additional logic is required to determine which VC is to use the single $p_iV:1$ arbiter. On the contrary, the critical path of the VA can be reduced in two aspects. One is that the 1st arbitration stage is removed. The other is that logics for VA request generation are simplified because the number of $p_iV:1$ arbiters is largely reduced.

Secondly, sharing $V:1$ arbiters in an input port increases the critical path of the VA. Before sharing, a $p_iV:1$ arbiter in the 2nd arbitration stage of the VA is implemented as the tree architecture shown in Figure 2.3 where the $V:1$ arbiter and the $p_i:1$ arbiter are in parallel. After the sharing, a $p_iV:1$ arbiter is realized as the architecture shown in Figure 2.4 where the $V:1$ arbiter and the $p_i:1$ arbiter are in serial. Meanwhile, additional logics after the $V:1$ arbiter are required to generate requests for the $p_i:1$ arbiter.

Finally, combining VA and SA arbiters increases the critical path of the SA. Before the combination, the 1st stage SA requests directly enter the 1st stage arbiters for arbitration. After the combination, the 1st stage SA requests have to wait for the results of the free output VC check block before they enter the 1st stage arbiters in order to avoid the deadlock problem described in Section 2.4.

In summary, total effects on the VA/SA critical paths depend on which simplification methods are applied.

2.6 EVALUATIONS

2.6.1 Design parameters

In summary, simplification of a generic VA includes three steps: 1). Change the output-VC-selection function, then reduce the number of $p_iV:1$ arbiters and remove the first stage of the generic VA (Section 2.3.2). 2). Share the $V:1$ arbiters at each input port (Section 2.3.3). 3). Combine the generic VA and the generic SA arbiters together (Section 2.3.4).

We evaluated three allocation architectures: a generic VA and a generic SA (the generic), a look-ahead VA and a generic SA (the look-ahead), and a combined VA and SA (the SVA) by simulations for the entire NoCs instead of the allocation components themselves. In the look-ahead VA, only the first simplification step is performed. The output-VC-selection block at an output port works as follows. First, no output VC is selected if all VCs of the output port are busy. Second, if some output VCs are idle and empty, select the first one of them. Third, if some output VCs are idle but none of them is empty, select the idle VC that has the most number of free buffer spaces. In the SVA, all the three simplification steps are performed. All other components of the NoCs are the same in the three architectures studied. The network and process parameters are shown in Table 2.7.

Table 2.7. NETWORK AND PROCESS PARAMETERS

Traffic	Uniform / Hotspot[6] / Transpose
Topology	4×4 mesh
Flow control	Virtual channel
Routing	XY
Buffer management	Credit-based
Pipeline	Generic VC pipeline[7]
Router radix	3/4/5
Buffer architecture	2/4/6 VCs per port, 4 flits per VC
Packet length	4/8/16 flits
Flit size	32 (random payload) + 4 (overhead)
Technology	130nm, HS
Frequency	250MHz

2.6.2 Network performances

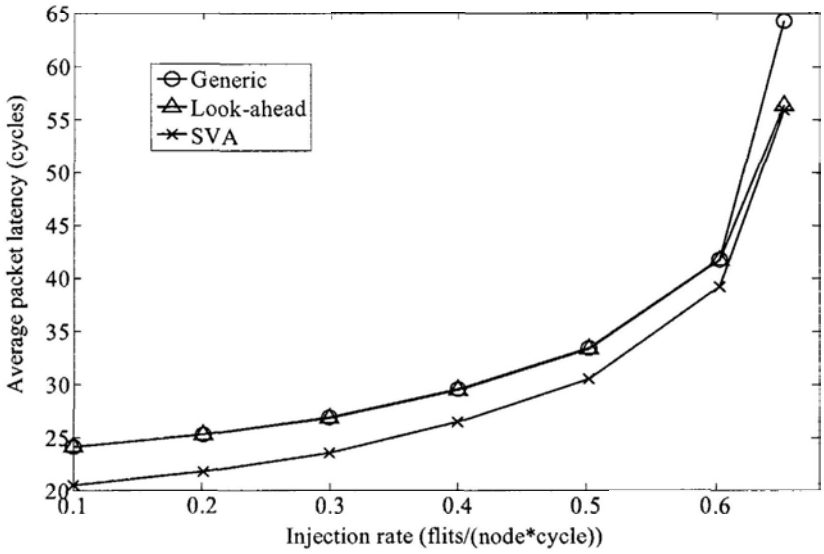
Network performances were obtained by a simulator modelled in SystemVerilog. Evaluations were performed for various network sizes, various Vs, various packet lengths, and a range of traffic patterns to validate whether the conclusion that the VA simplification presented in section IV has small effect on network performances is general. Saturation is defined as the highest level of injection rate for which the average throughput equals to the injection rate [12]. We only compared latencies before saturation.

Average packet latency as a function of traffic injection rate is plotted for different traffic patterns in Figure 2.17. The curve of the generic architecture nearly overlaps that of the look-ahead architecture for all tested traffic patterns. It means that the pb^k

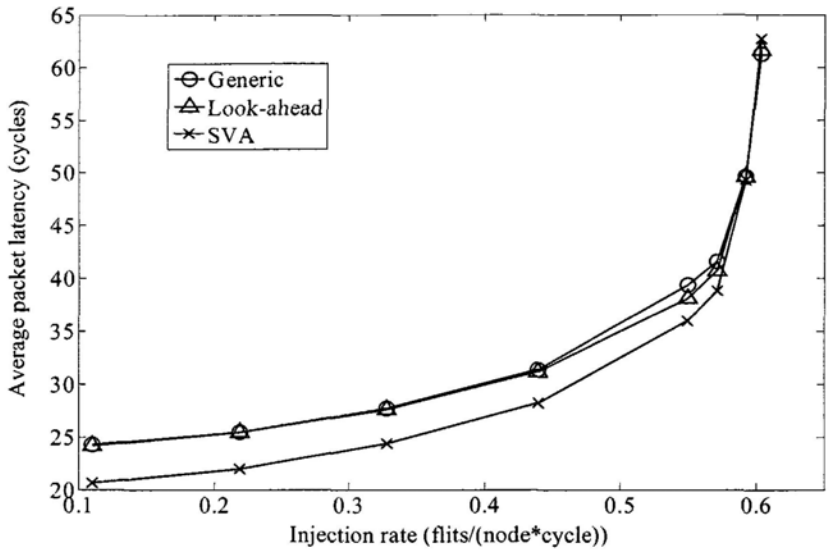
[6] The hotspot routers are $router_{11}$, $router_{22}$, $router_{31}$. They inject packets to the network with a 1.5x rate.

[7] Separated VA and SA stages for the generic and the look-ahead architectures whereas combined VA and SA stages for the SVA architecture.

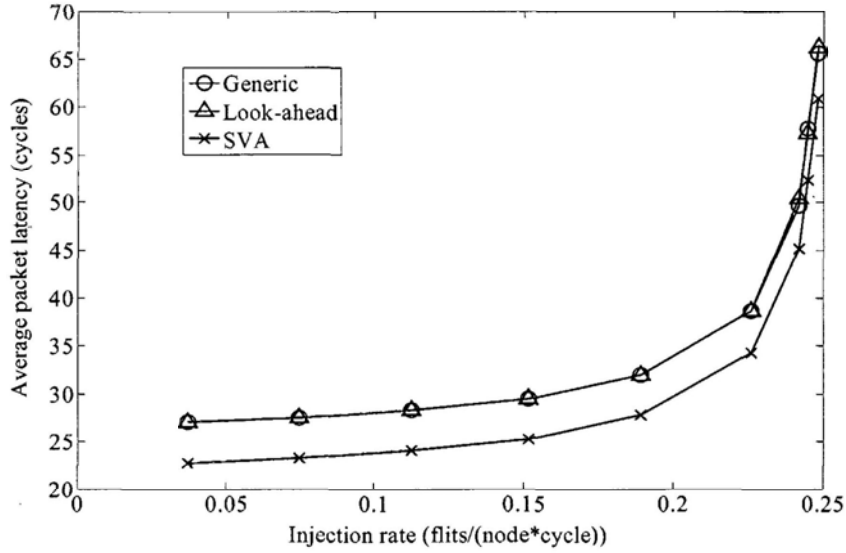
remains small and arbiter reduction has little impact on network latencies for different traffic patterns. The latency of the SVA is significantly smaller than the other two architectures at low and moderate network loads and becomes almost the same at high network loads. Figure 2.18 shows results when packet length is 8 and 16 flits respectively. We can see that reduction of latency by the SVA remains significant at high injection rates. Figure 2.19 describes results for 6×6 mesh. The trend is similar to that for 4×4 mesh. The SVA can reduce latency because it removes the VA pipeline stage for head flits. In addition, similar trends can be observed for various Vs (keeping the 4×4 network size, uniform traffic pattern and 4 flits per packet). The results for various Vs are not shown for clarity.



(a)

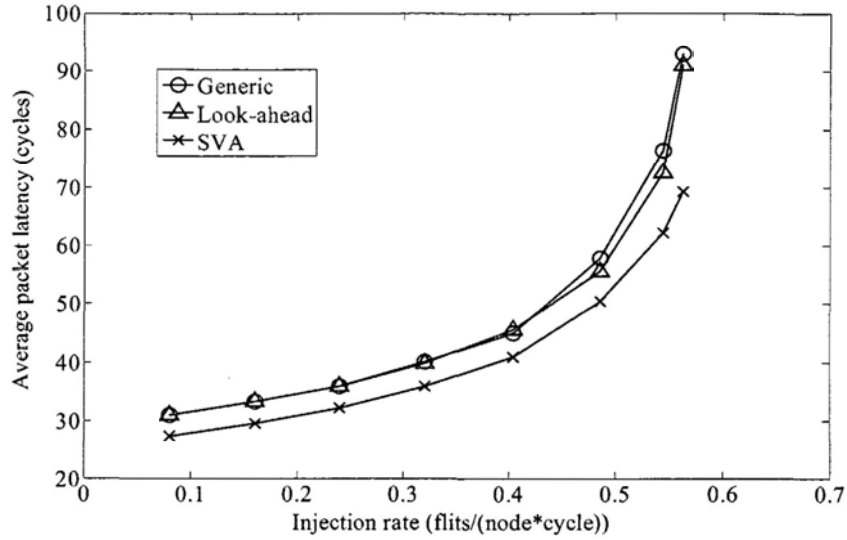


(b)

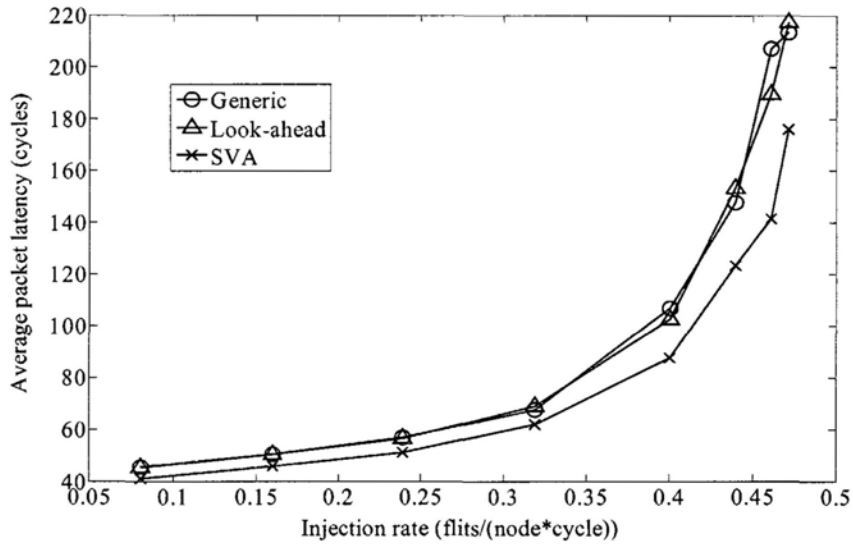


(c)

Figure 2.17. Average packet latency for various traffic patterns when network size is 4×4, V and packet length are 4. (a). Uniform (b). Hotspot. (c). Transpose.



(a)



(b)

Figure 2.18. Average packet latency for other packet lengths when network size is 4×4, V is 4, and traffic pattern is uniform. (a) 8flits. (b) 16 flits.

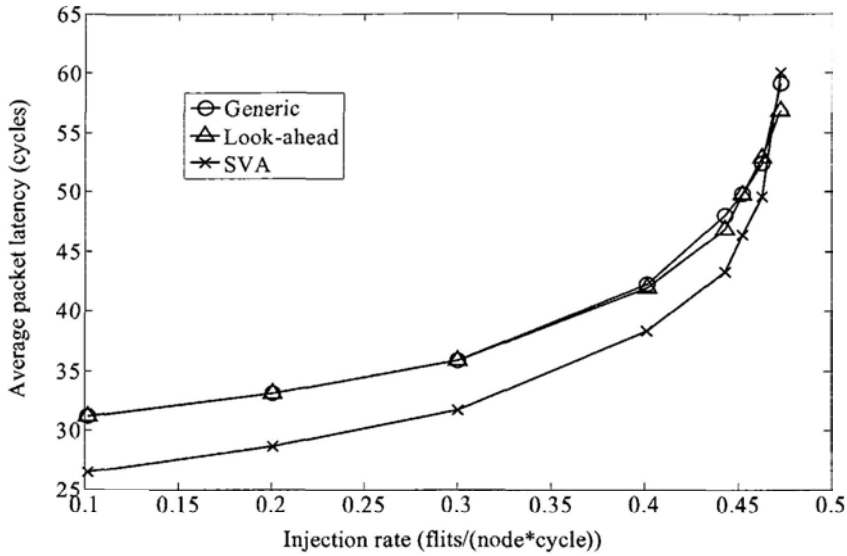


Figure 2.19. Average packet latency for 6×6 mesh when V is 4, packet length is 4 and traffic pattern is uniform.

2.6.3 Maximum router frequency

Table 2.8 summarizes maximum router frequencies for the three allocation architectures. The frequencies were obtained from Synopsys DC with worst case synthesis condition. The generic architecture and the SVA have similar frequencies while the look-ahead architecture has higher frequencies than the others. The reason is that only the simplification method of changing the output-VC-selection function is used in the look-ahead architecture while all three simplification means are applied in the SVA. As presented in Section 2.5, the *method 1* reduces the critical path delay whereas sharing of *V:1* arbiters (*method 2*) and combing VA and SA arbiters (*method 3*) increase the critical path delay. Therefore, router speed for the look-ahead architecture is always the highest while the speed for the SVA depends on the total effects of all the three methods. For example, when p_i is three and V is four, the SVA is faster than the generic because the delay reduction caused by *method 1* is larger than the delay increase caused by *method 2* and *method 3*. When p_i is four and V is

four, the SVA is slower than the generic because the delay reduction is smaller than the delay increase.

Table 2.8. MAX ROUTER FREQUENCY (MHz)

Router parameters	Generic	Look-ahead	SVA
p_i =3, V =4	427	526	442
p_i =4, V =4	403	476	388
p_i =5, V =2	435	526	435
p_i =5, V =4	385	435	385
p_i =5, V =6	323	385	333

2.6.4 Area and power costs at a certain frequency

Table 2.9 shows area costs of the three allocator architectures at the frequency of 250MHz. The look-ahead architecture reduces area by decreasing the number of $p_iV:1$ arbiters from p_oV to p_o and removing all the first stage arbiters of a generic VA. The SVA reduces even more area through sharing $V:1$ arbiters at each input port for a generic VA and combining arbiters of a generic VA and arbiters of a generic SA. Higher proportion of area is reduced as p_i or V increases. For a 5-port, 4-VC router, the look-ahead architecture and the SVA reduce allocator area by 57.17% and 68.43% respectively. They reduce router area by 23.65% and 28.30% respectively because the generic architecture consumes 41.36% area in the router.

Table 2.9. AREA (GATE COUNT) OF THE THREE ALLOCATORS

Router parameters	Generic	Look-ahead	SVA
p_i =3, V =4	5355	2525	2123
p_i =4, V =4	10433	4586	3785
p_i =5, V =2	3697	2973	2437
p_i =5, V =4	17676	7570	5581
p_i =5, V =6	49314	13581	9873

In the same way as described in Section 2.2, we calculated power consumption at 250MHz for many injection rates of uniform traffic because power is highly related

to network loads. Table 2.10 demonstrates power consumed by the three allocator architectures at zero-load and saturated-load. The look-ahead architecture reduces power at both zero-load and saturated-load for all tested parameters. The SVA has the smallest power at zero-load for all cases because it has the lowest logic area. However, power consumption of the SVA at saturated-load surpasses that of the look-ahead architecture for the three 5-port cases and even surpasses that of the generic architecture for the 5-port, 2-VC case. The reason is as follows. On the one hand, the SVA reduces logic gates and thus reduces power consumption (positive effect). On the other hand, the sharing of logics in the SVA associates more logics together, and thus makes it possible that a logic transition of a net will lead to logic transitions of more nets. As a result, it increases average logic transition and therefore increases power consumption (negative effect). For most cases, the positive effect is larger and thus the SVA consumes smaller power consumption. For other cases, the negative effect is larger and thus the SVA consumes larger power consumption.

Power versus injection rate for the 5-port, 4-VC case is plotted in Figure 2.20. It can be seen that the difference in power between the look-ahead architecture and the SVA is small. On average, the look-ahead architecture and the SVA save allocator power by 27.18% and 30.78% respectively. Considering that the generic allocators consume 32.65% power in the router, the look-ahead architecture and the SVA reduce power by 8.87% and 10.05% respectively for the router. The router power savings increases to 16.44% (the look-ahead) and 17.07% (the SVA) for the 5-port, 6-VC router.

Table 2.10. POWER (mW) OF THE THREE ALLOCATORS (ZERO-LOAD | SATURATED-LOAD)

Router parameters	Generic	Look-ahead	SVA

p_i =3, V =4	0.54 | 2.13	0.39 | 1.65	0.32 | 1.58
p_i =4, V =4	0.67 | 3.51	0.46 | 2.67	0.36 | 2.64
p_i =5, V =2	0.45 | 1.44	0.43 | 1.44	0.34 | 1.61
p_i =5, V =4	0.81 | 4.93	0.52 | 3.80	0.41 | 3.96
p_i =5, V =6	1.20 | 8.65	0.56 | 5.23	0.44 | 5.46

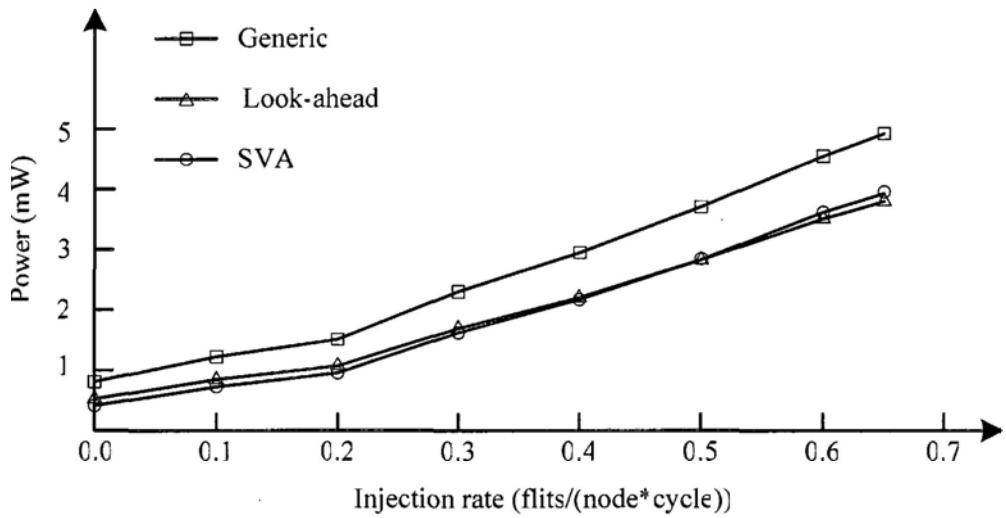


Figure 2.20. Power of the allocators at various injection rates

2.6.5 Discussion

When networks run at frequencies which can not be met by the SVA, the look-ahead architecture is the only reasonable choice. Otherwise, if networks run at frequencies which can be achieved by the SVA, the SVA is better because it provides lower packet latency (in cycles), as well as lower area and power costs for most design cases.

2.7 SUMMARY

This chapter presents implementations of low-cost switch and VC allocators. Instead of studying the allocators in isolation, we study them in the context of the entire NoCs. Opportunities to reduce design costs are identified for the generic architectures through analyses and statistics. As a result, three simplification methods

including changing the output-VC-selection function, sharing V:1 arbiters for the VA at each input port and combining VA and SA arbiters are described and two low-cost allocators are proposed.

Sharing of arbiters by the VA and the SA makes VA requests and SA requests dependent on each other, possibly leading to a deadlock problem. Checking free output VC after VA requests entering arbiters is not deadlock free. The deadlock recovery scheme may lead to starvation problem. Thus, the deadlock avoidance scheme is used by checking free output VC before VA requests entering arbiters.

In addition, the three VA simplification methods affect the critical paths of the VA and SA pipeline stages. The first method to change the output-VC-selection function to return a single output VC from an output port reduces the critical path of the VA. However, the sharing of V:1 arbiters in an input port increases the critical path of the VA, and the sharing of VA and SA arbiters increases the critical path of the SA. Overall effects on the critical paths depend on which methods are used.

We did comprehensive evaluations for the allocators, including network-level performances, frequency, area, and power. Results show that both the look-ahead architecture and the SVA architecture achieve lower costs than the generic architecture without any adverse effects on network performances. The look-ahead architecture and the SVA have different application domains that are determined by the frequency constraint.

This chapter only addresses the separable iSLip allocator [41]. However, there are other advanced allocator schemes like separable lonely output allocator and wavefront allocator that have higher matching efficiency and are used in off-chip

networks [42, 43]. Thus, it is interesting to study these advanced allocators and reduce their design costs for NoCs.

In addition, this chapter shows great effectiveness to study router components in the context of the complete NoCs rather than the components themselves. This is a good research method that can be used to explore other components of a router.

CHAPTER 3. POWER-EFFICIENT EVC INSERTION METHODOLOGY

3.1 INTRODUCTION

Total router energy consumption when a flit travels from a source to a destination can be denoted as $E_{flit} = E_{router} \times H$, where E_{router} is average energy consumed by a router and H is the hop count. Thus, there are two directions to reduce the total router energy. One is to reduce E_{router} and the other is to reduce H. We work along the first direction to reduce E_{router} by designing cost-efficient virtual channel and switch allocators in Chapter 2 while work along the second direction to reduce H in this chapter.

The hop count is the number of routers that a flit traverses from the source router to the destination router. It is determined by network topology and routing algorithm. In a 2D mesh network, the hop count is equal to the Manhattan distance when deterministic routing is applied. It is a popular way to reduce the hop count of 2D mesh network through inserting some express physical channels to bypass intermediate routers, such as express cubes [37, 44, 45] and application-specific long links [17, 46]. However, inserting a new physical channel means adding a port for routers at both ends of the channel, which increases E_{router}. *Thus, the key is to reduce H while keeping overhead of E_{router} small.*

A new flow control mechanism, express virtual channel, was recently proposed to reduce H through virtually bypass intermediate routers [36]. The overhead of E_{router} is small because express channels are not built physically, but built virtually. The authors presented express pipelines, EVC router microarchitectures and evaluated EVC flow control by high-level models. However, they did not address EVC

insertion and assumed that EVC paths were added regularly. Thus, how to optimize the EVC insertion is still an open problem.

We optimize the EVC insertion with the main objective to reduce power consumption by exploiting communication characteristics of applications. We believe that emphasizing the role of communication characteristics increases the optimization room for EVCs insertion. This idea is based on two observations. Firstly, as the aggregate traffic load of a router pair[8] is generally different from that of another pair, they should be distinguished. Secondly, more power is saved if more traffic loads pass through it after an EVC path is inserted. Let's illustrate it by a 4×4 mesh with XY routing for transpose traffic (Figure 3.1)[9]. The normalized aggregate communication loads of router pairs demonstrate large variances, from zero to three. In the static EVC insertion, two EVC paths will be inserted. (In fact, totally sixteen EVC paths will be inserted. Only two of them are shown for clarity in this example). One is from router 00 to router 02, and another is from 02 to 00. However, the EVC path from 00 to 02 has definitely no power saving since its traffic load is zero. This bad EVC is added due to the insertion is done in a blind way. The EVC path from 02 to 00 has power saving of two units. On the contrary, in our AS-EVC insertion scheme, the router pair with the largest aggregate load (from 01 to 10) is found and an EVC path is inserted there, thereby leading to power reduction of three units. Clearly, a 1.5x power saving is obtained by only one smart EVC path compared to that by two static EVC paths.

---

[8] Note that a router pair is directed. Thus, the pair from r_i to r_j is different from the pair from r_j to r_i.

[9] For simplicity, the cost caused by the aggregate communication volume traveling an EVC source router is not considered in the illustration.

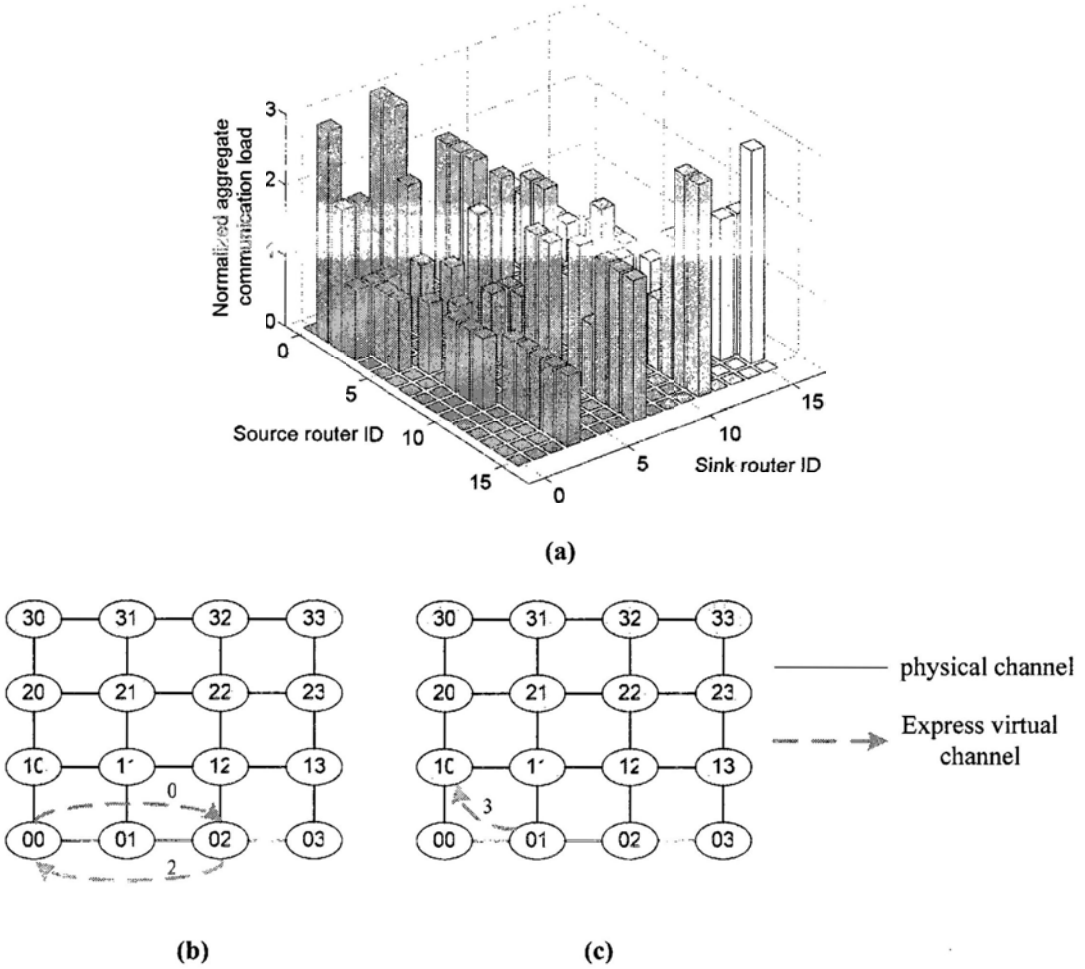


Figure 3.1. Illustrations of static EVC insertion and AS-EVC insertion. (a). Aggregate communication loads of router pairs. (b). An example of static EVC insertion. (c). An example of AS-EVC insertion

In order to further improve the efficiency of EVC insertion, we remove some limitations in the static EVC insertion. 1). EVC paths are not limited to be straight along X or Y dimension. Switch-dimension EVC paths can be inserted. 2). Two paths between r_i and r_j are considered separately. In this way, inserting an EVC path from r_i to r_j does not mean an EVC path will be inserted reversely from r_j to r_i. 3). A maximum interval instead of a fixed interval is set. The length of an EVC path can be any value smaller than the pre-set maximum interval. 4). EVC source and sink routers are allowed to be bypassed.

Our other contribution is to evaluate power consumptions through low-level VLSI implementations instead of high-level models. We performed evaluations for a wide range of traffic patterns including uniform, transpose, and TRIPS OCN traffics to show that the AS-EVC NoCs are generally better than static EVC and baseline NoCs. In addition, we performed evaluations for various network sizes such as 4×4 mesh, 6×6 mesh, and 10×4 mesh to show the AS-EVC method is scalable.

The structure of this chapter is as follows. Section 3.2 reviews related work. Following, Section 3.3 gives an overview of express virtual channel flow control. Section 3.4 presents application-specific EVC insertion methodology. Then, Section 3.5 demonstrates evaluations on power consumption. Finally, Section 3.6 concludes this thesis chapter.

3.2 RELATED WORK

Express cube was firstly proposed to improve network performances of k-ary n-cube off-chip interconnection networks that are node-limited [47]. The main idea of the express cube technology is to connect non-adjacent nodes with long express physical channels, so that long-distance traffic can mainly travel along the EPCs and skip intermediate routers. As a result, delay to traverse intermediate routers can be removed.

When packets travel along express physical channels, power consumed by intermediate routers can also be saved. Thus, Wang et al. in [37, 44, 45] applied the express cube for power and energy efficient NoCs designs. Express physical channels are regularly inserted in the express cube. Figure 3.2 (a) depicts an express cube based on a 5×5 mesh. Express physical channels are those channels that connect a subset of routers to their v-router away neighbors. v denotes the number of

routers spanned by an express physical channel and is two in the example. Totally twenty express physical channels are added.

Different from the express cube in which express physical channels are inserted in a regular way, Ogras et al. [17, 46] inserted express physical channels in an application-specific fashion with the main objective to reduce packet latency and improve network throughput. Communication volumes were used to calculate critical traffic values so that only few most beneficial express physical channels were inserted. Figure 3.2 (b) shows a 5×5 mesh with application-specific long links. Only three express physical channels are added.

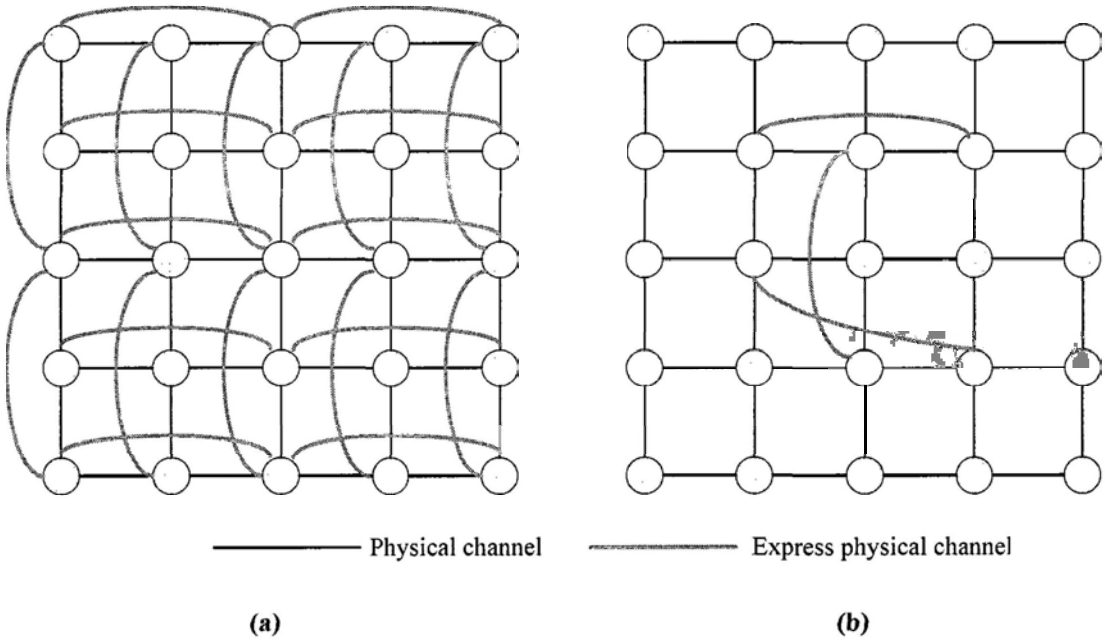


Figure 3.2. Bypass through express physical channels. (a). Express cube. (b). Application-specific long link.

After express physical channels are inserted, many packets will travel along them and skip intermediate routers. As a result, average hop count is reduced and energy consumption is saved. However, adding an express physical channel means to add a new port at both end routers of this channel (The two routers are called EPC router). Figure 3.3 compares microarchitectures of a non-EPC router and an EPC router. A

non-EPC router has five ports (west, north, east, south, and local) and a 5×5 crossbar. An EPC router with an EPC has six ports and a 6×6 crossbar. The added port (it requires extra buffer space and associated control logic) and the larger crossbar result in large overhead at the EPC router. FPGA prototypes in [46] show that 3-port, 4-port, 5-port, and 6-port routers utilize 219, 304, 397, and 503 slices respectively. This is to say, moving from 3-to-4, 4-to-5, and 5-to-6 increases the router area by 38.8%, 30.6%, and 26.7% in respective. Clearly, EPC routers consume more energy than non-EPC routers because of their area overhead, thus partially offset energy savings caused by reduction of average hop count.

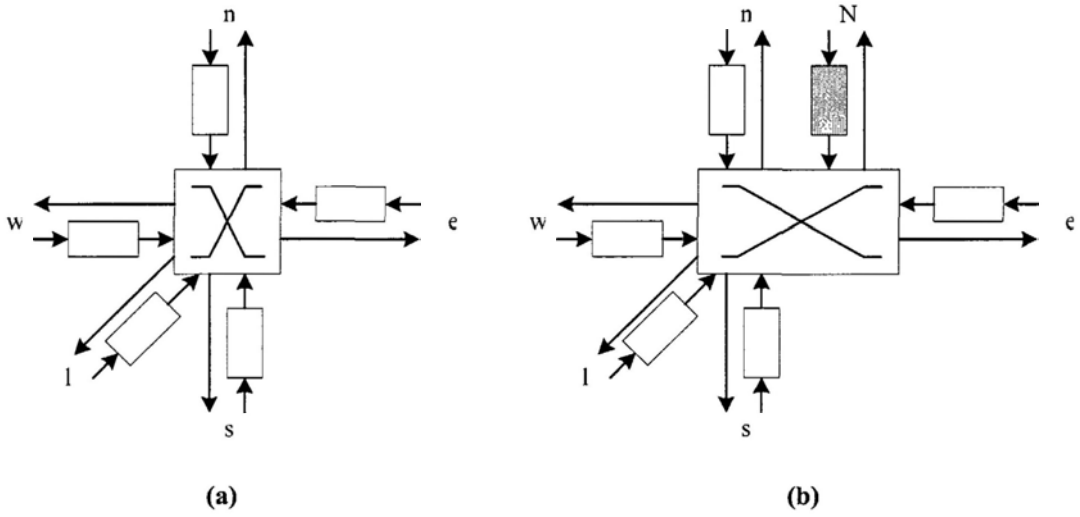


Figure 3.3. Router microarchitectures. (a). A non-EPC router (b). An EPC router with one EPC.

Low-power and *energy-efficient* are two sometimes confusing concepts. *Low-power* refers to absolute low power consumption, while *energy-efficient* pursues low energy cost per unit work. Although these two concepts are not necessarily mutually-exclusive, it is important to make clear which one is the optimization target. Wang et al. compared express cube and 2D torus using analytical power models [44]. When flit size remains unchanged, the express cube consumes larger power consumption than 2-D torus because of extra wires and complicated EPC routers. However, it can

sustain even higher throughput and the energy cost per flit is reduced when network size is large. Hence, the express cube is not a *low-power*, but an *energy-efficient* topology. Ogras et al. compared a 4×4 mesh and a 4×4 mesh with application-specific long links by FPGA prototypes in [46, 48]. The results show that the mesh with express physical channels is *energy-efficient*, but not *low-power*. Although an *energy-efficient* NoCs can be changed to a *low-power* NoCs through flit size reduction or frequency/voltage scaling techniques to reduce the high throughput to the just-meet-requirement value, these techniques have overheads. In our opinion, bypass through express physical channels, either the express cube or the application-specific long links, is suitable for *energy-efficient* NoCs designs, but not for *low-power* NoCs designs. In order to obtain *low-power* NoCs designs, we need a bypass technique that has only small router overhead when reducing average hop count. In the next section, we will introduce such a technique.

3.3 EXPRESS VIRTUAL CHANNEL FLOW CONTROL

Unlike the previous two bypass techniques (the express cube and the application-specific long links) in which intermediate routers are bypassed through express *physical* channels, express virtual channel is a new technique to bypass intermediate routers through express *virtual* channels. Both the previous techniques are topological technique because they change network topologies whereas EVC is a flow control technique where express virtual channels are built through smart control on flit flows. We have an overview of the EVC flow control in this section because our work in this and the next chapters are based on this technique. The details about it can be found in [36].

3.3.1 EVC router pipelines

Some special notations are used in the thesis. They are described as follows and illustrated in Figure 3.4.

- EVC source router. The router at which an EVC path originates. The corresponding output port is called an EVC source port.
- EVC sink router. The router at which an EVC path terminates. The corresponding input port is called an EVC sink port.
- EVC bypass router. The intermediate routers covered by an EVC path. The corresponding input/output port is called an EVC bypass input/output port.
- NVC lane. The VC lane which is allocated in a similar fashion as in the traditional VC flow control and is responsible for buffering packets through a single-hop physical channel.
- EVC lane. The VC lane which buffers packets travelling along an EVC path. Thus, only EVC sink routers have EVC lanes.

In addition, a flit is an EVC flit when it is travelling intermediate routers along an EVC path. Otherwise, it is a NVC flit.

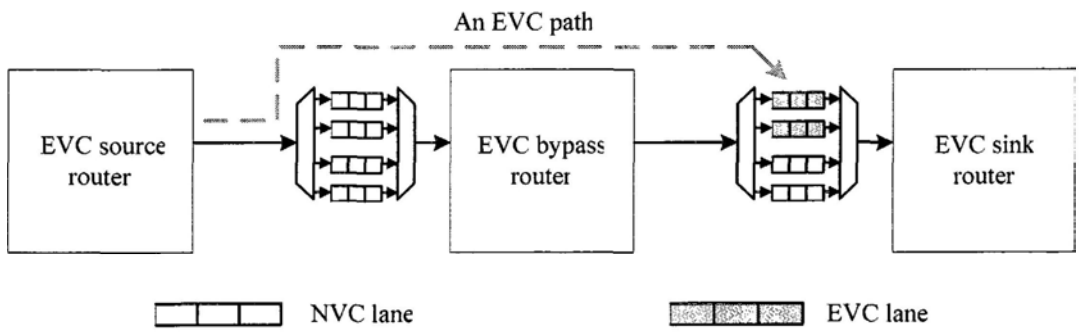


Figure 3.4. Illustration of EVC components

Figure 3.5 shows EVC router pipelines. The non-express pipeline is used for NVC flits. Functions of the pipeline stages are described in Section 1.2.5. The SVA stage, which combines the VA and SA stages together, is presented in Section 2.3. When an EVC flit arrives at an EVC bypass router, it goes through either the express pipeline or the aggressive express pipeline, depending on microarchitecture of the EVC bypass router. In the express pipeline, the EVC flit skips BW, VA and SA, and advances directly to the ST stage. The BW stage can be skipped because the EVC flit does not need to be buffered in the bypass router. The VA stage can be skipped because the EVC flit is not required to be saved in NVC lanes of the next router. The SA stage can be removed due to the EVC flit always have higher priorities over NVC flits and are thus able to pass through the crossbar switch without any contention. In the aggressive express pipeline, the ST stage is further skipped. In this case, the EVC flit will bypass the crossbar switch as well. The aggressive express pipeline removes all pipeline stages in a traditional VC router and is only left with the LT stage, which makes the one hop to pass the VC router be reduced.

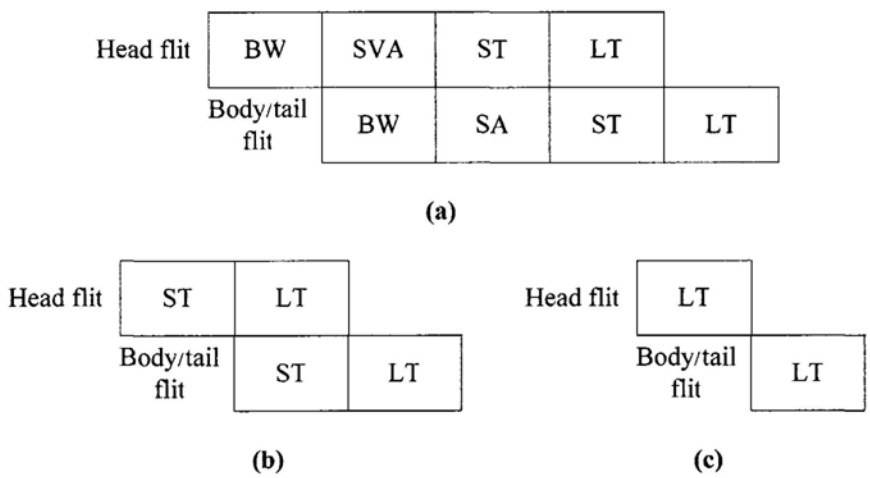


Figure 3.5. EVC router pipelines [36]. (a) Non-express pipeline. (b) Express pipeline. (c) Aggressive express pipeline.

As many packets going through EVC paths skip pipelines of intermediate routers, average hop count is reduced and thus average packet latency is decreased. Given a

particular topology and routing scheme, network throughput is largely determined by the flow control mechanism. The EVC flow control is able to build particular communication flows in the network, thereby improves resource utilization and reduces contention, and thus pushes network throughput. In addition, the total energy that a flit consumes at a router is given as [49]:

$$E_{router} = E_{wrt} + E_{read} + E_{arb} + E_{xb} \quad (1)$$

where E_{wrt} and E_{read} are the energy dissipated by buffer write and read, E_{arb} is the energy consumed by control logic, including routing computation, VC allocation, switch allocation etc., E_{xb} is the energy to traverse the crossbar switch. Ideally, E_{wrt}, E_{read}, and E_{arb} can be entirely saved for an EVC flit when the express pipeline is used. Even E_{xb} can be saved if the aggressive express pipeline is applied.

3.3.2 EVC router microarchitectures

Some router components are added/changed to realize the EVC flow control compared to the generic VC router microarchitecture that is shown in Figure 1.4 (a). Figure 3.6 presents EVC router microarchitectures. Differences of EVC source, sink, and bypass routers from the generic VC router are filled by different patterns.

For an EVC source router, a separate EVC allocator is added to allocate EVC lanes for packets that will travel EVC paths. For an EVC bypass router, the crossbar switch will remain unchanged in the express pipeline. Otherwise, it will be aggressively designed to bypass the ST stage as well in the aggressive express pipeline. For an EVC sink router, some NVC lanes are changed to EVC lanes to buffer packets travelling on EVC paths.

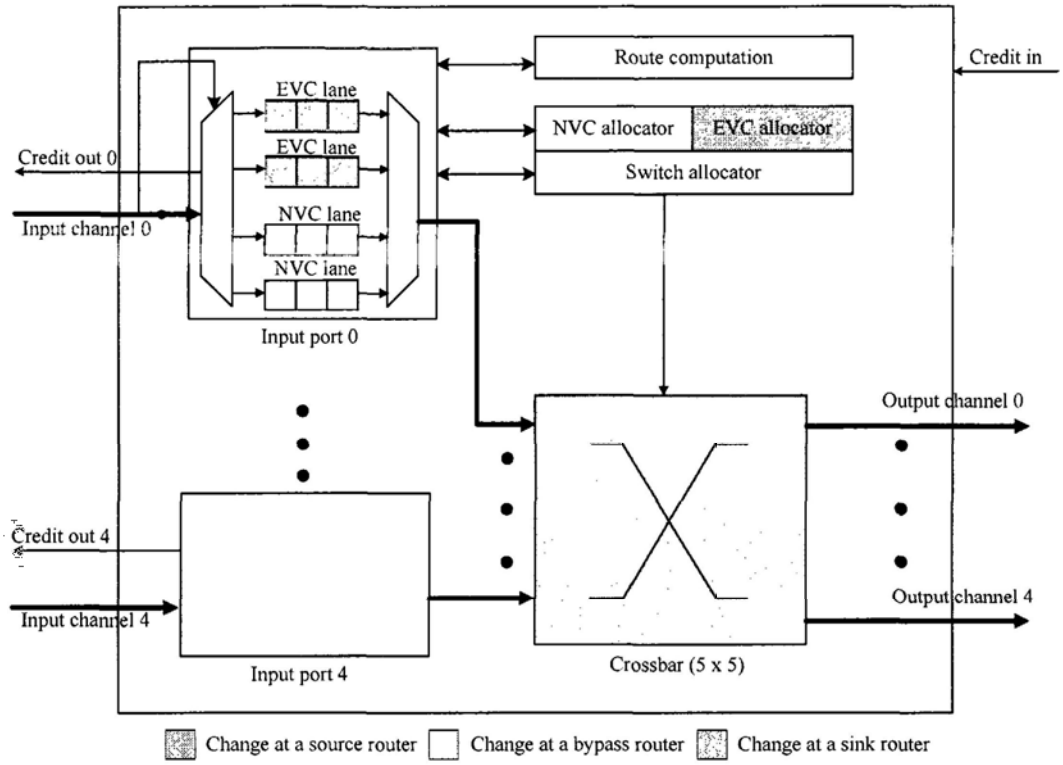


Figure 3.6. EVC router microarchitectures.

3.3.3 Static EVCs network

Figure 3.7 presents a static EVCs network based on a 5×5 mesh. Like the express cube network that is demonstrated in Figure 3.2 (a), express channels are those channels that connect a subset of routers to their v-router away neighbors. The only difference is that express channels in a static EVCs network are *virtual* channels whereas those in an express cube network are *physical* channels. Express virtual channels are inserted regularly along X and Y dimensions and all of them have a uniform interval. *Thus, the uniform interval, v, is the most important parameter for a static EVCs network when network topology and routing strategy are defined.*

Let us illustrate how to utilize EVC paths by an example. Assume that the PE connected to router 01 sends packet *A* to the PE connected to router 34. Packet *A* travels form the source to the destination in the following steps. 1). It flows from local port of router 01 to west port of router 02, going through all pipeline stages of

router 01. It is then buffered in an NVC lane at west port of router 02 because it does not travel along the EVC path that is from router 00 to router 02. 2). Since the EVC path from router 02 to router 04 is on the routing path of packet *A* (assume XY routing scheme is used), packet *A* travels from west port of router 02 to west port of router 04 along this EVC path. It goes through router pipelines of router 02, skips router pipelines of router 03, and is finally buffered in an EVC lane at west port of router 04. 3). Like the step 2, packet *A* goes through the EVC path from router 04 to router 24. It propagates router pipelines of router 04, skips router pipelines of router 14, and is buffered in an EVC lane at south port router 24. 4). Packet *A* flows from south port of router 24 to south port of router 34. 5). Like step 1, packet *A* goes from south port to local port of router 34. In summary, packet *A* goes through five hops (01, 02, 04, 24, and 34) and skips two hops (03 and 14). Compared to the generic mesh network, two hops are reduced.

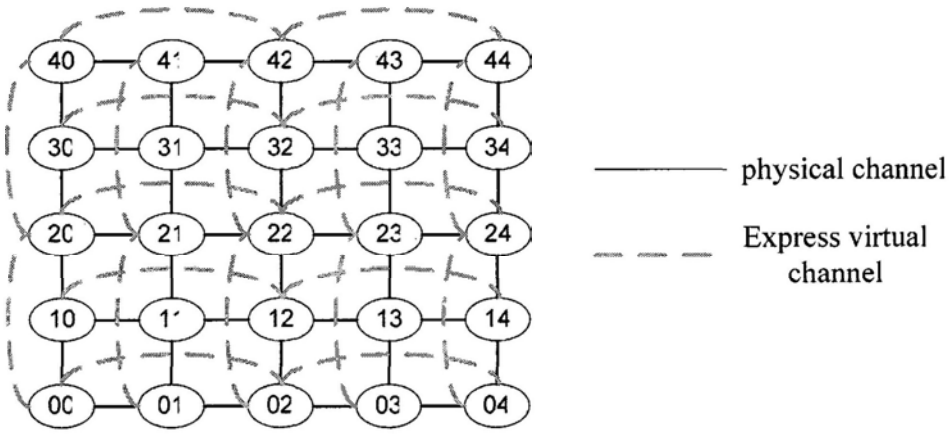


Figure 3.7. Example of a static EVCs network.

Many rules are used to constrain insertion of EVCs in a static EVCs network. They are summarized as follows.

- EVC paths are inserted in a bidirectional manner. It means that an EVC path will be inserted from r_j to r_i if an EVC path is inserted from r_i to r_j. For simplicity, a pair of EVC paths, one in each direction, is represented by one

green and dotted line in Figure 3.7. This is the same as insertion of EPC paths. However, inserting EPC paths in a bidirectional fashion aims to keep the number of input ports equal to the number of output ports for EPC routers. Nevertheless, this is unnecessary when EVC paths are inserted because adding an EVC path does not add any input port. This rule often leads to bad EVC paths to be inserted, for example, the EVC path from router 00 to 02 in Figure 3.1.

- All EVC paths have a uniform interval. The uniform interval, v, is the only one parameter to determine architecture of a static EVCs network for a given topology. Efficiency of EVC paths is heavily dependent on their utilizations. In other words, higher efficiency will be achieved if more traffic loads travel along EVC paths. However, although the uniform interval makes the EVC insertion algorithm very simple, it always leads to low efficiency because routers that have large communication loads between each other do not have the uniform interval in most networks.

- EVC paths are restricted to be along one dimension, either X or Y dimension. Thus, packets have to go through all router pipelines when turning to a different dimension.

- Routers are distinguished as either an EVC source/sink router or an EVC bypass router. This is to say, an EVC source/sink router can never be bypassed. This rule prevents some good EVC paths from being inserted as well. Especially, bypassing an EVC source router always saves more energy because an EVC source router consumes more energy than a generic VC router (normal router) due to added control logic.

3.4 APPLICATION-SPECIFIC EVCS INSERTION METHODOLOGY

3.4.1 Problem formulation

Given an application communication graph *CG*, a topology graph *TG* and a mapping function *M* [50], communication volumes between network routers can be calculated, where is the start point of our work. A 2D mesh topology with $m \times n$ tiles is studied. However, the proposed algorithm can be applied to other topologies with small modifications.

Simply stated, assuming a reasonable mapping (it means that the mapping is optimized for some objective) has been done from an application to a network topology, our objective is to decide which router pair should an EVC is inserted to, such that the maximum power saving is achieved. We firstly make some definitions to formulate the problem.

Definition 1: A *router communication graph, RCG = G(R,C)*, is a directed graph, where R is the set of routers and C is the set of communications. For a communication $c_{i,j} \in C$, $c_{i,j}$ represents the communication volume from a source router r_i to a sink router r_j. In other words, $c_{i,j}$ only includes the traffic generated from r_i and consumed by r_j.

Definition 2: A *router aggregate communication graph, RAG = G (R, A, B)*, is a directed graph, where R is the set of routers, A is the set of aggregate communications between router pairs, and B is the set of aggregate communications travelling routers. For an aggregate communication $a_{i,j} \in A$, $a_{i,j}$ means the aggregate communication load from r_i to r_j. Note that $a_{i,j}$ includes all the traffics flowing from r_i to r_j. For an aggregate communication $b_i \in B$, b_i denotes the aggregate

communication travelling r_i. The calculations for $a_{i,j}$ and b_i are explained in Section 3.4.2.

In addition, many parameters are used in this thesis chapter. They are listed in Table 3.1 for reference.

Table 3.1. PARAMETER LIST FOR AS-EVC INSERTION

Parameter	Description
DM	Manhattan distance travelled by a message. $DM_{i,j} = \mid ix - jx \mid + \mid iy - jy \mid$
DV	Virtual distance travelled by a message. The computation is illustrated in Figure 3.8.
E	Energy consumption of a component.
f	The normalized inter-router communication volume. $f_{i,j} = \frac{c_{i,j}}{\sum_i \sum_{j \neq i} c_{i,j}}$
g	The routing algorithm related coefficient.
α	Aggressive express pipeline: α=0. Non-aggressive express pipeline: α=1.
β	The energy ratio of a crossbar to a router.
λ	The energy ratio of an EVC source router to a normal router.
μ	The average inter-node distance.

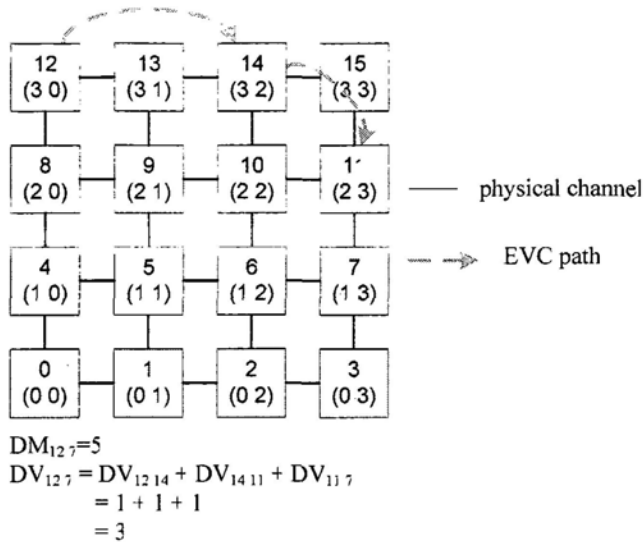


Figure 3.8. Illustration of $DV_{i,j}$ computations. $DV_{12,14}$ is 1 ($DM_{12,14}$ minus 1) because r_{13} is skipped. Similarly, $DV_{14,11}$ is 1 because r_{11} is skipped.

Using these notations, the problem to insert EVCs in an application-specific way can be formulated as follows.

Given

- The router communication graph *RCG*
- The deterministic routing algorithm
- The EVC insertion rules

Determine

- The set of EVCs to be added

Such that

- The power saving is maximized, subject to the EVC insertion rules.

Our algorithm inserts the most beneficial EVC at every iteration and it stops as soon as a pre-set threshold is checked. For low-power NoCs, the pre-set threshold is the minimum energy saving by an EVC path. It is set as zero in our experiments to achieve the maximum total energy saving for a network. Also, it can be defined as a non-zero value to prevent EVC paths with low energy savings from being inserted. Besides, other objectives, such as minimizing average packet latency, can be set to replace the goal of maximizing power saving.

3.4.2 Determination of the most beneficial EVC

The $a_{i,j}$ and b_i in a *RAG* are denoted as

$$a_{i,j} = \sum_{c_{p,q} \in C} c_{p,q} \times g(i,j,p,q) \quad (2)$$

$$b_i = \sum_{c_{p,q} \in C} c_{p,q} \times g(i,p,q) \quad (3)$$

where $g(i,j,p,q)$ is 1 when the routing path from r_i to r_j is covered by the routing path from r_p to r_q. Likewise, g(i,p,q) is 1 if r_i is covered by the routing path from r_p to r_q. Otherwise, they are zeros. The computations for $a_{i,j}$ and b_i are illustrated in Figure 3.9. Both traffics from r_{14} to r_4 and traffics from r_{13} to r_8 have to flow from r13 to r8, so $a_{13,8}$ is the sum of $c_{14,4}$ and $c_{13,8}$, say 200MB. However, only traffics from r_{14} to r_4 flows from r_{12} to r_4 so that $a_{12,4}$ is 100MB. b_{13} is 300MB because all of $c_{14,4}$, $c_{13,8}$, and $c_{13,15}$ must traverse r_{13}.

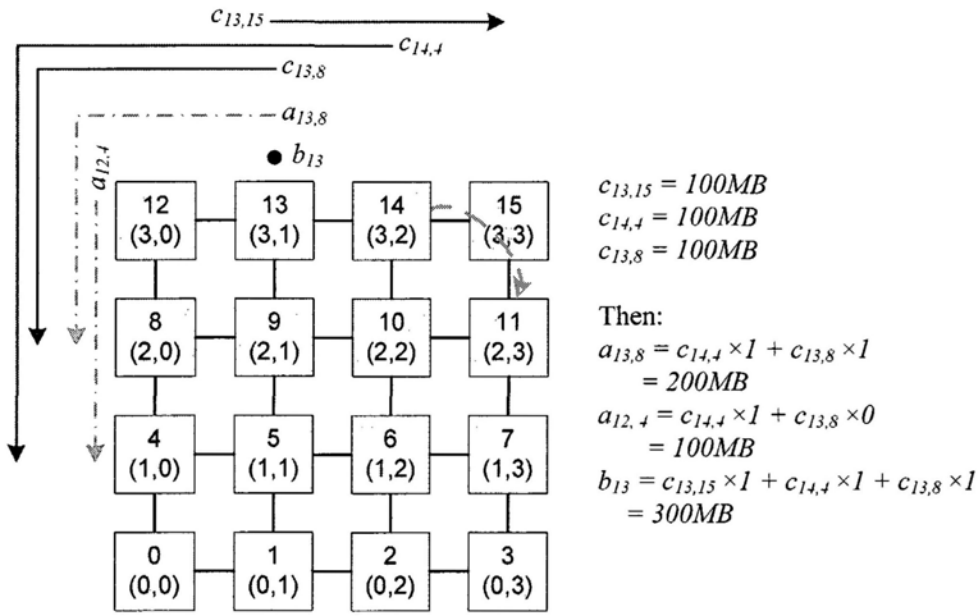


Figure 3.9. Illustration of $a_{i,j}$ and b_i computations.

When a flit travels from a source router to a sink router, total energy consumption includes energy consumed by routers and energy consumed by links. Energy consumed by links is the same whether or not a flit uses EVC paths because the total

distance of links traversed by the flit remains the same. Therefore, it is unnecessary to analyse energy models of links. Only energy reduction and overhead models of routers are required to be built.

Figure 3.6 presents microarchitectures for EVC source, sink, and bypass routers. At an EVC source router, control logics are the main overhead. We expect energy cost of the added control logics is small since router energy is dominated by data path instead of control path [33]. At an EVC sink router, there is no buffer overhead when we assume total buffer lanes in sink input ports remain unchanged. They are divided into EVC lanes and NVC lanes. Meanwhile, there is no control logic cost because packets stored in EVC lanes are processed in the same manner as those stored in NVC lanes. Therefore, there is no energy cost. At a bypass router, there are little additional bypass setup logic and wires when aggressive express pipeline is applied. To simplify the following analysis, we ignore little energy cost at a bypass router and only take into account energy cost at an EVC source router.

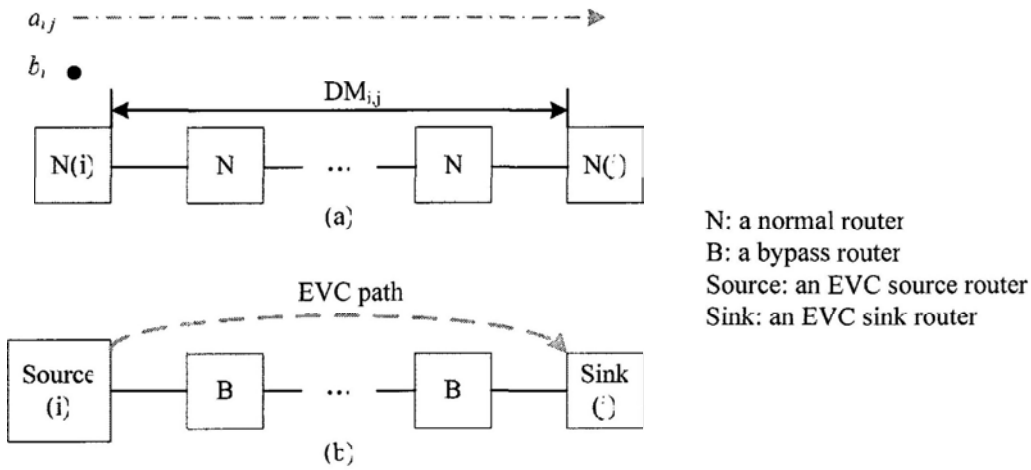


Figure 3.10. EVC reduces energy consumption.

Figure 3.10 illustrates the application of an EVC to a linear array. A regular linear array is shown in Figure 3.10 (a). The Manhattan distance is $DM_{i,j}$ and the aggregate

communication volume is $a_{i,j}$. Router energy consumption to transmit $a_{i,j}$ from r_i to r_j is[10]

$$E_{a1} = a_{i,j} \times DM_{i,j} \times E_{router} \quad (4)$$

A linear array with an EVC path is shown in Figure 3.10 (b). Energy consumption to traversing $a_{i,j}$ from r_i to r_j is

$$E_{a2} = a_{i,j} \times E_{router} + a_{i,j} \times (DM_{i,j} - 1) \times \alpha \times E_{xb} \quad (5)$$

where the first component is the EVC sink router energy dissipation and the second component is the energy to bypass the $DM_{i,j}$-*1* intermediate routers. Therefore, the energy reduction is

$$\Delta E_a = a_{i,j} \times (DM_{i,j} - 1) \times (1 - \alpha\beta) \times E_{router} \quad (6)$$

On the other hand, the energy cost caused by the EVC router r_i is

$$\Delta E_b = b_i \times (\lambda - 1) \times E_{router} \quad (7)$$

Totally, the energy saving of this EVC insertion is calculated as:

$$\Delta E = [a_{i,j} \times (DM_{i,j} - 1) \times (1 - \alpha\beta) - b_i \times (\lambda - 1)] \times E_{router} \quad (8)$$

The equation (8) shows that energy saving is highly related to the aggregate communication volumes $a_{i,j}$ and b_i, which highlights the significance to insert EVCs in an application-specific fashion. Also, it shows that a longer EVC path has larger energy reduction. However, the interval for EVC insertions should be carefully

[10] The energy dissipation to travel the EVC source router r_i is not included in E_a. Instead, it is included in E_b.

selected because a long EVC path occupies many physical channels of intermediate routers.

3.4.3 EVC insertion flow

The flow of a greedy insertion algorithm is described in Figure 3.11. When no pre-set threshold is hit, the algorithm keeps inserting the most beneficial EVC path in the rest EVC paths.

The flow consists of two processes: EVC evaluation and EVC insertion. In the EVC evaluation process, a *RAG* is firstly calculated based on a *RCG* and routing algorithm inputs. Then, EVCs are inserted for all possible pairs of routers. Next, energy saving for each EVC path is computed using the energy models. Meanwhile, an EVC table is generated, with the most beneficial EVC being on the top while the least one being at the bottom.

The EVC paths in the EVC table are then inserted in the EVC insertion process in an iterative way. At each time, the top one in the EVC table is firstly selected. Then, it is checked whether or not this EVC violates any EVC insertion rule. If no violation happens, the information about this EVC is stored in the inserted-EVC set. Otherwise, this EVC is removed from the top of the EVC table and the next EVC is selected. This procedure repeats until a pre-set threshold is hit. Once this takes place, output the inserted-EVC set.

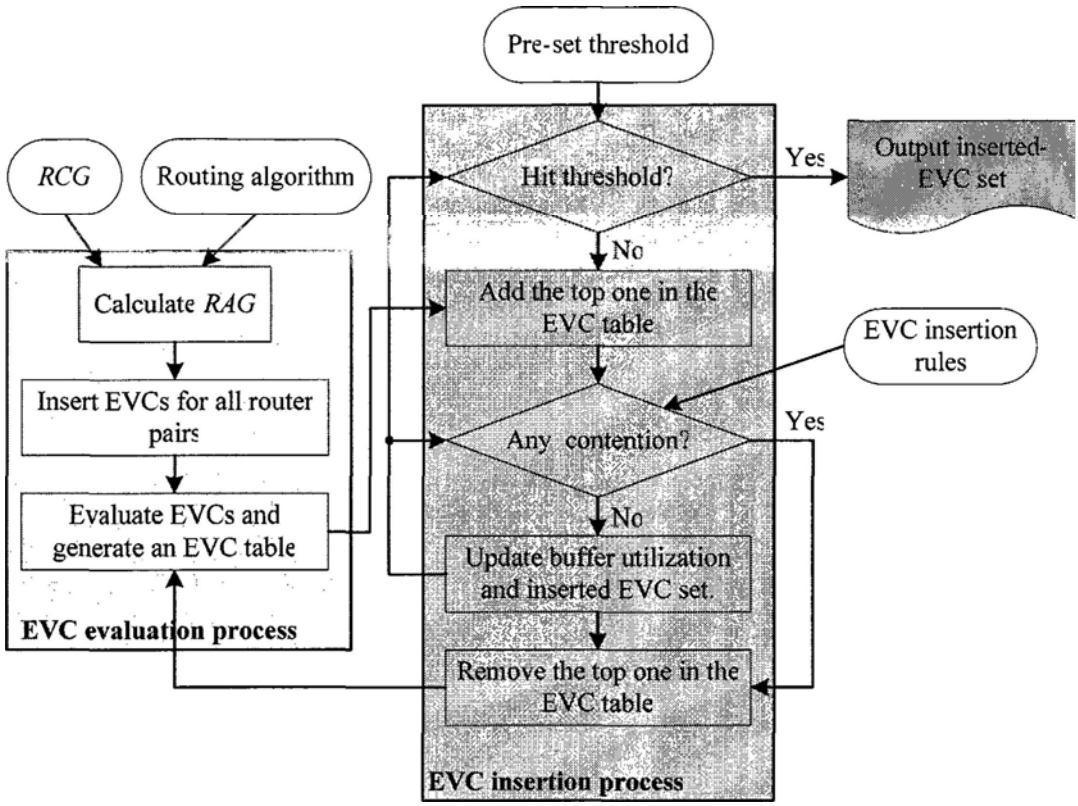


Figure 3.11. Greedy insertion algorithm.

Each EVC insertion has to comply with several rules. Firstly, it can not contend physical channels which have been already occupied by the previously inserted EVCs. That is to say, no EVC overlapping is allowed. Secondly, a router can have maximum four EVCs, including both EVCs sourcing from it and EVCs sinking at it. These rules reduce the EVC insertion flexibility, and thus result in some good EVCs can not be added. However, they make EVC control logics simple to be implemented. Thirdly, an EVC path can not exceed the maximum insertion interval because a long EVC path occupies many physical channels. Although it reduces large energy consumption, it prevents a lot of following EVCs from being inserted. Totally, it always leads to bad results.

3.5 EVALUATIONS

3.5.1 Experimental infrastructure

The AS-EVC insertion methodology was evaluated for both synthetic and real traffic loads. We compared normal mesh networks (baseline), mesh networks with static EVC insertions (static EVCs) [36], and mesh networks with AS-EVC insertions for each traffic pattern. We estimated power savings using proposed high-level models for each topology and traffic pattern as the interval of static EVCs changes from two to five and found that the maximum power savings are all obtained at the interval of two. Thus, the interval of static EVCs is fixed as two for all traffics. Similarly, the maximum AS-EVC intervals for various topologies and traffic patterns are determined (Table 3.2). λ is set as 1.05 and β is defined as 0.25 empirically.

Table 3.2. THE MAXIMUM EVC INTERVALS FOR AS-EVC.

Topology	Traffic	Max interval
4×4	uniform	2
	transpose	4
6×6	uniform	2
	transpose	4
10×4	apsi, gzip, swim, parser	4

A baseline router has five ports, four FIFOs per input port, and four-flit deep buffers for each FIFO. A flit is 69-bit wide, consisting of 64-bit payload and 5-bit flit control overhead. Some router microarchitecture optimizations such as look-ahead routing, combined VC and switch allocation were incorporated in the baseline router. If an input port is the sink port of an EVC path, four FIFOs at this port are divided into two EVC lanes and two NVC lanes. Aggressive express pipeline was used unless otherwise stated.

NoCs supporting EVC flow control was modelled using SystemVerilog. After EVC insertions, corresponding input ports at EVC sink routers have two NVC lanes and two EVC lanes. Thereby, the numbers of NVCs at input ports are no longer uniform. Some of them have two NVC lanes whereas others have four NVC lanes. Our models handle this problem by setting the number of NVCs at each input port as a parameter. All arbitration logics for NVCs are also controlled by the parameter to reduce control logic redundancy.

Instead of using high-level models for fast power evaluations, power evaluations were performed in post-synthesis stage for two reasons. Firstly, the accuracy is acceptable because we don not need to calculate power dissipations of inter-router physical links that consume the same power for the three compared architectures. Secondly, effort to do post-layout evaluations for a wide range of traffic patterns is unacceptable. UMC 130nm library with 1.2V power supply voltage was applied. All simulations run at 250MHz. *For each traffic pattern, to ensure the compared three NoCs have nearly the same throughput when their power profiles were obtained, the injection rate was set before any of the three NoCs enters saturation.*

3.5.2 Synthetic traffic loads

We considered uniform and transpose as synthetic traffics. Uniform traffic assumes randomly distributed destinations. Transpose traffic assumes the destination node for packets generated by a node is always the symmetric node with respect to the diagonal. Therefore, it achieves the maximum degree of temporal locality.

The average inter-node distance is an important dynamic property of networks because it represents the average number of routers travelled by packets. It is computed as[11] [46]:

$$\mu = \sum_{i} \sum_{j \neq i} f_{i,j}(DV_{i,j} + 1) \qquad (9)$$

Clearly, μ determines average packet delay without contention, and power dissipation of routers. A larger reduction of μ indicates that larger power reduction may be obtained. Meanwhile, it is easy to be computed. Hence, it is a useful metric to estimate the power effect of EVCs insertion in the early stage. However, a larger μ decrease does not definitely mean a higher power saving because it assumes an ideal condition where power consumption of a router can be entirely removed if it is bypassed and no power overhead is generated by EVCs insertion.

Let us firstly demonstrate the impact of EVCs for a 4×4 mesh network (Figure 3.12). Compared to the baseline, static EVCs reduces total router power by 6.81% for uniform traffic. This reduction increases to 7.41% when using AS-EVC. For transpose traffic, the power reduction is 8.44% and 23.49% for static EVCs and AS-EVC respectively.

Compare AS-EVC with static EVCs. When the traffic changes from uniform to transpose, power decrease by static EVCs shifts a little from 6.81% to 7.41%. However, power reduction by AS-EVC increases significantly from 8.44% to 23.49%. This claims that AS-EVC effectively exploit the characteristics of the highly-specific transpose traffic whereas static EVCs loses a huge optimization room

[11] Use $DV_{i,j} + 1$ instead of $DV_{i,j}$ because we assume that a packet takes one hop to eject out at the sink router.

because it considers transpose traffic in the same way as uniform traffic. Normalized average inter-node distance (Figure 3.13) supports this conclusion as well. Reduction of μ by static EVCs only grows from 15.68% to 16.73% while it rises significantly from 18.96% to 39.80% when applying AS-EVC. Figure 3.12 (b) compares total router power savings of the two methods for the same traffics. AS-EVC outperforms static EVCs by 23.98% for uniform traffic, and 216.99% for transpose traffic.

Scalability analysis. To evaluate the scalability of AS-EVC, we investigate a 6×6 mesh network under the same two traffics (Figure 3.14). As can be seen, AS-EVC continues to show a considerable power gain as compared to the baseline, with the power reduction of 11.01% under uniform traffic and 20.48% under transpose traffic. However, the gain over static EVCs decreases when network size increases. AS-EVC only reduces power 2.45% more than static EVCs for uniform traffic. It implies that static EVCs scheme is enough for large networks with randomly distributed loads. However, the improvement is still pronounced for transpose traffic where static EVCs saves 17.6mW while AS-EVC reduces 36.5mW, with 107.31% more power saving.

It is interesting to observe that power saved for a 6×6 network (20.48%), under transpose traffic, is smaller than a 4×4 network (23.49%) although the former (48.31%) obtains a bigger μ reduction than the latter (39.80%). It indicates that although power reduced by bypassing intermediate routers for a 6×6 network is bigger than a 4×4 network, power overhead caused by EVC source routers for a 6×6 network is bigger than a 4×4 network, and the second effect overwhelms the fist effect.

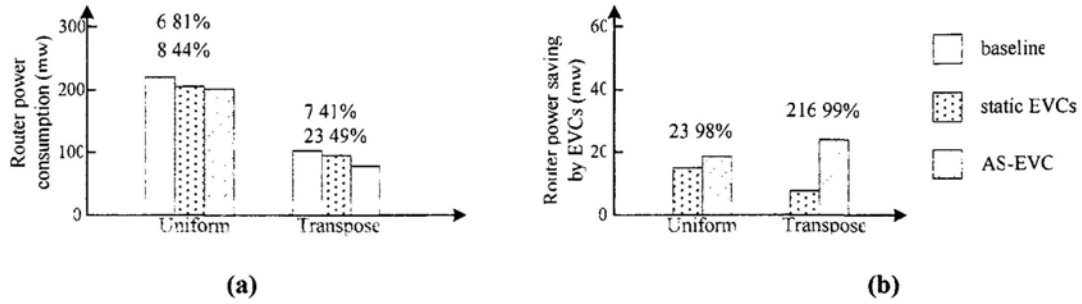


Figure 3.12. The entire NoCs power for a 4x4 mesh network.

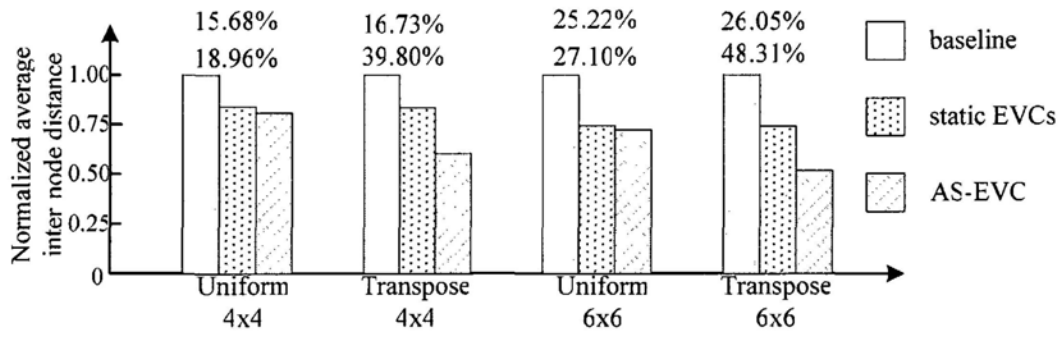


Figure 3.13. Normalized μ for synthetic traffics.

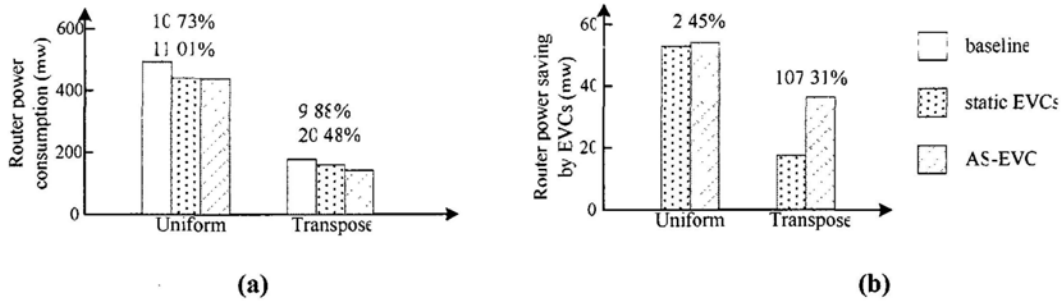


Figure 3.14. The entire NoCs power for a 6x6 mesh network.

Impact of express pipelines. Figure 3.15 shows power for a 4×4 mesh network when the express pipeline is applied. As expected, compared to the aggressive express pipeline (Figure 3.12 (a)), less power consumptions are saved for both uniform traffic and transpose traffic. For instance, the express pipeline reduces power by 5.85% whereas the aggressive express pipeline saves power by 8.44% for uniform traffic when the AS-EVC scheme is used. The main reason is that packets flowing through an EVC path have to traverse crossbar switches at intermediate routers at the express pipeline.

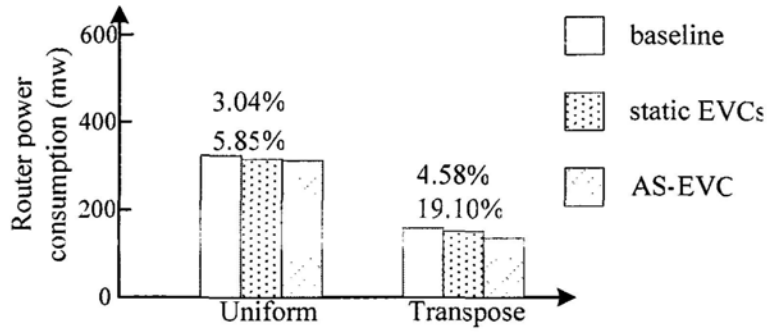


Figure 3.15. Power at express pipeline.

3.5.3 Real traffic loads

The benchmarks in the Minne-SPEC suit [51] were used to evaluate the impact on realistic traffics. Firstly, a benchmark in the Minne-SPEC suit was fed into the TRIPS on-chip network (*OCN*) simulator to capture an *OCN* traffic trace. This *OCN* trace was then applied to a traffic decoder to generate a *RCG*. *OCN* is a wormhole routed, 4×10 mesh network with YX routing. It serves as an infrastructure to interconnect the two TRIPS processor cores, the individual banks that form the second level cache and the I/O units [52, 53]. We equivalently mapped *OCN* to a 10×4 mesh with XY routing since our AS-EVC algorithm and router models are based on XY routing.

Figure 3.16 present simulation results of Minne-SPEC benchmarks. These graphs follow the same trend as the experiments for synthetic traffics, with AS-EVC clearly outperforming both baseline and static EVCs structures. Power reduction compared to the baseline architecture is above 12% for all tested benchmarks, with the most (15.82%, 31.3mw) for *gzip* and the least (12.99%, 30.1mw) for *apsi*. *This is a significant improvement because the power reduction is not over power of a single router component, but over total power of all routers in a NoCs*. The gain over the static EVCs is bigger than 35% for all traffics, with an average value of 57.14%. The largest improvement is seen for *gzip*. While the static EVCs reduces 17.5mw, the

AS-EVC decreases 31.3mw (78.86% more). The AS-EVC obtains this improvement by inserting only 20 EVC paths for *gzip*, 32 less than the static EVCs.

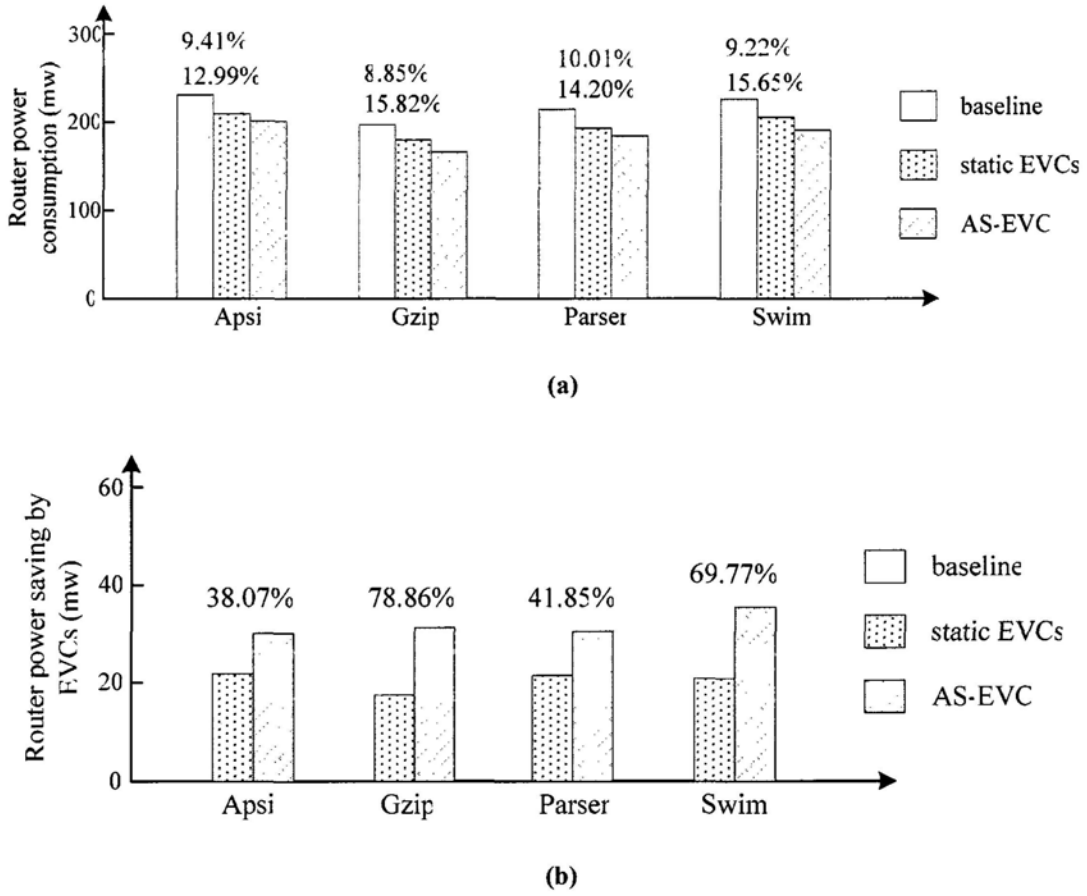


Figure 3.16. The entire NoCs power for TRIPS OCN traffics.

3.5.4 Detailed area and power profiles

To further demonstrate how EVCs technique reduces power, we analyze total standby power and total stream power. Since consistent results have been obtained for the TRIPS OCN traffic loads, we report only the results for *swim* benchmark.

Total power of NoCs consists of standby power and stream power. NoCs dissipates a lot of standby power even when it is completely idle. It is a fixed cost for a specific architecture. Stream power represents additional power when packets stream from their sources to their sinks. More stream power is consumed if more packets are processed.

The principle of the EVC flow control is to skip some operations in intermediate routers as packets flow along EVC paths. In other words, the EVC technique can only save stream power. No power can be reduced by the EVCs technique if no packets are routing. On the other hand, it increases standby power because it requires some extra control logics. As seen in Figure 3.17, standby power overhead is 1.50% for the AS-EVC. It increases to 3.22% for the static EVCs in which more EVC paths are inserted. However, stream power is reduced by 20.09% and 12.43% for the AS-EVC and the static EVCs respectively.

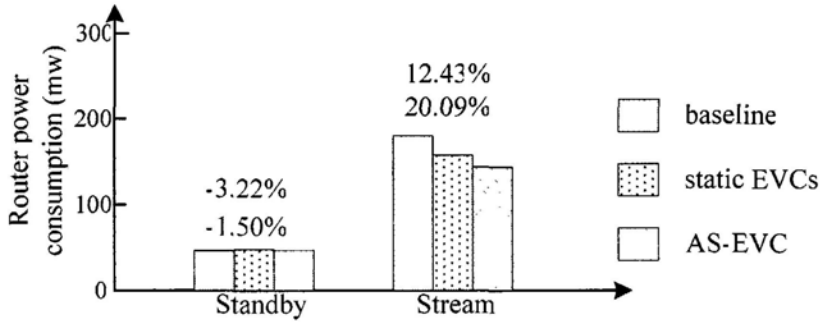


Figure 3.17. Power profile for the TRIPS OCN swim traffic.

The number of logic gates of the entire AS-EVC NoCs for *swim* traffic goes up from 1683.13K to 1720.74K, which only increases 2.23%. A single normal router has 47604 logic gates. The area of EVC source routers with different number of source paths and EVC bypass routers with various number of bypass paths is summarized in Table 3 (EVC sink routers have the same area as a normal router, so the area of them is not reported.). It shows that adding four source paths only increases area by 7.89% and area overhead of adding bypass paths is even smaller. The area overhead is significantly smaller than that caused by the EPC technique (changing a five-port router to a six-port router increases the area by 26.7% [46]). Thus, the EVC technique is highly scalable in terms of area.

Table 3.3. AREA OF SOURCE AND BYPASS ROUTERS.

Source router	Gate count	Bypass router	Gate count
1-EVC	49295 (3.55%)	1-bypass	47781 (0.37%)
2-EVC	50179 (5.41%)	2-bypass	47960 (0.75%)
3-EVC	51092 (7.33%)	3-bypass	48140 (1.13%)
4-EVC	51359 (7.89%)	4-bypass	48347 (1.56%)

3.6 CONCLUSIONS AND DISCUSSIONS

We have proposed a novel, application-specific methodology to insert EVC paths for low-power NoCs in this thesis chapter. A *RAG* is firstly defined to help designers clearly know communication characteristics of an application. Then, simple power reduction and power overhead models are built to calculate power savings for all possible EVC paths. Finally, a greedy algorithm is applied to add EVC paths in an iterative way, subjecting to some insertion rules. In a word, for an application, the AS-EVC method is able to quickly insert appropriate EVCs early in the design stage.

We compared power consumptions of the baseline NoCs, the static EVCs NoCs, and the AS-EVC NoCs through VLSI implementations. Experiments on both synthetic and realistic workloads show that the AS-EVC NoCs achieve great improvements over total power of all routers compared to both the baseline and the static EVCs NoCs.

However, there are several directions to improve or extend the AS-EVC methodology. They are described as follows.

3.6.1 Build accurate power models

The main purpose of this thesis chapter is to study the effect of exploiting communication characteristics of applications during EVCs insertion on power consumptions. Although high-level power models are built to estimate power savings, they are inaccurate and are just used to compare different EVC paths. The models

emphasize the impact of communication volumes but ignore accuracy of router power consumptions. Therefore, power estimations using the high-level models are not accurate. Table 3.4 shows router power savings of the AS-EVC NoCs over the baseline NoCs for TRIPS OCN traffics. Power savings estimated using the high-level models are much exaggerated. For example, the high-level models estimate a power saving of 28.29% for the *apsi* traffic. But the real power saving obtained from ASIC tools is only 12.99%.

Table 3.4. ROUTER POWER SAVINGS FOR TRIPS OCN TRAFFICS BY AS-EVC NOCS.

Traffic patterns	Estimations through high-level models	Evaluations through ASIC tools
Apsi	28.29%	12.99%
Gzip	39.07%	15.82%
Parser	32.01%	14.20%
Swim	33.85%	15.65%

The high-level power models are inaccurate because we make several assumptions to simplify analyses. First, router energy is entirely saved when an EVC flit skips an EVC bypass router. This assumption is too optimistic because only part of router energy can be reduced in reality although an EVC flit skips all pipeline stages of an EVC bypass router. Second, when a NVC flit traverses an EVC source router, it consumes 1.05x energy than traversing a normal router no matter how many EVC paths originate from this router. Actually, an EVC source router with more EVC paths consumes more energy. Third, when a NVC flit goes through an EVC bypass router, it consumes the same energy as what it consumes to pass a normal router. In fact, the NVC flit consumes more energy because some logic is added in an EVC bypass router.

Thus, an important improvement is to build accurate high-level energy saving and overhead models. It will have two advantages. First, it can help designers evaluate

power savings of EVCs insertion in early design stage. Second, it helps to compare EVC paths accurately and thus insert better EVC paths.

3.6.2 Allow EVC overlapping

Currently EVC insertion does not allow any kind of EVC overlapping. This is to say, two EVC paths can not share the same physical port. Let us illustrate this rule by an example (Figure 3.18). Assuming EVC 1 (from r_1 to r_4) has already been inserted, neither EVC 2 nor EVC 3 can be inserted because they overlap with EVC 1. However, in fact, conflictions only happen when two EVC flits simultaneously ask for the same output port at an EVC bypass router. Therefore, rules for EVC overlapping should be:

- An EVC source port and an EVC sink port can definitely be overlapped since an EVC flit is processed in the same way as a NVC flit at EVC source/sink routers.
- An EVC bypass input port can certainly be overlapped because at most one EVC flit arrives at the input port in each clock cycle.
- An EVC bypass output port at an EVC bypass router can be overlapped if the EVC paths share the same EVC bypass input port at the same EVC bypass router.

Under the new EVC overlapping rules, both EVC 2 and EVC 3 can be inserted after EVC 1 is inserted. It is obvious that the new rules increase flexibility of EVCs insertion and do not lead of conflictions. Although control logic will become more complicated, we believe that gains are significantly larger than overheads.

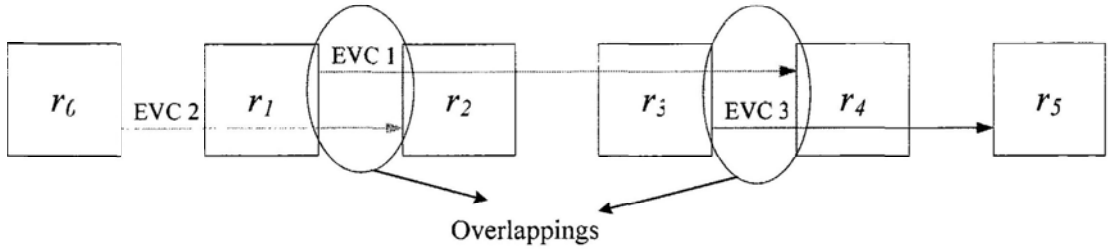


Figure 3.18. Illustration of EVC overlapping.

3.6.3 Compare the EVC with the EPC

Another future direction is to compare application-specific EVCs with application-specific EPCs. On the one hand, inserting EVCs obtains latency, throughput, and energy gains with low costs. However, as a flow-control technique, the EVC benefits global packets at the cost of increasing contention delay of packets those are locally buffered in EVC bypass routers due to the shared buffers, crossbar and physical links. On the other hand, inserting EPCs reduces latency for global packets without blocking packets that are traversing EPC bypass routers. Nevertheless, power and area costs of fatter routers are too high. We suppose that the EPC is better for performance-driven designs whereas the EVC is better for power-driven designs although both the EPC and the EVC techniques can reduce latency and power. Thus, it is an interesting direction to make a comparison of them or explore a mid-way between them to exploit the best of both techniques.

CHAPTER 4. COST-EFFICIENT EVC NOCS IMPLEMENTATIONS

4.1 INTRODUCTION

Express virtual channel is a new flow control mechanism to reduce H through virtually bypass intermediate routers [36]. The authors presented express pipelines, EVC router microarchitectures and evaluated the EVC flow control by high-level models. However, they did not address several important issues that are towards implement the EVC flow control in realistic NoCs designs.

- Optimization of EVCs insertion.
- Hardware designs for EVC routers.
- Buffer architecture for EVC networks.
- Low-power techniques to realize power savings of EVC bypass routers.
- Accurate and detailed prototypes and evaluations.

In Chapter 3, we concentrate on the optimization of EVCs insertion through an application-specific method. Evaluations were performed for a wide range of network sizes and traffic patterns because the objective is to validate the effectiveness of the AS-EVC method. Thus, they were done in the post-synthesis stage that is not the most accurate. Meanwhile, buffer architecture remains uniform for simplicity. In this chapter, we will address the rest of the important issues and contribute in the following aspects.

- We present detailed hardware designs for both EVC source routers and EVC sink routers. At an EVC source router, a head flit process block is designed to

identify whether a flit goes though an EVC path and to load parameters for new packets. A combined switch-VC allocator and its associated logic are designed to allocate NVC and EVC lanes simultaneously, to control network starvation, and to generate flags of EVC flits. At an EVC bypass router, bypass setup logic and bypass datapath are designed for both express and aggressive express pipelines.

- We propose a statistical approach to customize buffer architecture for networks with EVCs. In this approach, the number of EVC/NVC lanes at each input port is fully customized according to utilization statistics of these lanes. Likewise, the buffer depth of EVC lanes at each input port is customized.

- We explore several conventional low power techniques to show how power can be saved when an EVC flit is bypassing an EVC bypass router. Clock gating is comprehensively studied to reduce both clock power and data-input power of buffers. Operand isolation is explored to save power for control components such as the RC and the SVA.

- We evaluate the baseline NoCs and the AS-EVC NoCs for the TRIPS OCN under swim benchmark and a 4×4 mesh under transpose traffic. The customized EVCs insertion is obtained through the AS-EVC method presented in chapter 3. The customized buffer architecture is gotten by the statistical approach. Accurate, detailed evaluations are performed on latency, throughput, area and power based on RTL-level simulations and physical implementations.

This thesis chapter is organized as follows. In Section 4.2, we describe related work. In Section 4.3 and 4.4, the components for EVC source routers and bypass

routers are presented respectively. Section 4.5 proposes a statistical approach to optimize the buffer architecture. Section 4.6 discusses several low-power techniques for EVC routers. After that, the physical implementations for the entire NoCs are presented in Section 4.7 and the results are reported in Section 4.8. Finally, we conclude the chapter in Section 4.9.

4.2 RELATED WORK

4.2.1 Topological techniques

Design and implementation of an entire NoCs using technology below 100nm were explored in [35]. Lee et al. designed a low-power NoCs for high performance Systems-on-Chip [13, 54]. The physical implementation in 180nm CMOS technology shows that the NoCs consumes 51mW of power. In these works, the topologies are fully customized to maximum network performances and to reduce power costs. Accordingly, the number and size of routers are customized.

Although fully customized topologies can achieve high performance, they lead to non-structured wiring which can be problematic. Problems like crosstalk and timing closure may offset the advantages expected from customization. Also, many realistic networks are not completely irregular. Hence, Ogras et al [17, 18] implemented a 4×4 mesh NoCs with application specific long links insertion. However, inserting a new physical link adds a new port to both the end routers of the link, resulting in high power and area costs.

We suggest a better approach so that customization can be fully exploited while keeping the benefits of structured wiring and avoiding high-radix routers. On the one hand, we improve network performances and reduce power through customized EVCs insertion and customized buffer architecture. On the other hand, because long

virtual paths instead of long physical links are added, wiring remains structured and the radix of routers is not changed.

4.2.2 Clock gating

Clock gating (CG) is one of the most successful and widely used techniques for power reduction [55-58]. It dynamically shuts off the clock to blocks of a design that are idle or not producing meaningful results. Power reduction of CG depends on power consumed by the blocks and time to turn off them.

The basic idea of CG is to AND/OR the clock with an enable signal, so that a flip-flop only receives the clock when the enable signal is logic high. Although adding an AND/OR gate along the clock path is the simplest method, glitches on the enable signal are propagated to the clock pin of the flip-flop and thus generate errors. Thus, designers usually use integrated CG cells provided in standard libraries. As shown in Figure 4.1, there are two kinds of CG cells: logic low disabled and logic high disabled. The logic low disabled cell turns off the clock by keeping the gated clock pin (gclk) logic low, which is generally used to disable positive edge-triggered flip-flops. The logic high disabled cell is the opposite and is usually used to disable negative edge-triggered flip-flops.

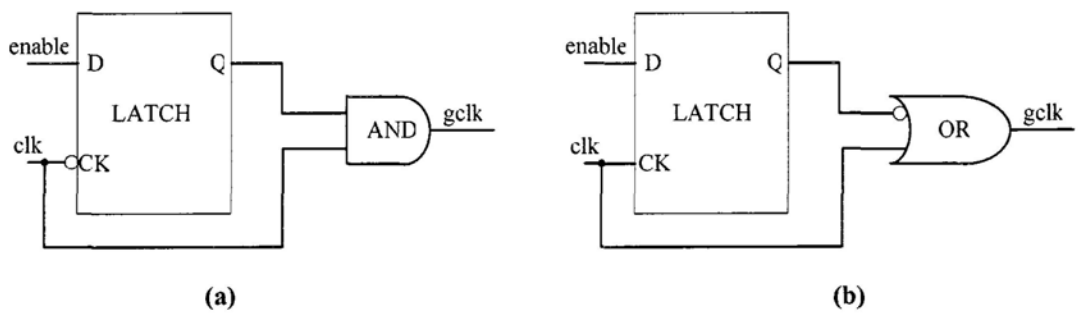
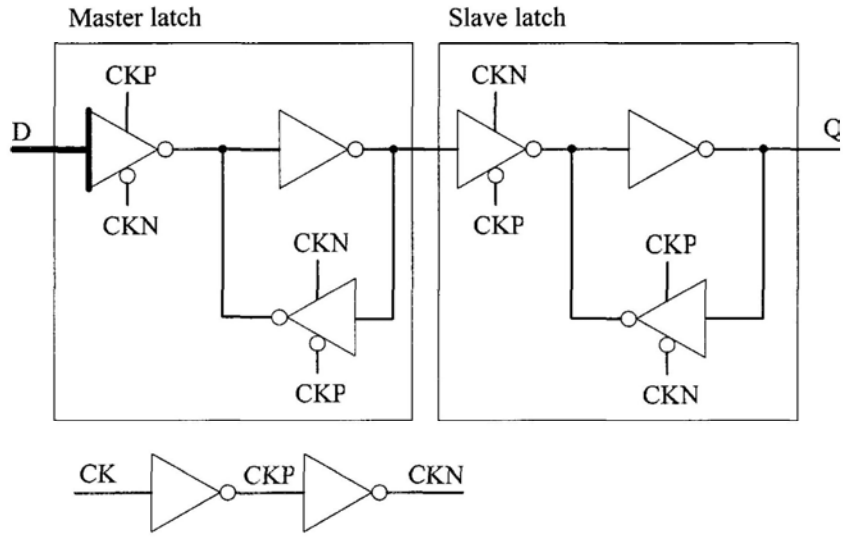


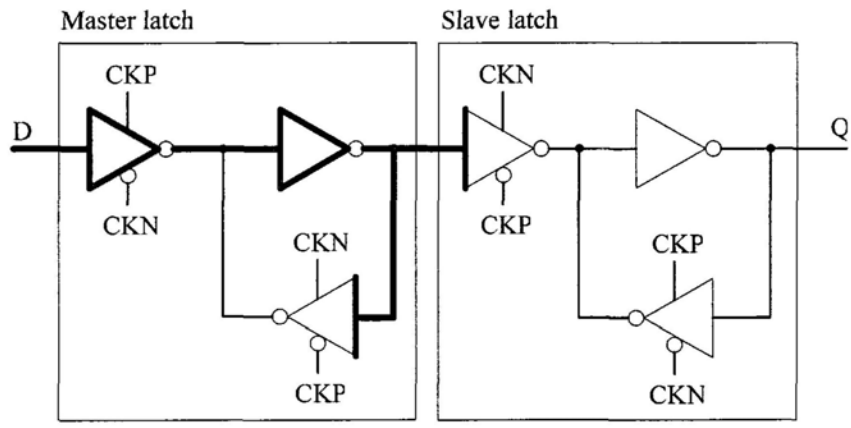
Figure 4.1. CG cells. (a). Logic low disabled. (b). Logic high disabled.

Power consumption of a positive edge-triggered flip-flop is described as follows assuming transitions on the data input signal D and a stable clock signal CK. In

Figure 4.2 (a), the clock signal is logic high (CK='1'). Since the master latch is not transparent in this case, only the input gate capacitances (marked bold) are reloaded. Figure 4.2 (b) shows the situation when the clock signal is logic low (CK='0'). Transitions on the data signal affect the internal nodes (marked bold) of the master latch and the gate capacitances of the slave latch. For this reason, the power consumption of the flip-flop is much higher if the clock signal is stabilized as logic low.



(a)



(b)

Figure 4.2. Schematic of a positive edge-triggered D flip-flop [59]. (a). CK is equal to 1. (b). CK is equal to 0.

4.3 EVC SOURCE ROUTER

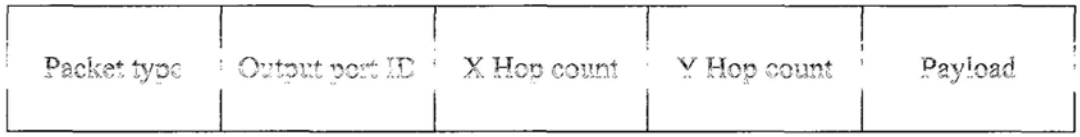
At EVC source routers, the control components need to handle both NVC flits and EVC flits. Thus, special functions for EVC flits are added in some control components. In this section, we present their designs.

4.3.1 Head flit process block

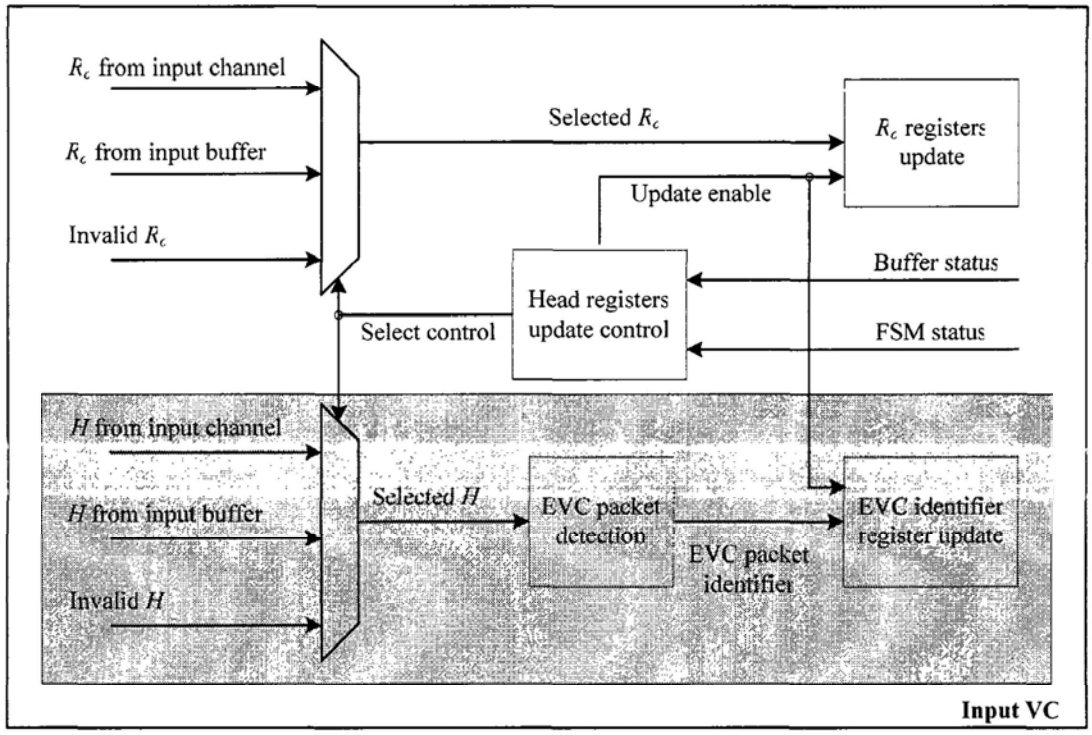
The format of a head flit is shown in Figure 4.3 (a). The packet type is used for special purpose such as QoS. The output port ID (R_o) directs the head flit to the appropriate output port of the current router. The hop counts in X direction and Y direction are used to determine the appropriate output port ID in preparation for use in the next router. Each input VC has a head flit process block. For normal routers, the head flit process block (Figure 4.3 (b)) only extracts *Ro* and saves it to registers. For EVC source routers, it needs to additionally detect whether a head flit can take advantage of the EVC paths available by comparing the hop counts in the head flit to the hop counts representing the EVC paths originating from this router. If an EVC path is covered by the required route of the head flit, the head flit will go through the EVC path. Otherwise, it will traverse a normal path.

The head flit process block determines when to update the R_o and EVC identifier registers and where to get the head information for update. It updates the registers for a new packet at the cycle right before the tail flit of a packet is leaving. The R_o and EVC identifier registers are updated according to the input buffer and the Finite State Machine (FSM) status of the input VC. When the head flit of a new packet is already queued up in the buffer, R_o and H from this buffer are used. When a head flit destined for this input VC is coming in, R_o and H from this coming head flit are used. In cases

other than the two previously described situations, any invalid R_o and H (out of the known range) are used.



(a)



(b)

Figure 4.3. (a). Head flit format. (b). Head flit process block.

4.3.2 Switch-VC allocator

The block diagram of a switch-VC allocator (SVA) and its associated logics are depicted in Figure 4.4. The SVA combines VC allocation and switch allocation together.

Each output port has an output VC control sub-block. The NVC status table keeps detailed information of all output NVCs in the output port q, such as busy, empty, full etc. The NVC selector searches for a NVC which is free to be allocated and saves

its ID to the free-NVC FIFO. The output of the free-NVC FIFO indicates which NVC can be first made available to those input VCs that are connected to the output port q. If the output port q is an EVC source port, a set of EVC modules are required to control the EVC lanes at the corresponding EVC sink port. Functions of the EVC modules are the same as those in the NVC counterparts.

Each input VC has a resource availability check sub-block. It verifies whether the input VC meets the requirements to generate a request based on the information from the output VC control blocks. The NVC/EVC allocation check module works only on head flits. A signal is set to be true if the free-NVC/free-EVC FIFO is not empty. Otherwise, it is set to be false. Checking allocation condition has to be done before generating requests for the input port arbiter because a check performed afterward may cause deadlock dependency between the input VCs. The NVC/EVC credit check module operates on all flits. It verifies whether there are buffer spaces available in the assigned output NVC/EVC.

Each input port has a request generator for input port arbiter, an input port arbiter, and a request generator for output port arbiter. The request generator for input port arbiter creates valid requests for the input VCs which meet all the resource requirements. The input port arbiter grants one of the requests. The request generator for output port arbiter then determines which output port arbiter the winner will go to. The winners from each input port proceed to the output port arbiter which selects one among the winners. The NVC/EVC status table of the corresponding output VC control block will be updated accordingly. In summary, NVCs and EVCs are processed separately in both output VC control blocks and resource availability check blocks. However, the arbitration blocks do not separate EVCs from NVCs. In other words, they treat all input VCs the same way no matter which kind of output

VCs an input VC is eventually allocated. In addition, the NVC/EVC allocation and the switch allocation share the same arbitration blocks. *Thus, our proposed architecture will have much lower design costs than that using separate arbitration logics for NVC allocation, EVC allocation, and switch allocation as presented in [36].*

Each EVC path is provided a starvation avoidance block. At high injection rates, it is possible that an EVC source router always sends EVC flits along an EVC path, leading to NVC flits locally buffered at each EVC bypass router on this path may never get a chance to use the crossbar and the physical channel. The starvation avoidance block counts the number of EVC flits going to an EVC path. Once the number hits a threshold, it stops sending EVC flits by blocking the corresponding input VC requests. After stopping for some cycles, it resumes sending EVC flits and the count. *The avoidance logic is located at an EVC source router instead of EVC bypass routers. As a result, no reverse starvation signals are sent from the bypass routers to the EVC source router. This reduces wire costs and is really different from the starvation control policy presented in [36].*

Each EVC path is provided an EVC flit flag generation block as well. It generates a signal each time an EVC flit wins the arbitration. It is sent out one cycle ahead the EVC flit to set up each EVC bypass router in advance.

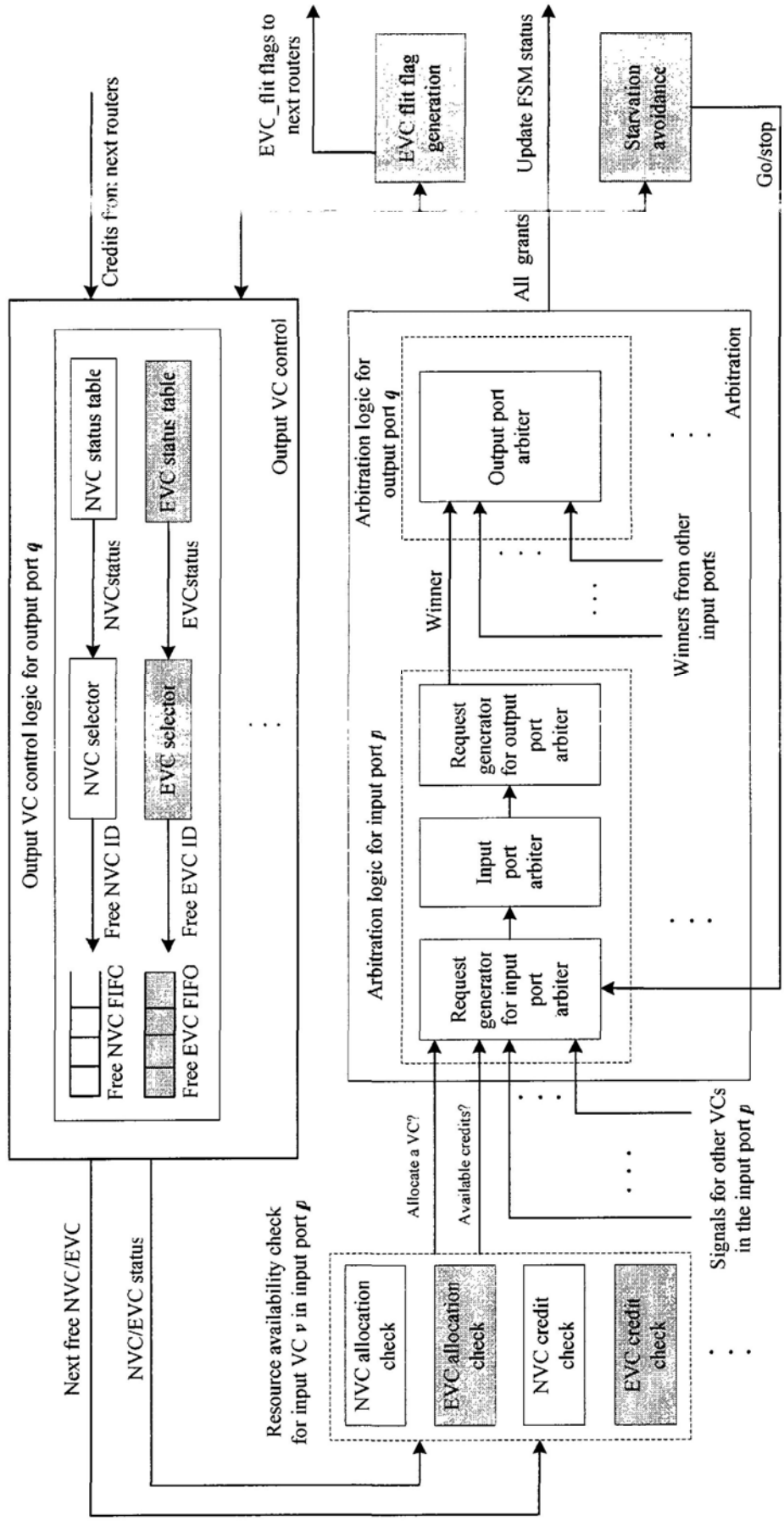


Figure 4.4. Switch-VC allocator and the associated logics.

4.4 EVC BYPASS ROUTER

In this section, we describe the bypass setup and the necessary changes to the data path for EVC bypass routers under both express and aggressive express pipelines.

4.4.1 Express bypass router

Figure 4.5 presents the EVC bypass router microarchitecture under express pipeline. We assume that an EVC path goes through the router input port i_0 to the router output port o_2. Since a single crossbar input/output port is provided for each router input/output port, the input/output port index of a crossbar is the same as that of a router. The 2-1 multiplexes are added at the crossbar input port i_0 to select between EVC flits and NVC flits. EVC flits have priority over NVC flits so that an EVC flit traverses the crossbar as soon as it arrives. To achieve this, the bypass setup block must configure the crossbar to logically connect o_2 to i_0 one cycle before the EVC flit arrives. In this case, both i_0 and o_2 of the crossbar are reserved for the EVC flit. As a result, no NVC flits can be sent from i_0 and no NVC flits can be sent to o_2 when the EVC flit is traversing the router. The bypass setup block informs the SVA to stop granting any input VC in i_0 and any input VC which is requesting an output VC in o_2.

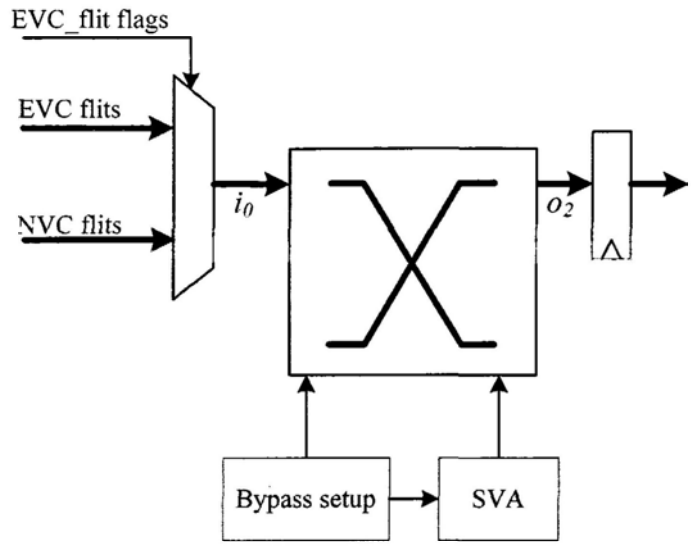


Figure 4.5. Express bypass router microarchitecture.

4.4.2 Aggressive express bypass router

Figure 4.6 shows the EVC bypass router microarchitecture under aggressive express pipeline. The 2-1 multiplexes are added not at i_0, but at o_2. As an EVC flit arrives, it advances directly to the multiplexes and thus skips the ST stage. In this architecture, the crossbar input port i_0 is not reserved. It means that it is possible for a flit buffered in input VCs of i_0 to traverse the crossbar while an EVC flit is going through the router. The crossbar output port o_2 may not necessarily be reserved, depending on when the EVC flit comes.

- A NVC flit f_1 traversed the crossbar and was saved in the ST pipeline registers at o_2 in the previous cycle. In the current cycle, an EVC flit f_2 arrives and occupies the output channel. Hence, f_1 remains being buffered in the ST pipeline registers. In this case, o_2 must be reserved to prevent a new NVC flit f_3 from going to it. Otherwise, f_1 will be overridden by f_3.

- No NVC flit was saved in the ST pipeline registers at o_2 in the previous cycle. In this case, there is no need to reserve o_2, which allows a NVC flit to travel the crossbar to o_2 when an EVC flit is passing the router.

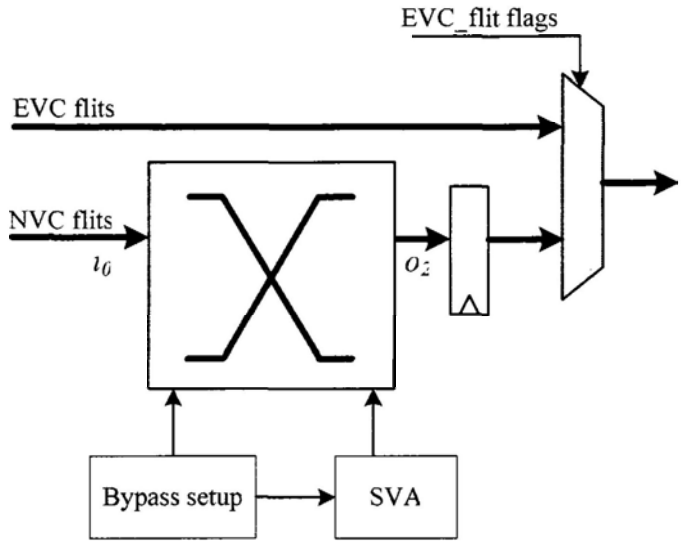


Figure 4.6. Aggressive express bypass router microarchitecture.

4.5 CUSTOMIZED BUFFER ARCHITECTURE

As an EVC path is inserted, the buffer architectures of the input ports along this path should be changed since the expected traffic will also change. As shown in Figure 4.7, at the EVC sink port, some EVC lanes are added to store EVC flits. Meanwhile, some NVC lanes are removed because less traffic is expected on the NVC lanes. In addition, since EVC flits do not take up buffers in the EVC bypass input ports, the number of NVC lanes in these bypass input ports can be reduced for the same reason. *Thus, it is important to determine how many EVC lanes will be inserted so that the number of NVC lanes in the EVC sink port and in the EVC bypass input ports can be optimized.* Buffer optimization can then be extended to other normal routers with small performance penalties.

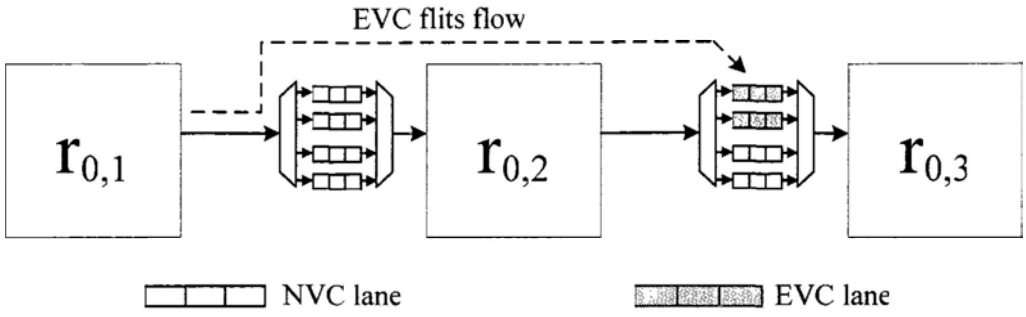


Figure 4.7. EVC flits flow.

We propose a statistical approach to customize buffer architecture. This idea is based on two observations. First, buffer utilizations are widely different at various input ports because they have different traffic characteristics. Second, assuming that there are four buffer lanes in an input port but only two of them are utilized in most cases, say 95%, removing two buffer lanes from this port will hardly affect network performances.

Figure 4.8 presents a procedure to customize the number of EVC/NVC lanes for input ports. Buffer architecture is initially set to be uniform. In step 2, we calculate p_i for each input port. p_i is the probability that totally i EVC/NVC lanes are utilized in an input port. In step 3, *we determine the minimum number of EVC/NVC lanes for each input port so that the accumulative utilization probability at each input port reaches th*. This process is represented in equation (10) where *th* is an adjustable threshold, and M is the maximum number of EVC/NVC lanes allowed for an input port. The parameter M is a design constraint because a large M generates costly control logic. In steps 4 and 5, we evaluate network performances for the new customized buffer architecture. If performances are acceptable, output the customized buffer architecture in step 6. Otherwise, increase *th* and rerun steps 3, 4, and 5. Likewise, the buffer depth for each input port can be customized. In this case,

we need to have the probability that totally i buffer spaces are utilized in an EVC/NVC lane.

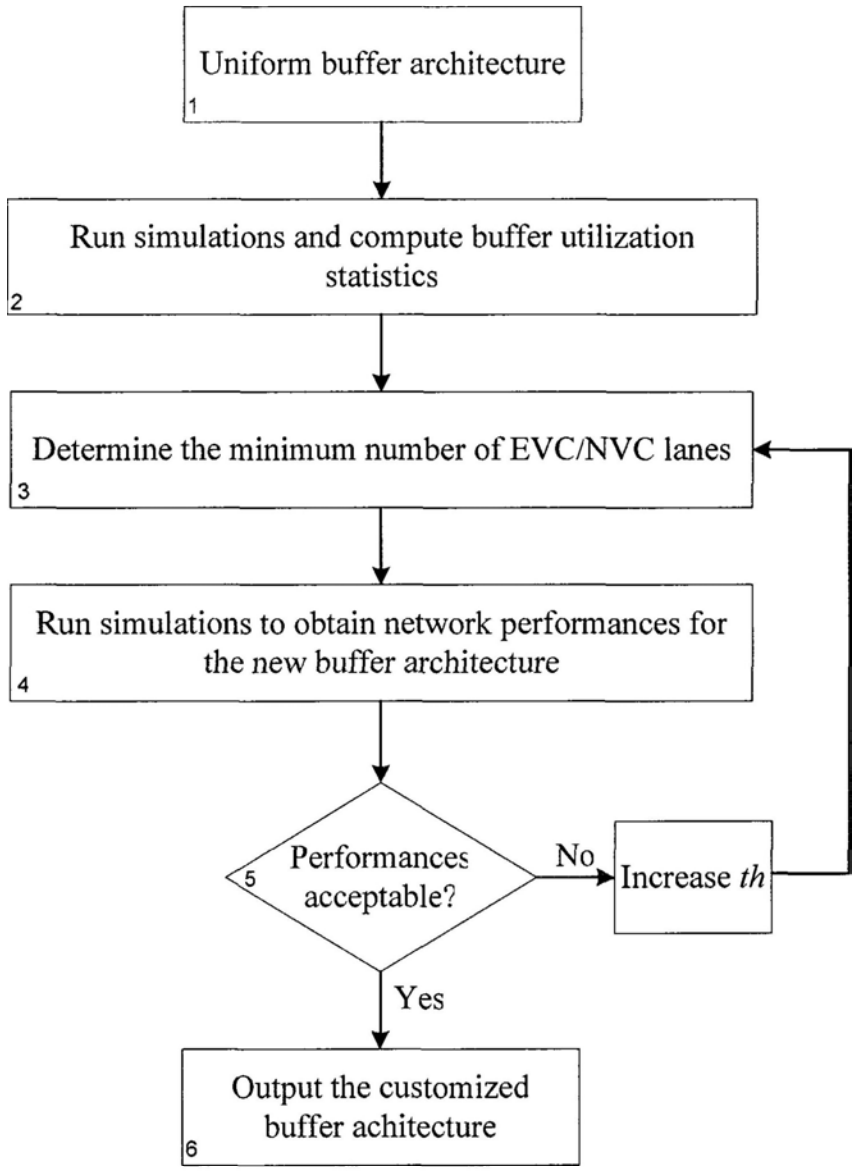
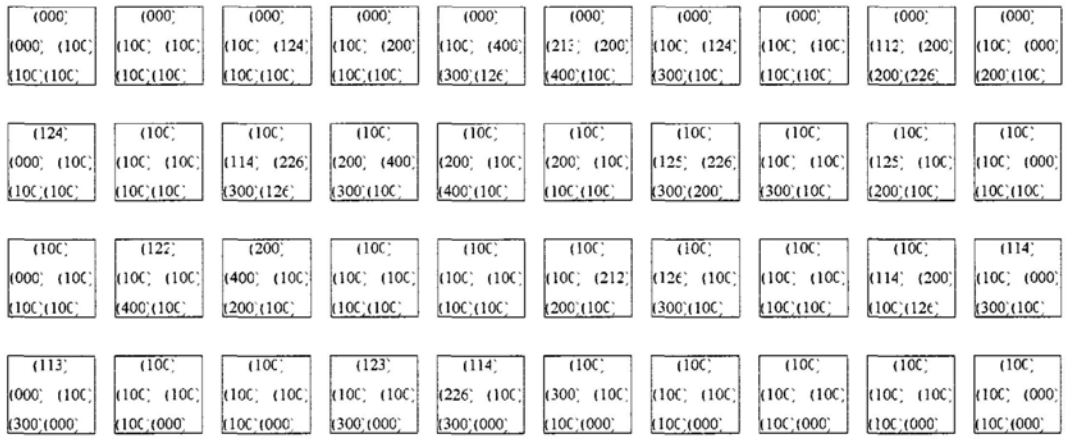


Figure 4.8. Buffer customization flow graph.

$$\min(m), \text{ subject to } \sum_{1}^{m} p_i \geq th \text{ and } 1 \leq m \leq M \quad (10)$$

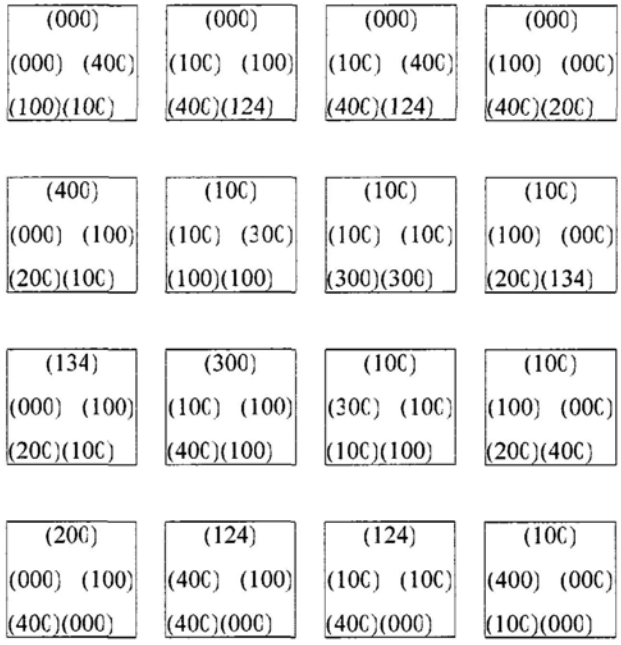
We compared the uniform buffer architectures and the customized buffer architectures for the TRIPS OCN under swim traffic and a 4×4 mesh under transpose traffic. In the uniform architecture, each input port has four VC lanes in total, with each lane accommodating four flits. If an input port is an EVC sink port,

four lanes will be equally divided into two EVC lanes and two NVC lanes for the swim traffic but into three EVC lanes and one NVC lane for the transpose traffic. The customized architectures are described in Figure 4.9 (a) and (b) respectively. A threshold of 0.95 is used. The configuration for a local input port is given at the corner of each box while the configurations for other input ports are placed against the related edges. The number of EVC/NVC lanes for each input port is fully customized. The depth of EVC lanes is also customized for each input port, ranging from the maximum of six flits to the minimum of two flits. However, NVC lanes of all the input ports have the uniform depth of four flits.



$(d_1d_2d_3)$ d_1: the number of NVC lanes; d_2: the number of EVC lanes; d_3: the depth of an EVC lane;

(a)



(b)

Figure 4.9. Customized buffer architectures for the TRIPS OCN swim traffic (a) and a 4×4 mesh with transpose traffic (b).

Table 4.1 and Table 4.2 summarize the results. For the swim traffic, the total buffer size drops from 2752 flits to 1112 flits, showing a big reduction of 59.59%. For the transpose traffic, the total buffer size is reduced by 31.64%. There is no deterioration in saturation throughputs for both traffics. It means that the customized buffer architecture can achieve the same saturation throughput with much less buffer than the uniform buffer architecture. Our synthesis results show that buffers account for 78.95% of router area when a flit width of 69 bits is used. Thus, a customized buffer architecture can save router area significantly with no performance degradation. Besides, a NoCs consumes a certain amount of standby power that mainly consists of leakage power and clock tree power. Thus, a customized buffer architecture can save a large amount of standby power because buffers usually dominate the leakage and clock tree power consumptions. Furthermore, control logic of routers becomes smaller as buffers reduce, which leads to more area and power savings.

Table 4.1. RESULTS FOR THE TRIPS OCN SWIM TRAFFIC.

	Saturation throughput	NVC buffers	EVC buffers	Total buffers
Uniform	0.175	2536	216	2752
Customized	0.175	936	176	1112
Reduction	0.00%	63.09%	18.52%	59.59%

Table 4.2. RESULTS FOR THE TRANSPOSE TRAFFIC

	Saturation throughput	NVC buffers	EVC buffers	Total buffers
Uniform	0.249	184	72	256
Customized	0.249	119	56	175
Reduction	0.00%	35.33%	22.22%	31.64%

Not only applicable to a single application system, the customized buffer idea can be applied to a multi-application system through reconfiguration. In a multi-application system, the buffers will be sufficiently sized so that all intended applications can be adequately served. Then, customized buffer architecture for each application will be obtained off-line and be stored in a look-up table. As the system switches from one application to another, it looks up the customized buffer architecture from the table for the new application and logically reconfigures the physical buffers through power gating technique to turn on/off some buffers. In this scenario, only power consumption is benefited because buffers are not physically removed, but are only turned off. Since the customized buffer architectures are calculated off-line and reconfigured on-line, no complex logics are needed to monitor network loads and to adjust buffer architectures dynamically. Thus, area and power costs are expected to be small. Compared to customized topologies [13-16, 35], the customized buffer does not affect the structures of both routers and links so is easy to be reconfigured. In short, the customized buffer can complement a regular topology

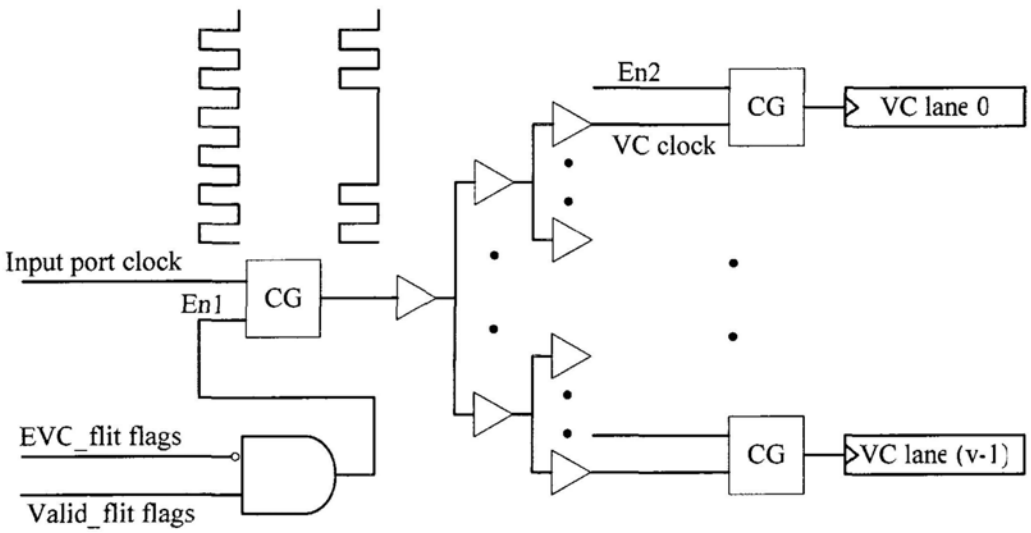
such as a mesh to support a large range of applications with reduced power consumptions.

4.6 LOW POWER TECHNIQUES

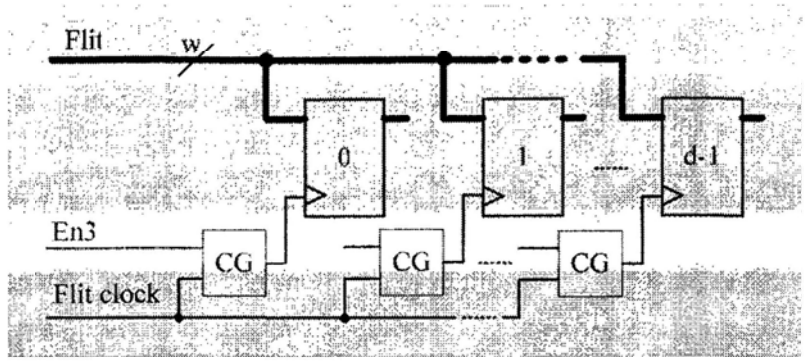
The total power consumed when a flit proceeds through a router is expressed in equation (1). Ideally, as an EVC flit skips an EVC bypass router, the E_{router} for this flit can be entirely saved. But this is not the case in real designs. In this section, we present low power techniques to save power for router components.

4.6.1 Buffers

Input buffers are partitioned by virtual channel, with a separate FIFO being provided for each VC. Each FIFO is implemented as a register file. In our example, an input port has v VC lanes and each VC lane is capable of buffering d w-bit flits.



(a)



(b)

Figure 4.10. Clock gating at different levels. (a). Port level and VC level. (b). Flit level.

E_{wrt} mainly consists of two components: clock power and data input power. Clock power includes both power dissipated by clock tree and power consumed by clock pins of registers. Clock gating (CG), a well-know low-power technique, is often applied to reduce clock power of buffers. As can be seen in Figure 4.10, CG may be used in three levels: port level, VC level, and flit level. For simplicity, the inputs for only one CG cell are shown at each level. A port level CG cell enables/disables the clock for all registers in a port ($v \times d \times w$ bits). A port clock is enabled (En_1) when there is a valid NVC flit at the input port. Once a port clock is turned off, it saves a large amount of power. A VC level CG cell enables/disables the clock for registers in a VC lane ($d \times w$ bits). A VC clock is enabled (En_2) when a valid NVC flit is being stored to any flit slot in the VC lane. When a VC clock is disabled, it saves a moderate amount of power. A flit level CG cell enables/disables the clock for registers of a flit (w bits). A flit clock is enabled (En_3) when a valid NVC flit is addressing the particular flit slot. Disabling the flit clock saves only a relatively small amount of power. However, the likeliness that a particular level of clock can be turned off goes the opposite way. In other words, a flit level CG has the highest probability of imposing itself while a port level CG has the least probability. Since

actual probabilities vary largely with network load, adopting an appropriate strategy for a particular CG level highly depends on traffic scenarios.

To evaluate the effectiveness of the three CG levels, we built a 5-port router, with 4 FIFOs per port, 4 flits per FIFO, and 69 bits per flit. The router was physically implemented in a conventional ASIC flow. Total power consumed by buffers of a router was calculated based on back-annotated netlist and switching activities generated from simulations. All simulations were run at 250MHz using uniform traffic. Figure 4.11 shows the buffer power consumption for different CG levels. As expected, all three CG levels save power in all traffic conditions. However, port level CG only works well when injection rate is extremely low. As injection rate increases, the power consumption increases quickly because the probability to disable a port level clock reduces sharply. Both VC level CG and flit level CG are better than port level CG for most injection rates. Also, VC level CG is better than flit level CG when injection rate is smaller than 0.2. After that, flit level CG is better. On average, VC level CG and flit level CG reduce power by 78.1% and 80.2% respectively compared to the case that no CG is applied. Flit level CG is used in the implementation exercise.

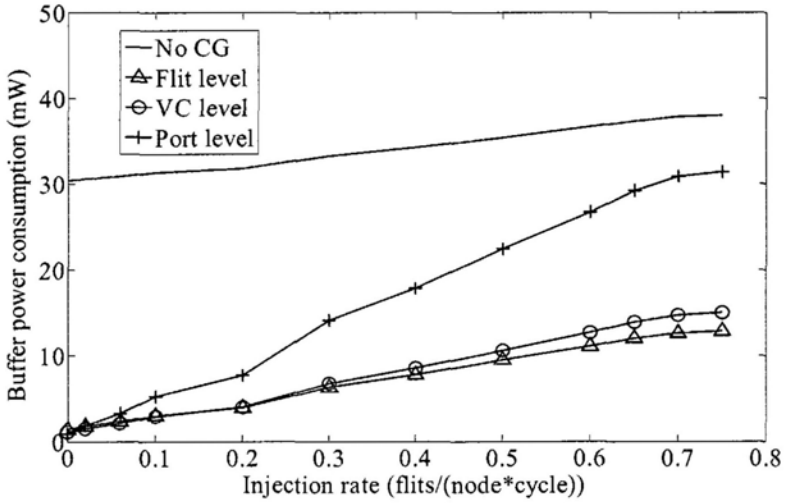


Figure 4.11. Comparison of different CG levels.

In a FIFO, only registers for one flit will change values at each clock cycle while the remaining registers are not altered. To reduce power consumed by the data inputs to the remaining registers, we simply change logic low disabled CG cells to logic high disabled CG cells. This works because the master latch of a flip-flop is not transparent when a logic high clock signal is applied. As shown in Figure 4.12, logic high disabled CG saves more power than logic low disabled CG in all traffic conditions. On average, 13.35% more power is saved. However, using logic high disabled CG cells leads to tight timing constraints on CG enable signals. The enable signal is just required to be ready before the end of the whole previous cycle for a logic low disabled CG cell because the enable signal can be passed to the output of the latch when the clock signal is logic low (Figure 4.1 (a)). However, it has to be ready in the first half phase of the previous clock cycle for a logic high disabled CG cell because it can only be passed to the output of the latch when the clock signal is logic high (Figure 4.1 (b)). Fortunately, the enable signal for the flit level CG cell of a flit slot can be simply obtained by checking whether a valid NVC flit is destined for this slot, and thus is early enough.

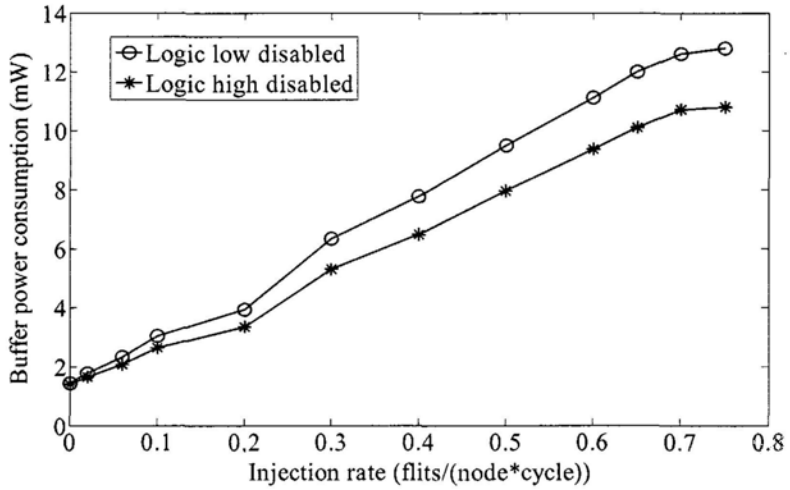


Figure 4.12. Compare of different CG cells.

4.6.2 Control logic

When the head flit of a new packet arrives at a router, RC is performed to determine the output port in the direction of the destination. The body/tail flits of the packet will follow the head flit to the same output port. Therefore, no RC operations are needed for body/tail flits. As a result, operand isolation can be applied to isolate RC blocks for body/tail flits. Likewise, even the RC operation for the head flit can be isolated if it is an EVC flit.

When an EVC flit arrives at a bypass router, it skips VA and SA operations. As mentioned in Section 4.4, both i_0 and o_2 of the crossbar are reserved for the EVC flit in the case of express pipeline. Thus, the router will ignore the results of the arbiters serving i_0 and o_2. Likewise, the router does not use results from the arbiter serving o_2 in the case of aggressive express pipeline. We isolate the arbiters in these conditions to reduce their power consumptions.

Figure 4.13 presents how operand isolation is implemented. An AND2 gate is inserted for each input signal. When isolation enable signal is asserted, all isolated input signals remain at logic 0. Therefore, no power is consumed by the block because there are no switching activities on its isolated input signals. To isolate a RC block, the isolation enable signal is asserted when a flit is not a NVC head flit. To isolate an arbiter, the isolation enable signal is asserted when the corresponding input/output port is reserved. These enable signals have to be ready before the corresponding input signals to avoid large delay overhead.

We evaluated the effectiveness of operation isolation for a RC block that implements a simple XY routing algorithm using the same method in the clock

gating evaluations. Figure 4.14 shows that applying operand isolation saves RC power by 62.85% on average.

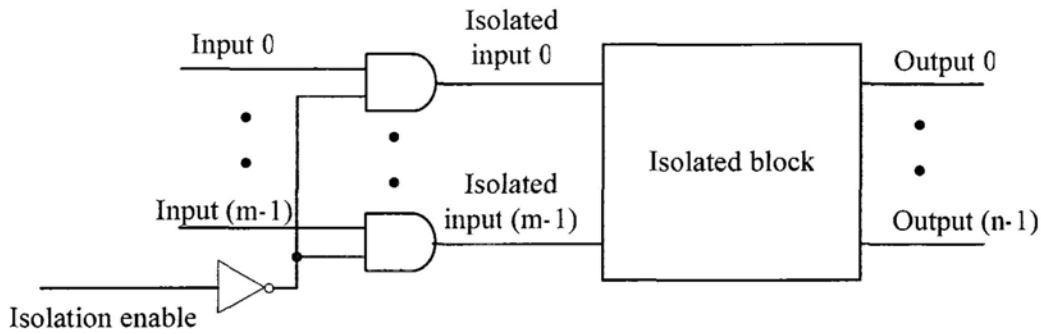


Figure 4.13. Operand isolation.

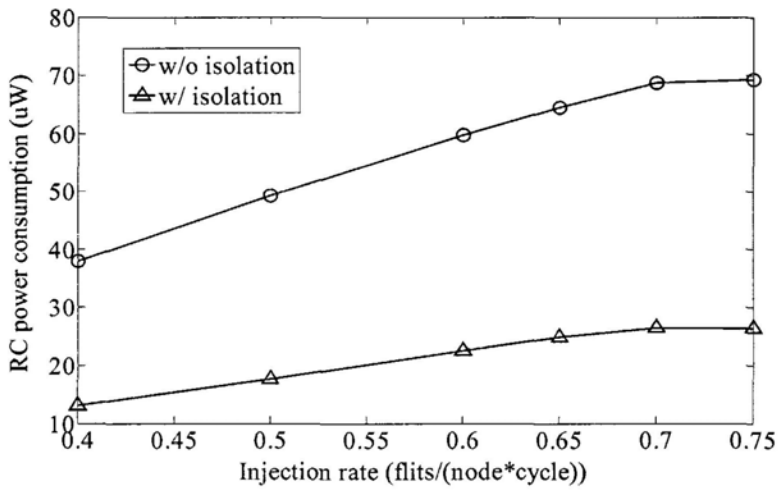


Figure 4.14. Power consumption for a RC block.

4.7 IMPLEMENTATION

4.7.1 Prototype architectures

We evaluated the baseline NoCs and the AS-EVC (application-specific EVC) NoCs for a realistic case (TRIPS OCN with the swim traffic) and a synthetic case (4×4 mesh with the transpose traffic). AS-EVC had been proved to have similar effectiveness on tested Minne-SPEC benchmarks like apsi, gzip, parser and swim in Chapter 3. Thus, without loss of generality, we selected the swim benchmark for the TRIPS OCN.

We prototyped a baseline NoCs and an AS-EVC NoCs for each case. The network and process parameters for the baseline NoCs are listed in Table 4.3. The AS-EVC NoCs has the same parameters as the baseline NoCs except that it uses aggressive express pipeline, the customized EVC paths (Table 4.4 for the swim traffic and Table 4.5 for the transpose traffic), and the customized buffer architecture (Figure 4.9 (a) for the swim traffic and Figure 4.9 (b) for the transpose traffic).

Table 4.3. BASELINE NETWORK AND PROCESS PARAMETERS

Traffic	TRIPS OCN swim; transpose
Topology	10×4 mesh; 4×4 mesh
Flow control	Virtual channel
Routing	XY
Buffer management	Credit-based
Pipeline	Non-express pipeline
Router radix	5
Buffer architecture	4 VCs per port, 4 flits per VC
Packet length	4 flits
Flit size	32 (payload) + 4 (overhead)
Technology	130nm, HS
Frequency	250MHz
Tile size	$1mm \times 1mm$

4.7.2 Customized EVCs insertion

We inserted EVC paths in an application-specific manner. Firstly, all possible EVC paths were evaluated based on communication volumes of any two tiles and power models. Then, EVC paths were inserted by a greedy algorithm subject to several insertion rules. The details for the insertion method are presented in Chapter 3. In this way, we inserted totally 23 EVC paths for the swim traffic and 6 EVC paths for the transpose traffic. Table 4.4 and Table 4.5 list these EVC paths with each path

identified by its source router and sink router indexes (the left-bottom router is represented as r_{00} throughout the thesis).

Table 4.4. EVC PATHS FOR THE SWIM TRAFFIC

source	sink	source	sink	source	sink	source	sink
r_{12}	r_{16}	r_{32}	r_{20}	r_{26}	r_{28}	r_{31}	r_{35}
r_{15}	r_{22}	r_{00}	r_{04}	r_{38}	r_{19}	r_{19}	r_{15}
r_{36}	r_{32}	r_{05}	r_{18}	r_{34}	r_{04}	r_{22}	r_{11}
r_{22}	r_{26}	r_{18}	r_{38}	r_{20}	r_{22}	r_{16}	r_{18}
r_{26}	r_{22}	r_{33}	r_{03}	r_{28}	r_{26}	r_{35}	r_{38}
r_{04}	r_{34}	r_{38}	r_{36}	r_{20}	r_{00}		

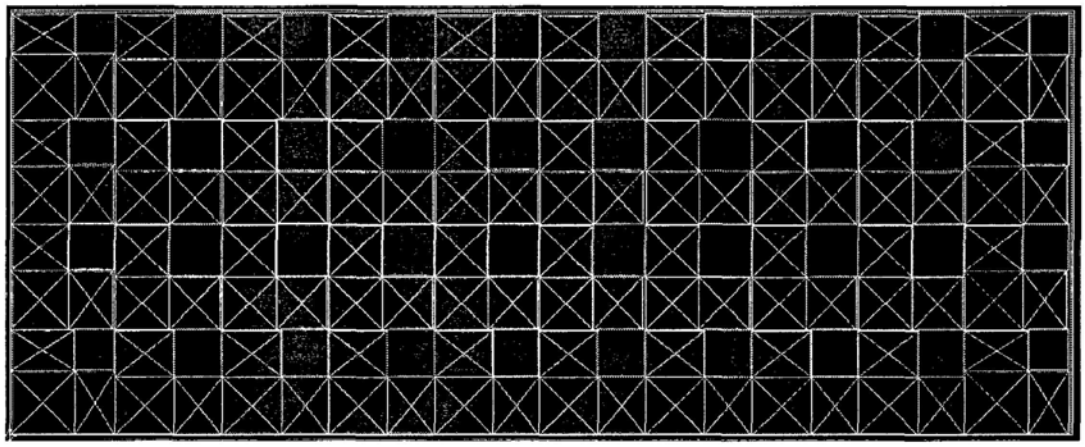
Table 4.5. EVC PATHS FOR THE TRANSPOSE TRAFFIC

source	sink	source	sink	source	sink	source	sink
r_{32}	r_{10}	r_{01}	r_{23}	r_{10}	r_{32}	r_{23}	r_{01}
r_{20}	r_{31}	r_{13}	r_{02}				

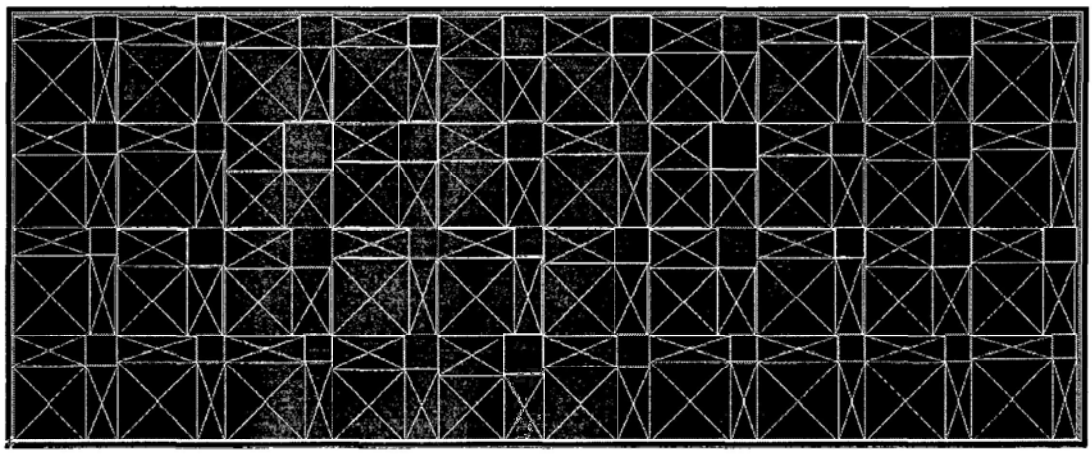
4.7.3 Physical implementation

We performed floorplanning, place and route for the NoCs using Synopsys Astro. Without loss of generality, we assumed that each router was located at the right-top corner of a tile. Thus, location of a router was determined when the corresponding tile was located. Each router was assigned to a rectangular *plangroup* where all logic cells for this router were only allowed to be placed within the *plangroup* area. In a realistic NoCs-based chip, each tile includes a router and an IP core. However, we implemented only a router for each tile because designing and implementing realistic IP cores require two much effort and are unnecessary in this exercise. Large spaces supposedly occupied by an IP core are included in each tile. To make the layout more realistic, we intentionally built *hard blockages* to prevent using these available spaces and reserved only a few rows for placing top-level standard cells (for example, registers for pipelined physical links) and routing between routers.

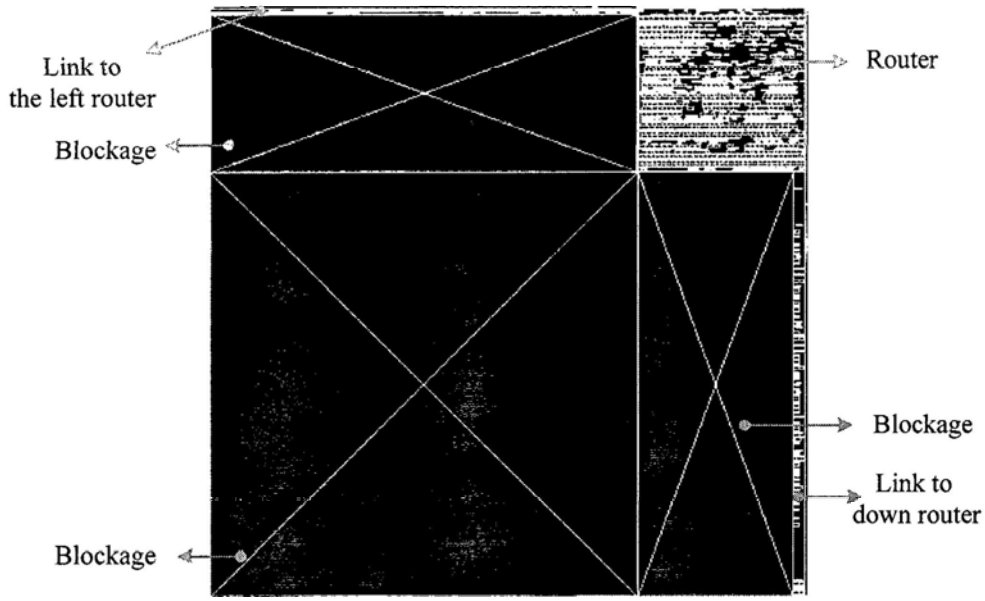
For clarity, only the layout micrographs for the TRIPS OCN swim traffic are shown in Figure 4.15. Since uniform buffer architecture was used, the routers in the baseline NoCs have the same size except for those at the corners or along the borders of the 10×4 mesh. But, the routers in the AS-EVC NoCs are varied in sizes due to customized EVCs insertion and customized buffer architecture. Only two links are shown in a single tile for simplicity. *To the best of our knowledge, this is the first time a near-realistic regular mesh NoCs with variable number of VCs and depths at different input ports has been physically implemented.*



(a)



(b)



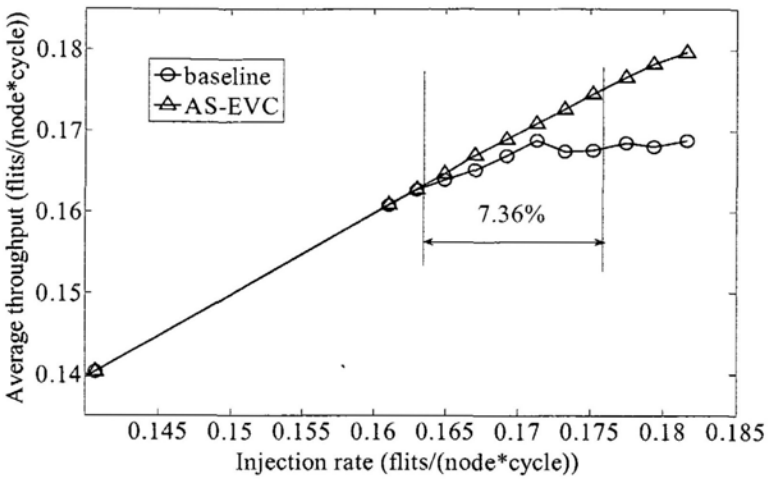
(c)

Figure 4.15. Layout micrographs. (a). The baseline NoCs. (b). The AS-EVC NoCs. (c). A single tile.

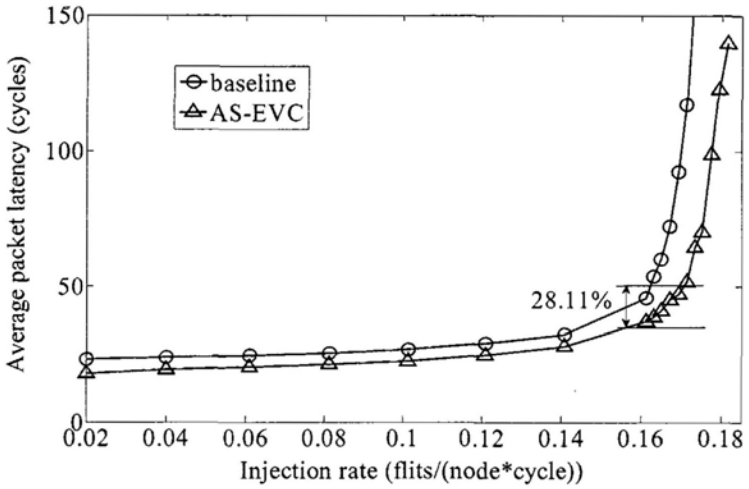
4.8 RESULTS

4.8.1 Network performances

We evaluated average packet latency and average throughput using SystemVerilog models. Figure 4.16 shows results for the swim traffic. The AS-EVC outperforms the baseline at all injection rates. Saturation is improved from 0.163 to 0.175 flits/(node*cycle), showing a 7.36% increase. Likewise, the latency at 0.163 flits/(node*cycle) drops from 53.62 to 38.55 cycles, giving a 28.2% reduction. Figure 4.17 demonstrates results for the transpose traffic. The AS-EVC has lower latency than the baseline at low and moderate injection rates. However, they have similar latencies at high injection rates and thus similar saturated throughputs.

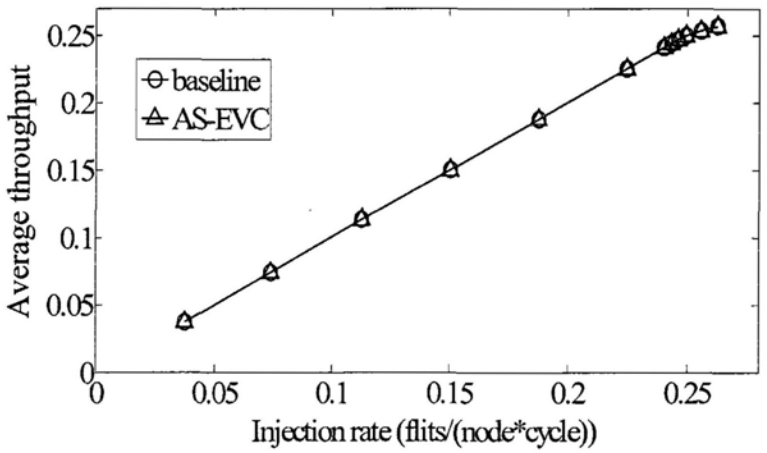


(a)

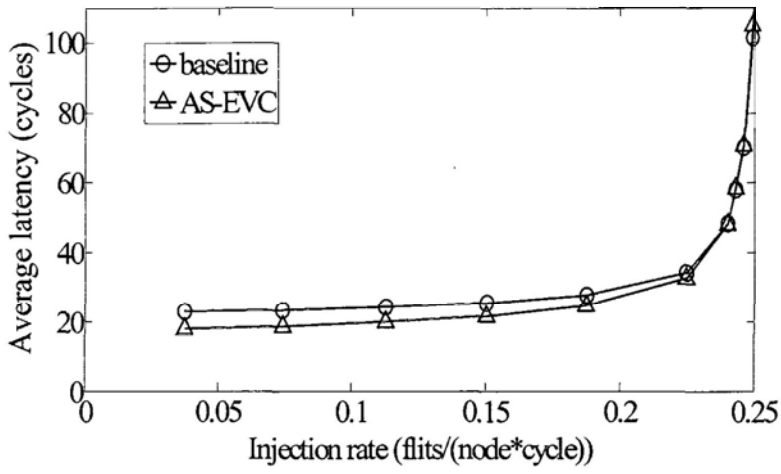


(b)

Figure 4.16. Average throughput (a) and average packet latency (b) for the swim traffic.



(a)



(b)

Figure 4.17. Average throughput (a) and average packet latency (b) for 4×4 mesh with the transpose traffic.

4.8.2 Area, power and energy

The performance improvements are obtained by using much smaller area and power costs. Synopsys DC results under worst case synthesis scenario show that the gate count decreases from 1095.49K to 538.45K (a reduction of 50.85%) for the 10×4 mesh with the swim traffic and drops from 405.24K to 239.72K (a reduction of 40.84%) for the 4×4 mesh with the transpose traffic.

Power figures were obtained from Synopsys PrimeTime PX using UMC 130nm library files, post-layout netlist, extracted RC, and post-layout switching activities as inputs. Standby and total power consumptions were calculated using the switching activities at zero-load and saturation point (0.163 flits/(node*cycle) for the swim traffic and 0.249 flits/(node*cycle) for the transpose traffic) respectively. Stream powers were obtained by subtracting standby powers from total powers.

Table 4.6 and Table 4.7 present power consumptions of the complete NoCs. Compared to the 10×4 baseline NoCs, the 10×4 AS-EVC NoCs reduces total power from 298.9mW to 196.5mW (34.26%), with total standby power reducing from

145.0mW to 89.6mW (38.21%) and total stream power decreasing from 153.9mW to 106.9mW (30.54%). Also, compared to the 4×4 baseline NoCs, the 4×4 AS-EVC NoCs reduces total standby power by 19.8mW (34.49%) and total stream power by 26.7mW (32.56%), and thus total power by 46.5mW (33.36%). The standby power savings are mainly resulted from large area reduction through customized buffer architecture. The stream power reductions are caused by a combination of area reduction, customized EVC paths to virtually bypass intermediate routers for many packets, and low-power techniques. It is clear that routers dominate in both standby power and stream power. This is mainly because a smaller flit size is used due to a large flit size will require too much computing resource in placing and routing the entire NoCs. In addition, total power and stream power savings reported here are much larger than the post-synthesis results reported in Chapter 3. For example, total power and stream power savings of all routers are 38.68% and 32.58% respectively for the swim traffic while they are 15.65% and 20.09% in respective in Chapter 3. The main reason is that experiments in Chapter 3 used uniform buffer architecture instead of customized buffer architecture. The customized buffer architecture not only reduces standby power of routers, but also increases stream power reduction of routers because load capacitances decreases as logic area reduces. Thus, we suppose that much larger power reductions will be achieved for other traffic patterns tested in Chapter 3 after their buffer architectures are customized.

Table 4.6. POWER CONSUMPTIONS FOR THE TWO ENTIRE 10×4 NOCS FOR THE SWIM TRAFFIC.

Component	Power (mW) Baseline | AS-EVC		
	Standby	Stream	Total
Routers	101.6 | 54.5	128.0 | 86.3	229.6 | 140.8
Others	43.4 |35.1	25.9 | 20.6	69.3 | 55.7
Total	145.0 | 89.6	153.9 | 106.9	298.9 | 196.5

Table 4.7. POWER CONSUMPTIONS FOR THE TWO ENTIRE 4×4 NOCS FOR THE TRANSPOSE TRAFFIC

Component	Power (mW) Baseline | AS-EVC		
	Standby	Stream	Total
Routers	39.3 | 23.7	70.6 | 45.2	109.9 | 68.9
Others	18.1 |13.9	11.4 | 10.1	29.5 | 24.0
Total	57.4 | 37.6	82.0 | 55.3	139.4 | 92.9

Table 4.8 shows area and standby power breakdowns for the router r_{24}. Area of buffers reduces from 20635 to 5149 (75.05%) and standby power drops from 1.180mW to 0.266mW (77.46%) because a large number of NVCs/EVCs are removed by customization. Since design complexity of the SVA highly depends on number and depth of NVCs/EVCs, its area and standby power decreases significantly by 70.53% and 44.54% respectively. The crossbar and related logic consume little standby power because there are few registers in them. Area for the top-level clock network and logic is not reported in synthesis results. But, we can see that they consume a large amount of standby power (power is reported in post-layout results) for both NoCs. Reduction of total buffers simplifies much the top-level clock network, and then reduces its standby power from 1.077mW to 0.498mW.

Table 4.8. AREA AND POWER BREAKDOWNS FOR THE ROUTER r_{24} (BASELINE | AS-EVC)

Component	Area (gate count)	Power (mW)
Buffers + logic	20635 | 5149	1.180 | 0.266
Crossbar + logic	2859 | 2679	0.058 | 0.057
SVA + logic	7154 | 2108	0.485 | 0.269
Top-level clock network and logic	NA	1.077 | 0.498
Total	30915 | 9994	2.800 | 1.090

Table 4.9 presents stream flit energy breakdown for the router r_{24}. Stream flit energy for each component was determined through multiplying the component's

stream power by simulation time and dividing by the number of flits travelling the router. For the AS-EVC case, we assumed that a NVC flit consumes the same stream energy as an EVC flit to simplify calculations. Calculation errors are small because NVC flits travelling the router r_{24} account for only 4% of total flits. It can be seen that total stream energy is 16.15pJ and 3.62pJ respectively when a flit travels or bypasses the router, showing a 77.59% reduction. Given effective clock gating and area reduction, a flit consumes 88.75% less energy when it bypasses the buffers. Likewise, combination of clock gating, operand isolation and area reduction saves the SVA energy by 96.74%. As seen in Figure 4.6, when a flit bypasses the crossbar, it skips the 5x5 switch fabric but has to go through the wires and the 2-1 multiplexes. As a result, only 46.3% energy is saved.

Table 4.9. STREAM FLIT ENERGY BREAKDOWN FOR THE ROUTER r_{24}.

Component	Baseline (pJ)	AS-EVC (pJ)	Reduction (%)
Buffers + logic	5.18	0.58	88.75
Crossbar + logic	4.87	2.62	46.3
SVA + logic	5.86	0.19	96.74
Top-level clock network and logic	0.24	0.23	3.75
Total	16.15	3.62	77.59

4.9 SUMMARY

In this chapter, we have proposed methods to design and to implement a NoCs supporting the EVCs technique with low power as the main objective. We have described cost-efficient hardware components, optimized buffer architectures, creative use of low power techniques and near-realistic ASIC prototypes to demonstrate how the EVCs flow control can be best exploited in practice.

Detailed physical implementations show that the AS-EVC NoCs has much smaller power and area costs than the baseline NoCs. Furthermore, significant savings are obtained with no network performance penalties but a small sacrifice in attaining the maximum speed.

Given the impressive results, we plan to implement realistic NoCs-based systems using the EVCs technique. Also, based on the power consumption results extracted from physical implementations, more accurate power saving and cost models than those used in Chapter 3 are expected to be built to help designers estimate power savings of EVCs insertion in early design stage.

CHAPTER 5. CONCLUSIONS

As interconnection networks are shifted from off-chip domain to on-chip domain, a critical challenge is to keep their design costs (area and power) small. This thesis concentrates on this challenge to explore cost-efficient NoCs architectures and design cost-efficient NoCs components.

5.1 CONTRIBUTIONS

This thesis focuses on cost reduction of routers because routers are much more costly than links when packet switching and virtual channel flow control are applied. The contributions are in two orthogonal aspects: router microarchitecture and network architecture.

In terms of router microarchitecture, we studied cost-efficient allocators for a router. Through investigating simulation results of a complete NoCs, we found large opportunities to reduce design costs of the generic virtual channel and switch allocators and then proposed two low-cost allocators: the look-ahead allocator and the combined switch and VC allocator. Evaluation results show that the proposed allocators can significantly reduce area and power costs compared to the generic allocator architecture without performance penalties. The cost-efficient allocators are orthogonal to cost-efficient router datapath components like buffer and crossbar, and thus combing them together will reduce more design costs for a router.

In terms of network architecture, we studied the express virtual channel flow control along two directions. The first direction is the high-level application-specific methodology to achieve maximum power savings for given applications. Based on calculations of communication volumes between routers and simple high-level power

models, the high-level method can quickly determine power-efficient EVC paths and thus is useful to explore a large design space in early design stage. Evaluation results for a wide range of design parameters and traffic patterns demonstrate that AS-EVC NoCs are more power efficient than both the baseline and the static EVCs NoCs.

The second direction is to study design and implementation issues for low-power NoCs supporting the EVC flow control. It includes four aspects. First, we designed power-efficient hardware components for EVC networks. Second, we optimized buffer architectures to reduce both area and power costs for EVC networks by a statistical approach. Third, we explored conventional low-power techniques like clock gating and operand isolation for EVC routers. Four, we performed accurate and detailed evaluations by ASIC implementations. Results show that up to 34.26% NoCs power is saved by the proposed techniques.

5.2 FUTURE WORK

There are many interesting topics for future directions inspired by the work described in the thesis.

Two topics may be further studied based on the results of low-cost allocators. On the one hand, the generic VA is actually a separable iSLip allocator. This architecture is widely applied in NoCs domain because its low implementation costs. However, there are other advanced allocator schemes like separable lonely output allocator and wavefront allocator that have higher matching efficiency and are used in off-chip networks. Thus, it is interesting to study these advanced allocators and reduce their design costs for NoCs. On the other hand, large opportunities to simplify the generic VA are identified by simulations for the entire NoCs. They would never be found if we just run simulations for the allocators themselves. Thus, the research method of

studying a component in the context of the entire NoCs can be used to explore other components.

We presented an application-specific methodology to smartly insert power-efficient EVC paths for NoCs. However, several problems can be further studied to optimize the methodology. First, modify the EVC insertion rule to allow overlapping of EVC paths in some cases. It can improve flexibility of EVC insertion, and thus allow some power-efficient EVC paths to be inserted. Second, simple power models currently used in the AS-EVC method are lack of accuracy. The energy cost when a flit travels an EVC source router and the energy saving when a flit skips an EVC bypass router can be accurately obtained from the ASIC implementations. Thus, more accurate power models can be built to estimate power saving for an EVC path. Third, the greedy algorithm currently used for EVC insertion is very simple, but not good to achieve the maximum overall value. Therefore, a more advanced algorithm is expected to be used for EVC insertion.

ASIC implementation results show that the AS-EVC NoCs save a large amount of area and power costs without performance penalties compared to the baseline NoCs. Inspired by them, an interesting future topic is to design a NoCs-based system for a realistic application with application-specific EVC paths. In addition, besides the single-application system, NoCs is expected to be widely used in systems that will support a variety of applications. Hence, it is a significant direction to study reconfigurable AS-EVC networks and reconfigurable customized buffer architectures to support multi-application systems.

In essential, EVC is a flow control technique where EVC flits and NVC flits share the same resources such as crossbar and physical channels. Thus, if physical links are the bottleneck in some hotspot regions of a network, using EVC paths in these

regions can not solve the bottleneck and thus will not improve network performances. Instead, adding costly EPC paths is effective to improve network performances in this case. As a result, it is an interesting topic to combine the EPC technique and the EVC technique for NoCs.

APPENDIX A. APPLICATION-SPECIFIC EVC INSERTION TOOL

The application-specific EVC insertion tool (named *EVCcustomize*) is accomplished through a Matlab program (as_evc_ideal.m). As shown in Figure A.1, *EVCcustomize* uses traffic pattern of an application (traffic_pattern.log) as the input and generate a list of EVC paths (evc_paths.log) and report parameter settings and estimated power savings (evc_insertion_report.log).

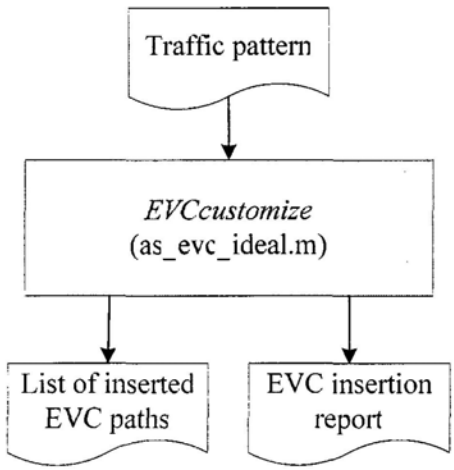


Figure A.1. The flow to use *EVCcustomize*

Many parameters are defined in *EVCcustomize*. They are described in Table A.1.

Table A.1. PARAMETERS IN *EVCcustomize*

Parameter name	Description
mesh_rows	The number of rows in a mesh network. In current, only mesh topology is supported.
mesh_columns	The number of columns in a mesh network
xb_to_router	The energy ratio of a crossbar to a router
energy_cost _percentage	The energy ratio of the overhead of an EVC source router to a normal router
pipeline_bypass	The type of bypass pipelines. 0: aggressive express pipeline, 1: express pipeline
maximum_interval	The allowable maximum interval of EVC paths
maximum_evc _perrouter	The maximum number of EVC paths (including both source EVC paths and sink EVC paths) that can be inserted in a

	router
bypass_evcrouter_enable	Whether an EVC source router can be bypassed. 1: enabled, 0: disabled
delta_energy_low_bound	The low bound (a threshold) for EVC insertion. An EVC path will not be inserted if its energy saving is smaller than the threshold.

Traffic pattern (traffic_pattern.log) for a $m \times n$ mesh is a $mn \times mn$ matrix. The value at the i^{th} row and j^{th} column of the matrix represents the traffic volume that are generated at the i^{th} router and consumed at the j^{th} router. Figure A.2 shows an example of the traffic pattern (traffic_pattern.log) for a 4×4 mesh with transpose traffic. It is a 16×16 matrix.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure A.2. An example of the traffic_pattern.log

Figure A.3 (a) shows an example of the list of inserted EVC paths (evc_paths.log) for the 4×4 transpose traffic. Each path is identified by its source router index (the first column) and sink router index (the second column). Figure A.3 (b) describes the index method used in the evc_paths.log file. The left-top router has an index of 1, the router at the top row and the second column has an index of 2, and so on.

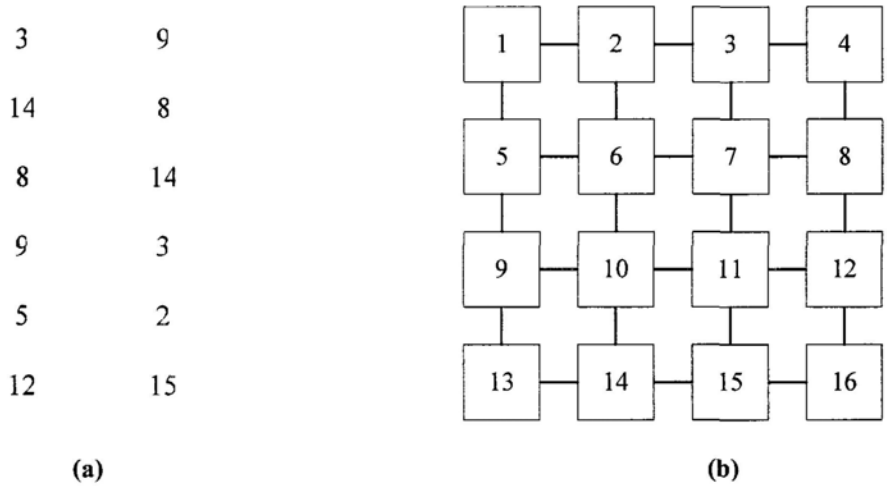


Figure A.3. Examples of the evc_paths.log (a) and the index method for routers (b)

Figure A.4 shows an example of the report file (evc_insertion_report.log) for the 4×4 mesh with transpose traffic. In the example, we investigate the sensitivity of energy saving to the parameter of maximum_interval. The first column is the name of the traffic pattern. The second and third columns describe numbers of rows and columns respectively for a mesh network. The fourth column represents the allowable maximum interval for EVC paths. We sweep this parameter in the example. The fifth column is the threshold of energy saving to insert EVC paths. Any EVC path that has a positive energy saving may be inserted in the example. The sixth column reports the energy saving calculated based on simple high-level energy models.

traffic_name	row	col	max_interval	del_energy_min	energy_saving
transpose	4	4	2	0.00000	0.27226
transpose	4	4	3	0.00000	0.28578
transpose	4	4	4	0.00000	0.37259
transpose	4	4	5	0.00000	0.37259
transpose	4	4	6	0.00000	0.37259

Figure A.4. An example of the evc_insertion_report.log

APPENDIX B. APPLICATION-SPECIFIC BUFFER CUSTOMIZATION TOOL

The application-specific buffer customization tool (named *BUFcustomize*) is implemented through Matlab programs (nvcs_customization_variable.m and evcs_customization_variable.m). Figure B.1 shows the flow to use *BUFcustomize*. Inputs include buffer utilization statistics files (generated in BUFFER_MODE RTL simulations described in Appendix.C), and the list of inserted EVC paths (evc_paths.log, only for EVC networks). Outputs are customized NVC architecture (customized_number_of_nvcs_inports.log) and customized EVC architecture (customized_number_of_evcs_evcpaths.log, only for EVC networks).

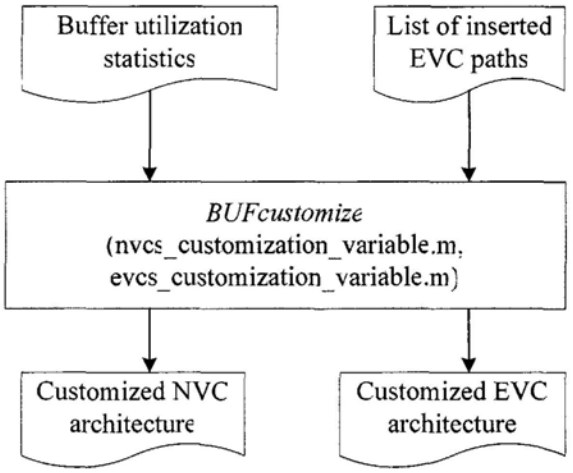


Figure B.1. The flow to use *BUFcustomize*

Table B.1 describes parameters used in *BUFcustomize*. It only supports mesh topologies with various network sizes now.

Table B.1. PARAMETERS IN *BUFcustomize*

Parameter name	Description
Parameters in nvcs_customize_variable.m	
MESH_ROWS	The number of rows in a mesh network
MESH_COLUMNS	The number of columns in a mesh network
MESH_NODES	The number of nodes (routers) in a mesh network

NVCS_PERPC	The uniform number of NVC lanes of each input port
NVC_BUFFER_DEPTH	The depth (specified in flits) of each NVC lane
TOTAL_BUFFERS_PERPC	Total buffers (specified in flits) of each input port
WARM_UP_PERCENTAGE	A value from 0 to 1. It represents what percentage of the simulation period is considered as warm up period.
CUMSUM_PROBABILITY _THRESHOLD	A value from 0 to 1. It specifies the threshold to determine how many VC lanes are enough for an input port.
Additional parameters in evcs_customization_variable.m	
EVCS_PERPC	The uniform number of EVC lanes of each EVC path
EVC_BUFFER_DEPTH	The uniform depth (specified in flits) of each EVC lane
TOTAL_BUFFERS_PEREVC	Total buffers (specified in flits) of each EVC path

Figure B.2 (a) and (b) show the customized NVC architecture and the customized EVC architecture for a 4×4 mesh with transpose traffic. The NVC results are a 16×5 matrix with each row representing a router and each column representing an input port (west, north, east, south, and local input ports from left to right). For example, the value of 4 at the 1st row and 3rd column means that there are 4 NVC lanes in the east input port of the router 1. The EVC results are a 6×1 matrix because there are totally six EVC paths in the network (Figure A.3 (a)).

0	0	4	1	1
1	0	1	1	4
1	0	4	1	4
1	0	0	2	4
0	4	1	1	2
1	1	3	1	1
1	1	1	3	3
1	1	0	1	2
0	1	1	1	2
1	3	1	1	4
3	1	1	1	1
1	1	0	4	2
0	2	1	0	4
4	1	1	0	4
1	1	1	0	4
4	1	0	0	1

(a)

3
3
2
2
2
2

(b)

Figure B.2. Examples of a customized_number_of_nvcs_inports.log (a) and a customized_number_of_evcs_evcpaths.log (b)

APPENDIX C. A FULLY-SYNTHESIZABLE PARAMETERIZED NOCS LIBRARY

C.1 INTRODUCTION

A fully-synthesizable parameterized NoCs library (named *NoClib*) is implemented through SystemVerilog and Verilog HDL. Therefore, *NoClib* can be used for both NoCs simulations and NoCs implementations on FPGA/ASIC. It has two significant features.

- It supports a wide range of router microarchitectures (pipelines). A baseline VC router with separate VC and switch allocations (Figure 1.4 (b)), a VC router with combined VC and switch allocation (Figure 2.9 (b)), a VC router with express pipeline (Figure 3.5 (b)), and a VC router with aggressive express pipeline (Figure 3.5 (c)). *NoClib* is the first NoCs library that supports the recently proposed express virtual channel flow control.
- It supports customized buffer architectures. In other words, it allows that different router input ports have different numbers of virtual channels and various EVC lanes have various depths of buffers. Furthermore, control logic of input ports will be also customized accordingly. *NoClib* is the first synthesizable library that can be used to implement NoCs with customized buffer architectures.

C.2 GLOBAL PARAMETERS

As described in Table C.1, many global parameters are used to define network topology, router microarchitecture, simulation environment, and implementation

environment. They are global parameters that are effective for an entire NoCs. All global parameters are defined in a file named parameters.v.

Table C.1. GLOBAL PARAMETERS IN *NoClib*

Parameter name	Description
Network parameters	
MESH_ROWS	The number of rows in a mesh network. Currently, only mesh topologies are supported.
MESH_COLUMNS	The number of columns in a mesh network
HOP_COUNT_BITS	The number of bits used to describe the interval of an EVC path. The most significant bit represents the direction and the rest bits represent the number of hops.
HX_BITS	The number of bits used to describe routing information along X dimension. The most significant bit represents the direction (0 for west and 1 for east) and the other bits represent the number of hops.
HY_BITS	The number of bits used to describe routing information along Y dimension. The most significant bit represents the direction (0 for north and 1 for south) and the other bits represent the number of hops.
Link/Channel parameters	
LINK_PIPELINE_STAGE	Determines number of pipeline stages of links between routers
CHANNEL_DATA_WIDTH	Width of payload data (specified in bits) in a flit. Thus, width of links between routers is the sum of width of payload data and width of control information (including VC identifier and flit type) in a flit.
Basic router parameters	
MAX_ROUTER_RADIX	Specifies the allowable maximum number of router input/output ports. At most five ports are supported now.
MAX_ROUTER_RADIX _INDEX_BITS	The number of bits used to describe the allowable maximum router radix
LOCAL_PORT_INDEX	The index for the local input/output port of a router that connects the local processing element and the router
MAX_TOTAL_VCS_PERPC	Specifies the allowable maximum number of

	VC lanes (including both NVC and EVC lanes) in router input ports. Currently, the supported range is from one to seven.
MAX_TOTAL_VCS_PERPC_INDEX_BITS	Number of bits used to represent the allowable maximum number of VC lanes.
NVC_BUFFER_DEPTH	Specifies uniform depth (in flits) of NVC lanes
NVC_BUFFER_DEPTH_BITS	Specifies number of bits used to represent the depth of NVC lanes
NVC_BUFFER_COUNTER_BITS	Determines number of bits for counting numbers of flits in NVC lanes
FLIT_TYPE_BITS	The number of bits used to represent types of flits. Currently there are four types of flits: INVALID, HEAD, DATA, and TAIL.
VCSR_G_BITS	The number of bits used to describe the status of VC lanes
UTURN_DISABLED	If set (1), U-turn is disabled in the routing algorithm. As a result, designs for other components can be simplified also. Currently only disabled U-turn is supported.
Router parameters for EVC flow control	
EVC_ENABLE	If set (1), the EVC flow control is supported
AGGRESSIVE_BYPASSPIPELINE	If set (1), aggressive express pipeline is used. Otherwise, express pipeline is used.
NORMAL, SOURCE, SINK, SOUSIN	Types of routers. SOUSIN means that the router is not only an EVC source router, but also an EVC sink router.
Simulation parameters	
PACKET_LENGTH	Length (specified in flits) of packets. Currently, only uniform-length packets are supported.
PACKET_TYPE_BITS	Specifies number of bits used to describe types of packets
PACKET_DATA_BITS	The number of bits of payload data (excluding all control fields like packet type, routing information, etc.) in a packet
DEBUG_MODE	If defined, all the assertions in the SystemVerilog codes will take effect to detect simulation errors in Questasim
LATENCY_MODE	If defined, the files for function simulations and network performance calculations will be outputted
BUFFER_MODE	If defined, the files to do statistics of buffer

	utilizations will be outputted
PATH	Specifies the path (a directory in the local computer) for all output files generated by RTL simulations
IRNAME	Defines the name of an injection rate
CONTROL_POINT_IR	Specifies the number to realize an injection rate[12]
Design and implementation parameters	
CLK_GATING_MODULE	If set (1), module-level clock gating is enabled
CLK_GATING_RTL	If set (1), RTL-level clock gating is enabled
LOGIC_HIGH_CG_FOR_RF	If set (1), use logic high disabled clock gating cells for register files. Otherwise, use logic low disabled clock gating cells.
MATRIX_CROSSBAR	If set (1), matrix-based crossbar architecture is used. Otherwise, mux-based crossbar architecture is used
MATRIX_CROSSBAR _SEGMENT	If set (1), enable to segment matrix-based crossbars
SEGMENT	Determines number of segments of a matrix-based crossbar

[12] We realize an injection rate by comparing a random number with a pre-set threshold (CONTROL_POINRT_IR). A packet at a router will be injected if the random number is no bigger than the threshold. For example, assuming the random number is in the range from zero to 100,000, the PACKET_LENGH is 4, and we want to set an injection rate of 0.5 flits/(node*cycle), the threshold (CONTROL_POINRT_IR) is calculated as 100000 * 0.5 / 4.

C.3 SIMULATION FRAMEWORK

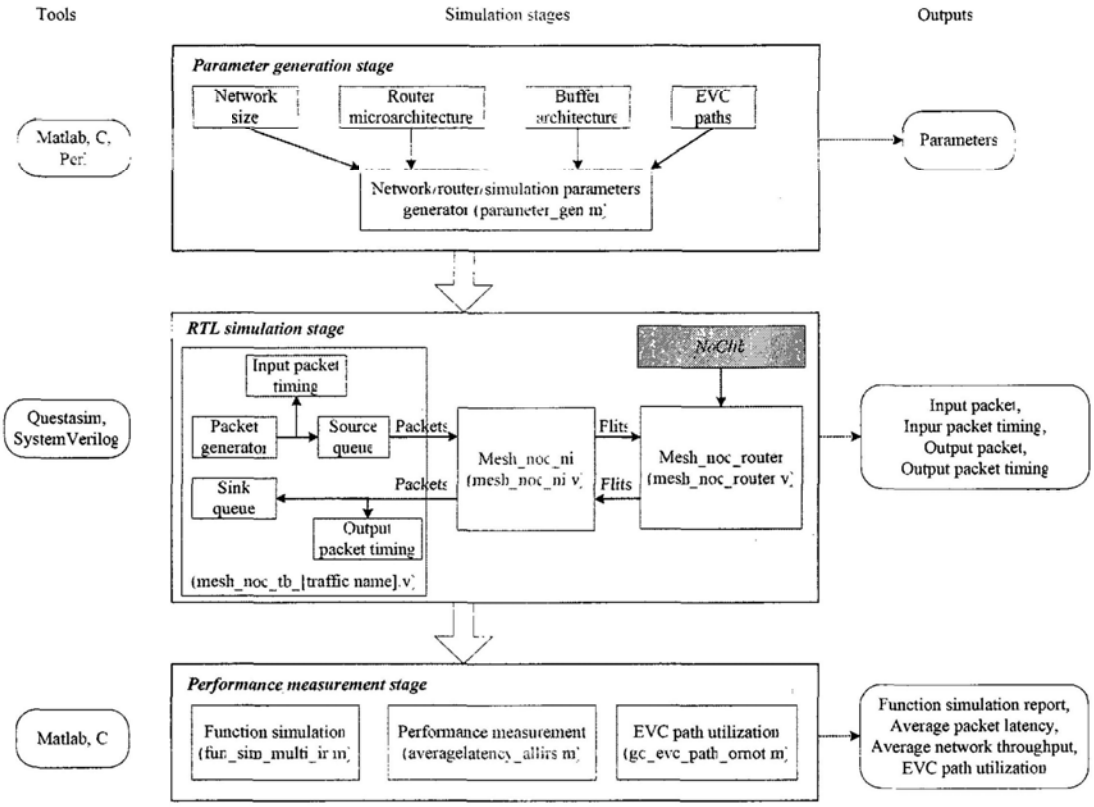


Figure C.1. Simulation framework using *NoClib*

Figure C.1 describes the framework to run NoCs simulations using *NoClib*. In the parameter generation stage, parameters are generated though a Matlab program (parameter_gen.m) based on network, router, buffer, and EVC configurations (EVC paths are only needed for EVC networks). The parameters are then used to configure the global parameters (parameters.v) shown in Table C.1 and the top-level blocks such as mesh_noc_router, mesh_noc_ni and testbench (mesh_noc_tb_[traffic name].v). The mesh_noc_router block includes all routers and links between routers in a mesh network while the mesh_noc_ni block includes all interfaces to inject/eject packets to/from routers. All other blocks such as packet generators, source queues, and so on are implemented in the top-level testbench block. After that, RTL simulations are run through tools like Questasim, which saves input/output packets and input/output packet timings to files. Finally, these files are used in the

performance measurement stage. The function simulation block checks whether all measured packets arrive at their sinks without any errors. The performance measurement block calculates average packet latency and average network throughput. The EVC path utilization block checks whether EVC packets go through EVC paths correctly and calculate the number of EVC packets travelling each EVC path. It is only used for EVC networks. In addition, we write a Perl program to automatically run the three simulation stages.

Since *NoClib* is developed using SystemVerilog, simulation tools must support syntax of SystemVerilog. In addition, many multi-dimension arrays are used in the top-level blocks (mesh_noc_router.v and mesh_noc_ni.v) to realize parameterized designs. Thus, simulation tools have to support multi-dimension arrays as well. To our knowledge, both Questasim by Mentor Graphics and VCS by Synopsys support most syntax of SystemVerilog. But VCS do not support multi-dimension arrays now (we tested the version of VCS.A-2008.09). Thus, we suggest to use Questasim for RTL simulations (we run RTL simulations using Questasim 6.3f and find no errors.). In addition, there is a way to run simulations for *NoClib*-based designs using tools that support only Verilog HDL. Firstly, synthesize designs through synthesis tools like Synopsys DC (we tested the version of DC.A-2007.12. It works.) and generate the gate-level netlist that is in the format of Verilog HDL. Then, run simulations using the gate-level netlist.

As mentioned before, *NoClib* is a fully-synthesizable library and thus can be used to implement NoCs on FPGA/ASIC for accurate area and power evaluations. The framework to implement NoCs using *NoClib* is the same as the conventional FPGA/ASIC implementation framework.

C.4 FUTURE WORK

NoClib can be enhanced in several directions to support more network/router architectures and wider range of parameters.

- Support multi network topologies. Various topologies require different sizes (numbers of input/output ports) of routers and different routing algorithms. If *NoClib* is used for simulations only, it will be easy to provide different sizes of routers through setting router radixes as parameters and to design a deadlock-free routing algorithm that supports multi topologies. However, if *NoClib* is also used for implementations, it will be difficult because costs of many router components will increase largely as router size increases.

- Enlarge the maximum allowable number of VC lanes in an input port. Currently, at most seven VC lanes (including both NVC and EVC lanes) are allowed in each input port. We can easily increase the maximum number by changing the two parameters (MAX_TOTAL_VCS_PERPC and MAX_TOTAL_VCS_PERPC_INDEX_BITS) if *NoClib* is used for simulations only. However, allowing a large number of VC lanes in input ports will generate very large control logic such as arbiters (for example, a *8:1* arbiter is much larger than a *4:1* arbiter). Thus, tree architecture based control logic is required for cost-efficient implementations.

- Support reconfigurable EVC architectures and buffer architectures. In current, EVC architectures and buffer architectures are both physically customized. This is to say, once a NoCs is customized for an application, EVC paths and buffer architectures for the NoCs can not be changed. As a result, the NoCs may not work well for other applications. However, it is

probable for a NoCs to support multi applications. Thus, it is important to design cost-efficient components to support reconfigurable EVC architectures and buffer architectures.

REFERENCES

[1] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Design, Automation and Test in Europe*, 2000, pp. 250-256.

[2] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Design Automation Conference*, 2001, pp. 684-689.

[3] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *Computer*, Vol. 35, No. 1, 2002, pp. 70-78.

[4] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "SPIN: A scalable, packet switched, on-chip micro-network," in *Design, Automation and Test in Europe*, 2003, pp. 70-73.

[5] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine*, Vol. 4, No. 2, 2004, pp. 18-31.

[6] "ITRS, International Technology Roadmap for Semiconductors," 2002.

[7] "ITRS, International Technology Roadmap for Semiconductors," 2007.

[8] Arteris Company, "A comparsion of network-on-chip and busses," 2005.

[9] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip," *Integration, the VLSI Journal*, Vol. 38, No. 1, 2004, pp. 19-42.

[10] J. D. Owens et al., "Research challenges for on-chip interconnection networks," *IEEE Micro*, Vol. 27, No. 5, 2007, pp. 96-108.

[11] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Computing Surveys*, Vol. 38, No. 1, 2006.

[12] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann Publishers, 2004.

[13] K. M. Lee et al., "A 51mW 1.6GHz on-chip network for low-power heterogeneous SoC platform," in *International Solid State Circuits Conference*, 2004, pp. 152-518.

[14] D. Bertozzi et al., "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 2, 2005, pp. 113-129.

[15] J. Xu and W. Wolf, "A design methodology for application-specific networks-on-chip," *ACM Transactions on Embedded Computing Systems*, Vol. 5, No. 2, 2006, pp. 263-280.

[16] V. D. Ngo, H. W. Choi, Y. Bae, and H. Cho, "The optimized tree-based network on chip topologies for H.264 decoder design," in *International Conference on Computer Engineering and Systems*, 2006, pp. 343-347.

[17] U. Y. Ogras and R. Marculescu, "Application-specific network-on-chip architecture customization via long-range link insertion," in *International Conference on Computer Aided Design*, 2005, pp. 246-253.

[18] U. Y. Ogras and R. Marculescu, "It's a small world after all: NoC performance optimization via long-range link insertion," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol. 14, 2006, pp. 693-706.

[19] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann Publishers, 2003.

[20] W. J. Dally, "Virtual-channel flow control," in *International Symposium on Computer Architecture*, 1990, pp. 60-68.

[21] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *International Symposium on Computer Architecture*, 2004, pp. 188-197.

[22] N. Kavaldjiev, G. J. M. Smit, and P. G. Jasen, "A virtual channel router for on-chip networks," in *IEEE International SoC Conference*, 2004, pp. 289-293.

[23] J. M. Kim et al., "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," in *International Symposium on Computer Architecture*, 2006, pp. 4-15.

[24] C. A. Nicopoulos, D. Park, and J. Kim, "VichaR: A dynamic virtual channel regulator for networks-on-chip routers," in *International Symposium on Microarchitecture*, 2006, pp. 333-346.

[25] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, "Adding slow-silent virtual channels for low-power on-chip networks," in *International Symposium on Networks-on-Chip*, 2008, pp. 23-32.

[26] M. Galles, "Spider: A high-speed network interconnect," *IEEE Micro*, Vol. 17, No. 1, 1997, pp. 34-39.

[27] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," in *Design Automation Conference*, 2002, pp. 524-529.

[28] X. Chen and L. S. Peh, "Leakage power modeling and optimization in interconnection networks," n *International Symposium on Low Power Electronics and Design*, 2003, pp. 90-95.

[29] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, 2006, pp. 2919-2933.

[30] S. R. Vangal et al., "An 80-tile sub-100-W teraFLOPS processor in 65-nm CMOS," *Journal of Solid-State Circuits*, Vol. 43, No. 1, 2008, pp. 29-41.

[31] E. Salminen, A. Kulmala, and T. D. Hamalainen, "Survey of network-on-chip proposals," in *OCP-IP White Paper*, 2008.

[32] M. B. Taylor et al., "The RAW microprocessor: A computational fabric for software circuits and general-purpose programme," *IEEE Micro*, Vol. 22, No. 2, 2002, pp. 25-35.

[33] A. Banerjee, R. Mullins, and S. Moore, "A power and energy exploration of network-on-chip architectures," in *International Symposium on Networks-on-Chip*, 2007, pp. 163-172.

[34] A. Banerjee et al., "An energy and performance exploration of network-on-chip architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 17, No. 3, 2009, pp. 319-329.

[35] A. Pullini et al., "NoC design and implementation in 65nm technology," in *International Symposium on Networks-on-Chip*, 2007, pp. 273-282.

[36] A. Kumar, L. S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," in *International Symposium on Computer Architecture (and IEEE Micro Top Picks 2008)*, 2007, pp. 150-161.

[37] H. S. Wang, L. S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *International Symposium on Microarchitecture*, 2003, pp. 105-116.

[38] T. H. Huang, U. Y. Ogras, and R. Marculescu, "Virtual channels planning for networks-on-chip," in *the 8th International Symposium on Quality Electronic Design*, 2007, pp. 879-884.

[39] L. S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *International Symposium on High-Performance Computer Architecture*, 2001, pp. 255-266.

[40] "Synopsys Company, http://www.synopsys.com/home.aspx," 2009.

[41] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, Vol. 7, No. 2, 1999, pp. 188-201.

[42] Y. Tamir and H. C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 1, 1993, pp. 13-27.

[43] S. S. Mukherjee et al., "A comparative study of arbitration algorithms for the Alpha 21364 pipelined router," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002, pp. 223-234.

[44] H. S. Wang. Power-efficient design for on-chip interconnection networks. PhD thesis, Princeton University, 2005.

[45] H. S. Wang, L. S. Peh, and S. Malik, "A technology-aware and energy-oriented topology exploration for on-chip networks," in *Design, Automation and Test in Europe*, 2005, pp. 1238-1243.

[46] U. Y. Ogras, R. Marculescu, H. G. Lee, and N. Chang, "Communication architecture optimization: Making the shortest path shorter in regular networks-on-chip," in *Design, Automation and Test in Europe*, 2006, pp. 712-717.

[47] W. J. Dally, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Transaction on Computers*, Vol. 40, No. 9, 1991, pp. 1016-1023.

[48] U. Y. Ogras et al, "Challenges and promising results in NoC prototyping using FPGAs," *IEEE Micro*, Vol. 27, No. 5, 2007, pp. 86-95.

[49] H. S. Wang, X. Zhu, L. S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *International Symposium on Microarchitecture*, 2002, pp. 294-305.

[50] M. Palesi et al., "Design of bandwidth aware and congestion avoiding efficient routing algorithms for networks-on-chip platforms," in *International Symposium on Networks-on-Chip*, 2008, pp. 97-106.

[51] A. J. Kleinosowski and D. J. Lilja, "MinneSPEC: A new SPEC benchmark workload for simulation-based computer architecture research," *Computer Architecture Letters*, Vol. 1, 2002.

[52] P. Gratz et al., "Implementation and evaluation of on-chip network architectures," in *International Conference on Computer Design*, 2006, pp. 477-484.

[53] P. Gratz et al., "On-chip interconnection networks of the TRIPS chip," *IEEE Micro*, Vol. 27, 2007, pp. 41-50.

[54] K. M. Lee, S. J. Lee, and H. J. Yoo, "Low-power network-on-chip for high-performance SoC design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 14, No. 2, 2006, pp. 148-160.

[55] M. Donno, A. Lvaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *Design Automation Conference*, 2003, pp. 622-627.

[56] P. Babighian, L. Benini, and E. Macii, "A Scalable algorithm for RTL insertion of gated clocks based on ODCs computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 1, 2005, pp. 29-42.

[57] C. Piguet, *Low-power Electronics Design*, CRC Press, 2005.

[58] R. Mullins, "Minimising dynamic power consumption in on-chip networks," in *International Symposium on System-on-Chip*, 2006, pp. 1-4.

[59] M. Mueller et al., "The impact of clock gating schemes on the power dissipation of synthesizable register files," in *International Symposium on Circuits and Systems*, 2004, pp. 609-612.