The Dissertation Committee for Woo Young Jang
Certifies that this is the approved version of the following dissertation:

Architecture and Physical Design for Advanced Networks-on-Chip

Committee:

David Z. Pan, Supervisor

Jacob A. Abraham

Adnan Aziz

Andreas Gerstlauer

Yin Zhang

Architecture and Physical Design for Advanced Networks-on-Chip

by

Woo Young Jang, B.E.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

The University of Texas at Austin

May 2011

Dedicated to my wife, Minkyung for her support, encouragement, and devotion.

Acknowledgements

First of all, I would like to express my deepest gratitude and appreciation to my advisor Prof. David Z. Pan for his invaluable advice, great generosity, and continuous encouragement throughout my research. I am extremely fortunate to study under his supervision. He has guided my research with remarkable insight and profound knowledge. Without his guidance and support, this dissertation would not have been completed.

I am also grateful to the members of my dissertation committee, Prof. Jacob A. Abraham, Prof. Adnan Aziz, Prof. Andreas Gerstlauer, and Prof. Yin Zhang. Their helpful advices on my research and their knowledge have guided me in my quest to do quality work.

I am indebted to many people in Samsung for their support and concern for me. In particular, I would like to thank Dr. Sun-Jae Cho, Mr. Byung-Hoan Chon, Dr. Seh-Woong Jeong, Dr. Jinhyun Kim, Dr. Donghoon Lee, and Mr. Youngbum Lee. Without their support and help, I could not have the great chance to study for a Ph.D. degree with the complete support of Samsung.

I would like to thank the colleagues at UTDA for their help and cheering up: James Ban, Ashutosh Chakraborty, Minsik Cho, Duo Ding, Jerrica Gao, Ou He, Kiwoon Kim, Anurag Kumar, Yen-Hung Lin, Katrina Lu, Joydeep Mitra, Jiwoo Pak, Sean Shi, Jae-Seok Yang, Bei Yu, Kun Yuan, Rein Zhang, and Yilin Zhang.

I sincerely appreciate the support and love from my father and mother who always encourage me to pursue my goal. I also thank my parents-in-law who always cheer me up and my daughter, Hannah who always makes me smile. Finally, I would like to express my sincerest and deepest gratitude and love to my wife, Minkyung, who

always supports and encourages me with great love. I will pay my debt back to her throughout my life.

Woo Young Jang

The University of Texas at Austin

May 2011

Architecture and Physical Design for Advanced Networks-on-Chip

Publication No._____

Woo Young Jang, Ph.D.

The University of Texas at Austin, 2011

Supervisor: David Z. Pan

The aggressive scaling of the semiconductor technology following the Moore's Law has delivered true system-on-chip (SoC) integration. Network-on-chip (NoC) has been recently introduced as an effective solution for scalable on-chip communication since dedicated point-to-point (P2P) interconnection and shared bus architecture become performance and power bottlenecks in the SoCs. This dissertation studies three critical NoC challenges such as latency, power, and compatibility with emerging technologies in aspect of an architecture and physical design level.

Latency is a key issue in NoC since the performance of applications considerably depends on resource sharing policies employed in an on-chip network. NoCs have been mainly developed to improve network-level performance that captures the inherent performance characteristics of a network itself, but the network-level optimizations are not directly related to application- or system-level performance. In addition, memory latency on NoC critically affects the performance of applications or systems. We propose a synchronous dynamic random access memory (SDRAM) aware NoC design to

optimize memory throughput, latency, and design complexity. Furthermore, it is extended to an application-aware NoC design to provide the quality-of-service (QoS) of memory for various applications.

NoC provides great on-chip communication. However, it brings no true relief to power budget when the on-chip network scales in terms of complexity/size and signal bandwidth. The combination of NoC and other techniques has the potential to reduce power. We study two power saving research topics for NoC: (a) we propose a voltage-frequency island (VFI) aware NoC optimization framework with a better tradeoff between power efficiency and design complexity to minimize both computation and on-chip communication power. (b) We formulate an application mapping problem to mixed integer quadratic programming (MIQP) with the purpose of reducing power consumption in various hard networks and develop highly efficient algorithms for the MIQP.

Regarding NoC compatible with new technologies, we focus on three dimensional (3D) die integration based on through-silicon vias (TSVs). Since an on-chip network design has been subject to not only application constraints but also design/manufacturing constraints, a 3D NoC design is required for innovation in interconnection networks. We propose a chemical-mechanical polishing (CMP) aware application-specific 3D NoC design that minimizes TSV height variation, thus reduces bonding failure, and meanwhile optimizes conventional NoC design objectives such as hop count, wirelength, power, and area.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Network-on-Chip Challenges in Ultra-Deep Submicron Era

The aggressive scaling of the semiconductor technology has enabled billions of transistors to be integrated to a single chip, following Moore's Law that the minimum feature size is scaled down at the rate of a factor 0.7 reduction every three years. The technology scaling trend has continued for more than half a century and it is expected to last until 2015 or later, according to the International Technology Roadmap for Semiconductors [48]. The effective reduction in size and cost provides higher chip performance in a smaller silicon area and thus enables the realization of scenarios deemed to belong to the domain of science fictions.

The continued feature size scaling has delivered the potential of true and complete system-on-chip (SoC) integration. However, as most SoC designs target the high-performance system level integration of existing heterogeneous cores with low power consumption, previous dedicated point-to-point (P2P) interconnections and shared bus architectures become performance bottlenecks due to the increasing communication between the cores. Furthermore, with the rapid technology scaling, the global interconnection causes critical delays and high energy consumption. To mitigate such issues, network-on-chip (NoC) has been recently introduced as an effective solution for the scalable on-chip communication [4][18]. As the better SoC platform for system integration, NoC makes interconnect structure and wiring complexity manageable easily such that the issues in a physical design such as floorplanning, placement, and routing can be well optimized. It leads to faster time-to-market by reduction in the number of

design re-spins. Therefore, the NoC has attracted great attentions for the current and future SoC designs.

Figure 1.1 illustrates general NoC architecture. Each processing element (PE) is attached to its own router via a network interface logic or a wrapper and the router is interconnected to different routers. When any PE receivers or transfers data to different PE, the requests and data are encoded or wrapped to a packet in the network interface logic or the wrapper and then the packet is delivered to its own router. The packet is stored at an input buffer and a routing logic in the router selects the path of the packet on a given network topology. If more than two packets arriving on different input buffers at the same time desire the same output channel, a flow-control mechanism resolves this contention. An output scheduler either detects if the input buffer of the next router is available or expects when the input buffer is available. After performing such operations, the packets are delivered to the next router on its path. This process is repeated until the packet arrives at its final destination.

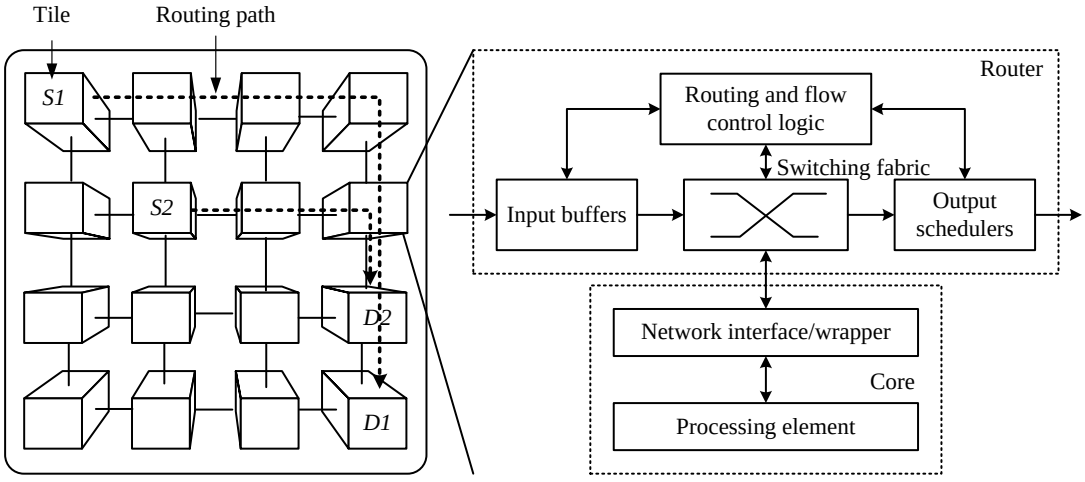


Figure 1.1: NoC architecture.

For the last decade, there have been many NoC researches to achieve greater design productivity and higher performance by handling increasing parallelism, manufacturing complexity, wiring problems, and reliability, where critical challenges for NoC include latency, power, and compatibility with new technologies [88]. Unfortunately, a number of researches gave an impression that NoC greatly improved SoC designs where it was utilized, but failed to show that NoC reduces latency and power consumption, compared to shared bus interconnects. In addition, as emerging technologies have become feasible, new constraints and design flows are required for innovation in NoC.

Latency is crucial to the success of NoC since an on-chip network with long latency can considerably deteriorate the overall application performance although its high throughput. NoCs have been mainly developed to improve network-level performance such as throughput or average network latency [5][21][31][59][63][76][83][89][94]. It captures the inherent performance characteristics of a network itself, but is not directly related to application-level or system-level performance. This is because each application demands different network performance and much of the system performance depends on not only on-chip networks but also shared memories, in particular, synchronous dynamic random access memories (SDRAMs). The application- and SDRAM-oblivious NoCs lead to reduced overall system performance. Therefore, latency in NoC is required to approach the characteristics of shared bus interconnects with the consideration of various applications and memories.

Power should be budgeted and traded off among different NoC optimization factors since it has also become a major issue. NoC itself is not efficient for power and even may consume higher power than shard bus interconnects due to additional power consumers such as router and network interface logics. However, the combination of

NoC and other techniques efficient for power has the potential to easily reduce power to allowable levels. A voltage-frequency island (VFI) paradigm is one of the desirable solutions for reducing power consumption in NoC since it is inefficient for all cores and links on NoC to operate at a single voltage level and clock speed [10][41][62][66][84] [118][119]. VFI enables fine-grained core-level power optimization by utilizing a unique voltage and clock for each island. The use of multiple voltages and clocks in NoC provides better performance-power tradeoffs than that of a single voltage and clock. In addition, application mapping which decides how to topologically place the selected set of cores onto the tiles of a network can greatly reduce both application latency and power consumption. NoC designers or programmers favor regular mesh architecture consisting of regular rectangle tiles on which homogeneous processors are placed since the regular mesh architecture makes the application mapping manageable [13][15][39][79][103]. On the contrary, most industrial SoC platforms consist of heterogeneous cores with different design areas, and thus they can be structured with an irregular mesh network or even a custom network. Therefore, since previous works have mainly optimized their application mapping on the regular mesh architecture, the application mapping algorithm is required to reduce application latency and power consumption in various networks.

The architecture and physical design for an on-chip network design should be compatible with emerging technologies since it has been always subject to technology constraints. With shrinking transistor and wire dimensions, variability and reliability have become important for NoC designs. In addition, as three dimensional (3D) die integration using through-silicon vias (TSVs) becomes viable, 3D NoC becomes new opportunities and challenges [80][102][120]. Since 3D NoC must satisfy not only application constraints such as latency, throughput, and power, but also manufacturing/design constraints imposed by 3D technologies such as the number of TSVs, chemical-

mechanical polishing (CMP), TSV stress, and temperature, 3D NoC design shall consider such constraints for interconnection networks.

Therefore, it is indispensable to propose novel architecture and physical designs for advanced NoCs in ultra-deep submicron era, which can address all these challenges in an effective and efficient manner. First, we need to improve system-level or application-level performance with consideration of various application demands and memories. Next, we propose a VFI based NoC design and an application mapping algorithm to reduce power consumption. Finally, we propose a 3D NoC design with consideration of both application constraints and manufacture/design constraints imposed by the 3D technology.

1.2 OVERVIEW AND CONTRIBUTIONS OF THIS DISSERTATION

The architectures and physical design techniques for advanced NoC, presented in this dissertation target the above mentioned challenges and are described in the next three chapters. The overall flow of the dissertation is as follows.

Chapter 2 presents SDRAM- and application-aware NoC designs to improve not only network-level performance but also application-level or system-level performance. The performance of various applications considerably depends on the resource sharing policies employed in an on-chip network. In particular, memory service for the applications becomes one of the most important issues since its performance becomes the bottleneck of the overall system. Unfortunately, its improvement aided by a memory subsystem is severely limited since diverse applications generate their specific memory requests with different latency constraints and data sizes. With consideration of different

demands of applications, our on-chip network shares the responsibility for the memory performance with the memory subsystem.

In Chapter 3, we propose a VFI based design flow and application mapping algorithms for a low power NoC design. The NoC design style fits nicely with the concept of VFI. There have been several design efforts to combine VFI based design style with the NoC interconnect mechanism. However, previous works are limited since VFI-awareness is partially applied in a NoC design. In Section 3.1, a systematic VFI-aware energy optimization framework that considers partitioning, mapping, and routing together is presented to improve the power efficiency of VFI-based NoC designs. In Section 3.2, we propose architecture-aware analytic application mapping (A3MAP) algorithms that are analogous to analytical communication minimization in a given NoC. The proposed A3MAP algorithms adaptively map cores to any different sized tiles on regular/irregular meshes and custom networks for the minimum power consumption under performance constraints.

In Chapter 4, we propose a CMP process-aware application-specific 3D NoC design that minimizes TSV height variation, thus reduces bonding failure, and meanwhile optimizes conventional NoC design objectives, such as hop count, wirelength, power, and area. Previous NoC design flows are not effective in 3D integration since they do not consider manufacturing/design constraints by TSVs. The key idea behind our 3D NoC design flow is to determine the CMP-aware network topology where different layers are interconnected by one-way links with the minimum hops and thus TSV height variation is minimized. This is the first work that addresses the 3D NoC design which considers architecture, physical design and manufacturing issues together.

Chapter 5 concludes this dissertation with summaries based on the results of the previous chapters as well as presents promising future research directions to further investigate architecture and physical design for advanced NoC.

Chapter 2

Memory-Aware NoC Design for Improving Application-Level Latency

Memory bandwidth and latency to feed a number of cores have become a key issue in the modern and future SoC design. SDRAM is commonly used as a shared memory since it provides high memory capacity and infinite endurance for modern computing systems. However, since the SoCs mainly interface with a single or dual SDRAM, there would be insufficient memory bandwidth to keep up with a number of high speed cores. For example, Intel Teraflop which is the state-of-the-art NoC and composed of 80 cores is supported by a dual shared memory [112]. If cores will have access to the single or dual memory at the same time, memory latency will be too long to provide real-time computing. As an effective solution of memory bandwidth and latency, 3D NoC based on TSV technology [78] is gaining momentum and industry adoption. 3D NoC can be embedded with a lot of SDRAMs on top of processing elements at different layers [67]. It achieves higher system performance and more reliable electrical features. Furthermore, it provides low power consumption, low electromagnetic interference (EMI), small die and printed circuit board (PCB) area and low pin density.

Most NoCs with a number of cores require a dedicated memory subsystem to control SDRAMs. The memory subsystem that schedules SDRAM requests and generates SDRAM interface signals is one of the most important components in SoCs since the performance of the entire system depends on its performance. However, the conventional memory subsystem still underperforms due to special operation flows of SDRAM [24]. For example, double data rate (DDR) II SDRAM utilization gets deteriorated up to 55% in a digital television (DTV) application [113], where memory utilization is defined as the number of clock cycles used for data transfer divided by the

number of total clock cycles. In addition, since on-chip networks are oblivious of applications and SDRAMs, their performances are not directly related to the performance observable at the application level or system level. Moreover, since the corresponding number of a memory subsystem must also be equipped to control a number of SDRAMs, the cost of an NoC design will rapidly increase. Therefore, considerable attention has been shifted toward memory-aware NoC exploration to improve memory utilization and latency with the economical design cost of NoC platform [27].

2.1 SDRAM-AWARE NOC DESIGN

A memory subsystem usually consists of three parts, i.e., a buffer, a SDRAM scheduler and a SDRAM interface signal generator (or memory controller), where a depth of buffer and an SDRAM scheduler for reordering dynamic SDRAM requests are key components for higher memory utilization and shorter memory latency. Panda et al. presented synthesis models for various off-chip memory access modes, as well as a technique for analyzing a behavior to determine memory accesses that can be optimized by exploiting the available memory features [90]. A memory scheduler proposed in [96] supports preemption and reordering to optimize offered net bandwidth and average latency. Schedulers discussed in [36] and [114] support preemption for high-priority requests to decouple latency and rate. In [1], PREDATOR is proposed with two step approaches: grouping memory requests and predictable arbitration for the group. A memory scheduler proposed in [44] adopts an adaptive history-based (AHB) scheduler that uses a history of recently scheduled operations to improve memory efficiency. However, the improvement just aided by such memory subsystems is severely limited

since diverse applications generate their specific memory requests with different latency constraints and the different data sizes.

Recently, microprocessors and shared buses considering SDRAM operations have been developed to support a guaranteed memory service. In [59], a memory bus was implemented to source-synchronous code division multiple access. A low-cost memory controller was present in [64] to maximize the benefit of useful prefetches and to minimize harms caused by useless prefetches. Cost-effective on-chip memory request issue mechanisms were proposed in [65] using SDRAM bank-level parallelism (BLP)-aware prefetch issue and BLP-preserving multi-core request issue. In [20], network interface architecture was proposed to cope with in-order delivery, resource utilization, and latency. A memory controller was integrated into this network interface to improve memory utilization and reduced both memory latency and network latency. However, they all do not provide an efficient priority memory service or an access granularity matching solution when using multiple SDRAMs.

Flow control in NoC is on how network resources, e.g. channel bandwidth, buffer capacity, and control state, are allocated to packets traversing a network. In previous works, congestion control is well studied for macro-networks. For example, decentralized control and predictive explicit-rate control are developed in [89], where sources adjust their traffic generation rates based on feedbacks received from bottleneck links. In [94], a predictive flow controller managing a packet injection rate to regulate the number of packet is proposed, based on traffic sources and router models. To minimize overall execution time and link utilization of applications, optimal link scheduling and shared buffer router architecture are proposed in [83]. An open-loop flow control scheme is proposed in [60] to reduce conflicts of data transfers from multiple memory modules to

the same masters. In addition to such congestion control mechanisms, flow controllers may be useful for scheduling packets for memories.

This section presents an SDRAM-aware NoC design to improve memory utilization and latency with a low design cost [50][54]. Our key ideas are twofold. First, if each NoC router schedules memory request packets, the packets arrive at a memory subsystem in the order that is friendly to SDRAM operations. Since our SDRAM-aware router uses existing resources to schedule the packets, e.g. input buffers for storing blocked packets and other flow-control mechanisms, additional circuitry is tiny. On the other hand, a heavy reordering buffer and a complex scheduler can be removed in a memory subsystem. Second, a scheduling scheme performed by multiple SDRAM-aware routers outperforms a scheduling scheme performed by a single memory subsystem. The reason is that the performance of single-stage scheduling mainly depends on the number of port/buffer in the single memory subsystem. However, the multi-stage scheduling uses all the buffers in multiple routers to schedule the memory request packets. Based on these ideas, the major novelty and contribution of this section include:

- We propose a novel NoC router architecture with explicit SDRAM-aware flow control to schedule SDRAM access requests instead of using the conventional memory subsystem.
- We propose SDRAM-aware flow control algorithms to resolve problems of bank conflict, data contention and short turn-around bank interleaving, which employs priority-based arbitration and multi-stage scheduling.
- We show that an NoC design embedding our SDRAM-aware router achieves higher memory utilization, shorter memory latency and cheaper design cost than the conventional NoC design with an SDRAM-unaware router.

- We show that performance of our SDRAM-aware router gets better for complex NoC architectures and high- performance SDRAM.

To the best of our knowledge, this is the first work that addresses a router scheduling memory requests instead of a memory subsystem. The rest of this section is organized as follows. In the next section, we survey related works. In Section 2.1.2, we review basic SDRAM operation principles and SDRAM request scheduling. In Section 2.1.3, the problem of the conventional SDRAM-unaware NoC router is presented and our basic solution is proposed. Section 2.1.4 presents detailed description of our SDRAM-aware router. Experimental results are shown in Section 2.1.5. Finally, Section 2.1.6 summarizes Section 2.1.

2.1.1 Basic SDRAM Operation

SDRAM has a three dimensional structure, i.e., a bank, a row, and a column as shown in Figure 2.1. Basic commands to access SDRAM are activation (ACT), read/write (R/W), and precharge (PRE), where the ACT command is executed with a bank address (BA) and a row address (RA), the R/W command is executed with BA and a column address (CA), and the PRE command is executed only with BA. A bank becomes active by an ACT command and idle by a PRE command. An R/W command can be executed only after a bank is activated. In Figure 2.1, when a bank is activated, one row data of the bank move to a row buffer of the bank. It takes t_{RCD} to complete an ACT command. Timing parameters of DDR I, II, and III SDRAM used in this work is shown in Table 2.1 [24]. As shown in Table 2.1, the faster clock rate is used in DDR SDRAM, the more clock cycles are required to complete SDRAM operations. For

example, DDR I SDRAM working at 133MHz clock frequency spends only two clock cycles activating a bank while DDR III SDRAM working at 800MHz clock frequency spends 11 clock cycles activating a bank. Then, an R/W command is executed on the active row buffer. After either read latency called column access strobe (CAS) latency (CL) or write latency (WL), successive data go from or to SDRAM. Finally, a PRE command is executed to deactivate the active row buffer in the bank, i.e., data in the row buffer move to the bank of the row buffer. It takes the bank state t_{RP} to become an idle state.

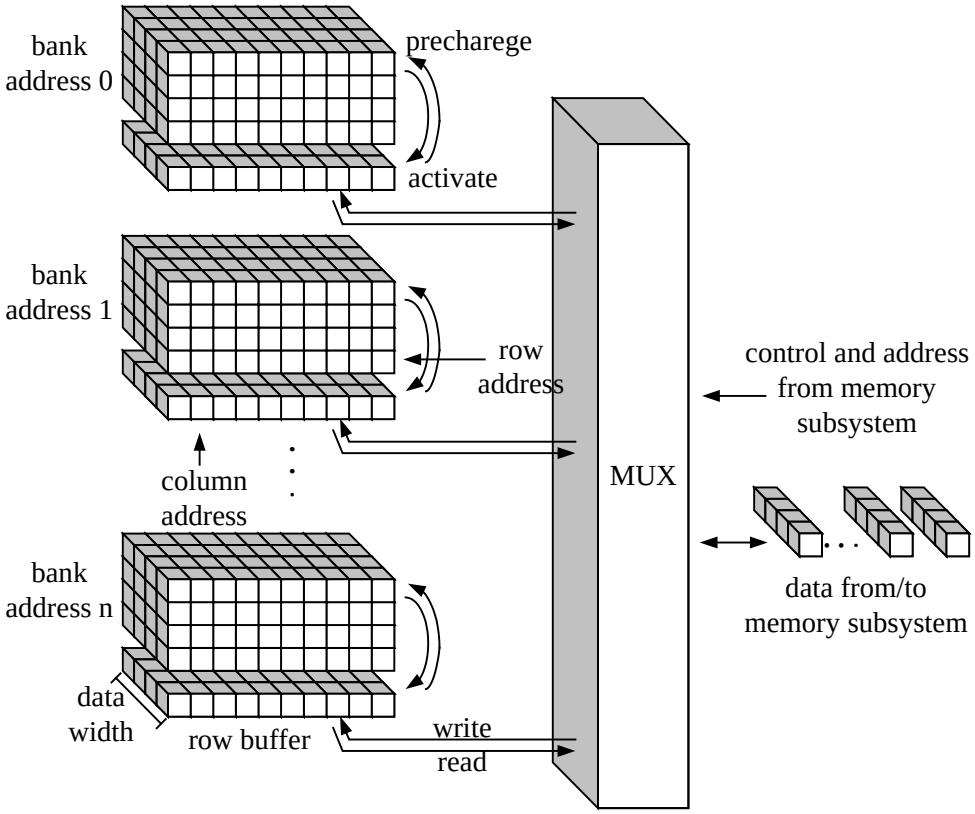


Figure 2.1: SDRAM architecture and activation, read/write, and deactivation operations.

2.1.2 SDRAM Scheduling

SDRAM consists of independent multiple banks whereas address and data pin/wire resources serialize accesses to different banks, as shown in Figure 2.1. The benefit of this architecture is that pin/wire resources between SDRAM and SoC can be saved and commands to different banks can be pipelined, i.e., while data are transferred to or from any bank, the rest of bank becomes idle and active for the latter request. Based on this principle, memory subsystems schedule SDRAM access requests. However, the improvement of memory performance is still limited due to special operation flows of SDRAMs and clock cycles wasted by timing constraints in Table 2.1. Moreover, it is much worse in high performance SDRAMs. Main factors which deteriorate memory performance are bank conflict, data contention, and short turn-around bank interleaving explained in the next three subsections.

2.1.2.1 Bank Conflict

Continuously accessing one bank with different RAs is called *bank conflict* which is the most critical to SDRAM performance. Since a bank activated by the former request should get idle and then active for the latter request again, a lot of clock cycles are required to complete these operations. For example, in Figure 2.2, there are two SDRAM schedulers reordering four read requests, i.e., read 1 (RA 0, BA 0, CA 0), read 2 (RA 1, BA 0, CA 0), read 3 (RA 0, BA 1, CA 0), and read 4 (RA 1, BA 1, CA 0). We assume that all schedulers work for DDR II SDRAM at 333MHz clock frequency. In Figure 2.2 (a), let them scheduled in the order, read 1, read 2, read 3, and read 4 by scheduler 1. After performing read 1, read 2 cannot be immediately executed since a row buffer of bank 0 is already occupied by data of RA 0. Hence, a PRE command should release the open row buffer of bank 0 and then an ACT command should be executed to fill the row

buffer of bank 0 with data of RA 1. On the contrary, read 3 can be pipelined, called *bank interleaving*, since it has different BA with read 2. As shown in Figure 2.2(a), while the bank 0 is activated and accessed for read 2, bank 1 gets activated for read 3. As a result, data 3 accessed by read 3 are generated with no loss of clock cycle. The last read 4 conflicts with read 3 since they have the same BAs, but different RAs.

On the contrary, scheduler 2 changes the execution order of four read requests, read 1, read 3, read 2, and read 4 as shown in Figure 2.2(b). Since this order does not cause any bank conflict, all read requests are pipelined. That means the second SDRAM scheduler lets all requests completed faster and latency of data 3 and 4 be shorter than the first SDRAM scheduler. In this example, the first scheduler achieves 9.5% (= 4 data/42 clock cycles) memory utilization and the second scheduler achieves 13.3% (= 4 data/30 clock cycles) memory utilization. Therefore, the second one is more desirable.

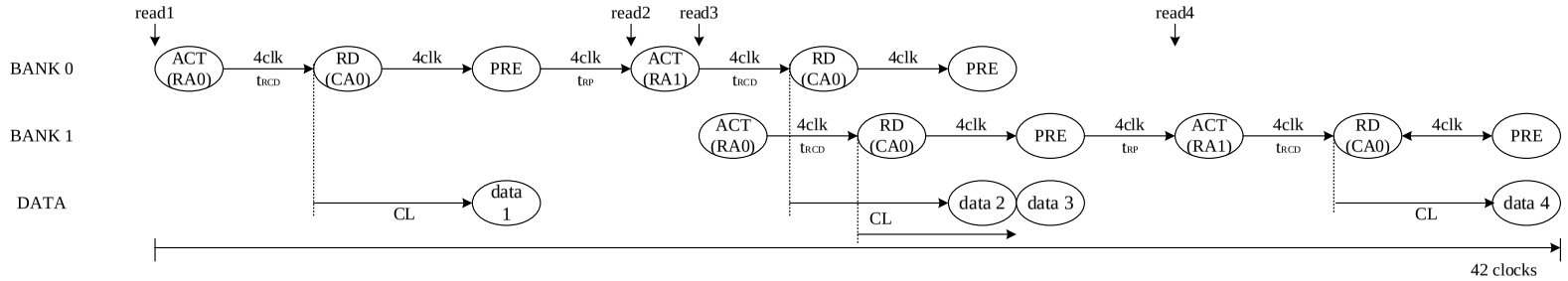
2.1.2.2 Data Contention

A case of a write request followed by a read request or a read request followed by a write request is called *data contention*. Data pins/wires are bidirectional in most SDRAMs while control and address pins/wires are unidirectional. As a result, input data may be collided with output data. To transfer data to SDRAM after receiving data from SDRAM, there should be at least one clock cycle interval between writing data and reading data in DDR I/II SDRAM. Since internal read-to-write command delay time (t_{RTW}) is required in DDR III SDRAM in Table 2.1, an interval between read data and write data happens up to two clock cycles. t_{RTW} is $CL+t_{CCD}+2-WL$ if burst length (BL) is 8 or t_{RTW} is $CL+t_{CCD}/2+2-WL$ if BL is 4. Hence, data contention is naturally hidden behind this delay time in DDR III SDRAM.

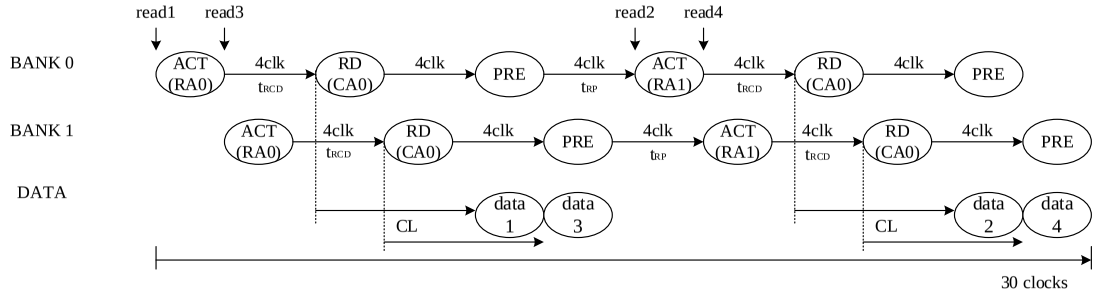
Timing parameter	DDR I SDRAM			DDR II SDRAM				DDR III SDRAM			
	133MHz	167MHz	200MHz	200MHz	267MHz	333MHz	400MHz	400MHz	533MHz	667MHz	800MHz
CL[b]	2	2.5	3	3	4	4	6	6	8	10	11
WL[c]	1	1	1	2	3	3	5	5	6	7	8
t_{RCD}[d]	2	3	3	3	4	4	6	6	8	10	11
t_{CCD}[e]	1	1	1	2	2	2	2	4	4	4	4
t_{RP}[f]	2	3	3	3	4	4	6	6	8	9	11
t_{WR}[g]	2	3	3	3	4	5	6	6	8	10	12
t_{WTR}[h]	1	1	2	2	2	3	3	4	4	5	6
t_{RTW}[i]	-	-	-	-	-	-	-	7	8	9	9

[a] We assume that posted CAS (Column Access Strobe) additive latency (AL) is 0 in DDR II/III SDRAM.
[b] CAS latency or read latency.
[c] Write latency.
[d] RAS (Row Access Strobe) to CAS delay time.
[e] CAS-to-CAS command delay
[f] Row precharge time.
[g] Write recovery time.
[h] Internal write-to-read command delay time.
[i] Internal read-to-write command delay time for DDR III SDRAM. If Burst Length (BL) is 8, it is $CL+t_{CCD}+2-WL$ and if BL is 4, it is 4 ($=CL+t_{CCD}/2+2-WL$). In this table, BL is 8.

Table 2.1: Timing parameter of DDR I, II[a], and III[a] SDRAM.



(a) Scheduler 1: read1 (RA0, BA0, CA0), read2 (RA1, BA0, CA0), read3 (RA0, BA1, CA0) and read4 (RA1, BA1, CA0)

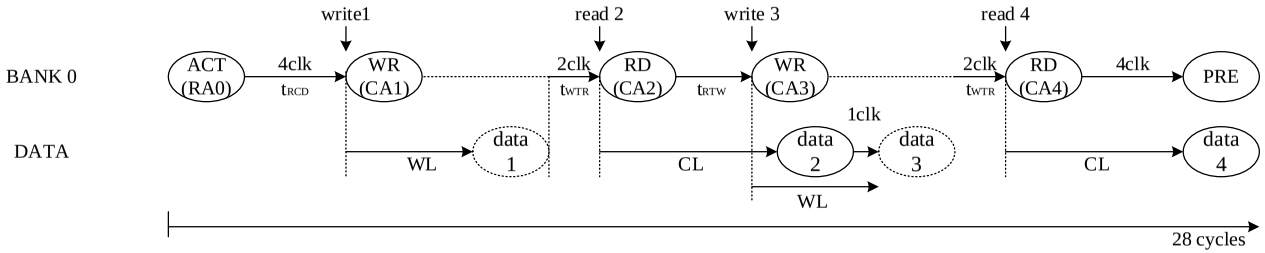


(b) Scheduler 2: read1 (RA0, BA0, CA0), read3 (RA0, BA1, CA0), read2 (RA1, BA0, CA0) and read4 (RA1, BA1, CA0)

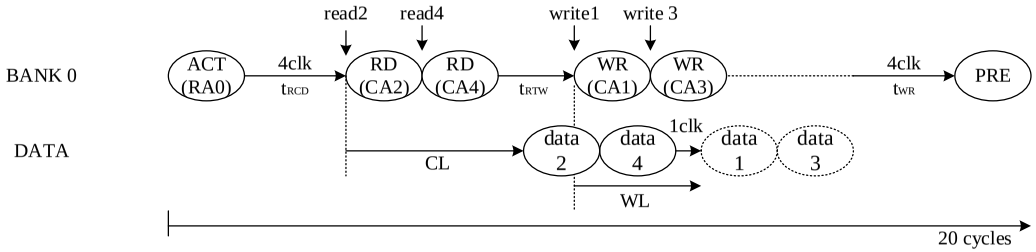
Figure 2.2: Examples showing bank conflict and interleaving in DDR II SDRAM @333MHz.

On the other hand, a read command following a write command needs internal write-to-read command delay time (t_{WTR}) to be executed. Then, after read latency or CL, reading data can be received from SDRAM. Write-to-read data contention is naturally hidden behind t_{WTR} and CL, but they cause memory utilization and memory latency degraded critically. Therefore, continuous read or write requests are preferred to access SDRAM efficiently.

For example, in Figure 2.3, there are two SDRAM schedulers reordering two write requests and two read requests, i.e., write 1 (RA 0, BA 0, CA 1), read 2 (RA 0, BA 0, CA 2), write 3 (RA 0, BA 0, CA 3), and read 4 (RA 0, BA 0, CA 4). All schedulers interface with DDR II SDRAM working at 266MHz clock frequency. As shown in Figure 2.3(a), let them scheduled in the order, write 1, read 2, write 3, and read 4 by scheduler 1. In this figure, read 2 cannot be immediately performed after writing all data 1 since t_{WTR} is required to accept the next read command. Furthermore, since data 2 are received from SDRAM after read latency or CL, a read request following a write request wastes total t_{WTR} and CL cycles even if bank conflict does not happen between two requests. If both bank conflict and data contention happen simultaneously, bank conflict is commonly prioritized. Since bank conflict wastes more clock cycles than data contention, data contention is hidden behind bank conflict. On the contrary, a write request following a read request has no internal command delays in DDR I/II SDRAM. Instead, a write command performing write 3 should be given to DDR SDRAM when it does not cause any collision with data 2. Most DDR I/II SDRAM schedulers get at least one clock cycle interval between read data and write data. If DDR III SDRAM is used, data contention is hidden naturally behind t_{RTW}. The last read 4 requires both t_{WTR} and CL before transferring data 4.



(a) Scheduler 1: write1 (RA0, BA0, CA1), read 2 (RA0, BA0, CA2), write 3 (RA0, BA0, CA3), read 4 (RA0, BA0, CA4)



(b) Scheduler 2: read 2 (RA0, BA0, CA2), read 4 (RA0, BA0, CA4), write 1 (RA0, BA0, CA1), write 3 (RA0, BA0, CA3)

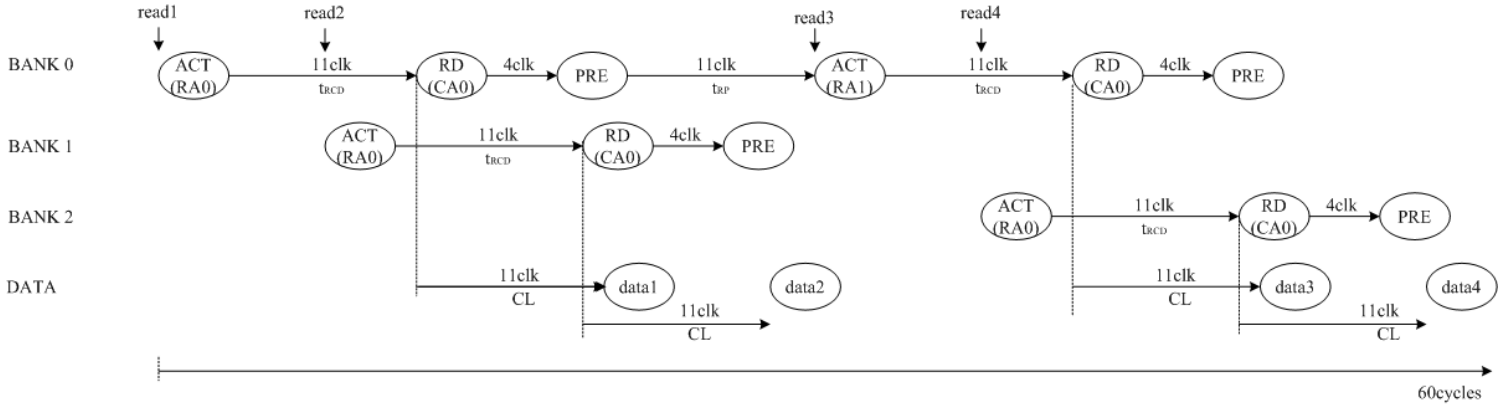
Figure 2.3: Examples showing data contention in DDR II SDRAM @266MHz.

On the contrary, scheduler 2 changes the order of two write requests and two read requests, i.e., read 2, read 4, write 1 and write 3 as shown in Figure 2.3(b). Since this order causes one data contention wasting just one clock cycle, all read/write requests are performed faster than scheduler 1. In common SDRAM operations, after writing data 3, write recovery time (t_{WR}) is required to accept a PRE command. Scheduler 1 and scheduler 2 take 28 and 20 clock cycles, respectively, until bank 0 becomes idle after performing all requests. As a result, scheduler 1 achieves 14.3% (= 4 data/28 clock cycles) memory utilization and scheduler 2 achieves 20% (= 4 data/20 clock cycles) memory utilization. Therefore, continuous read or write requests are encouraged to access SDRAM efficiently.

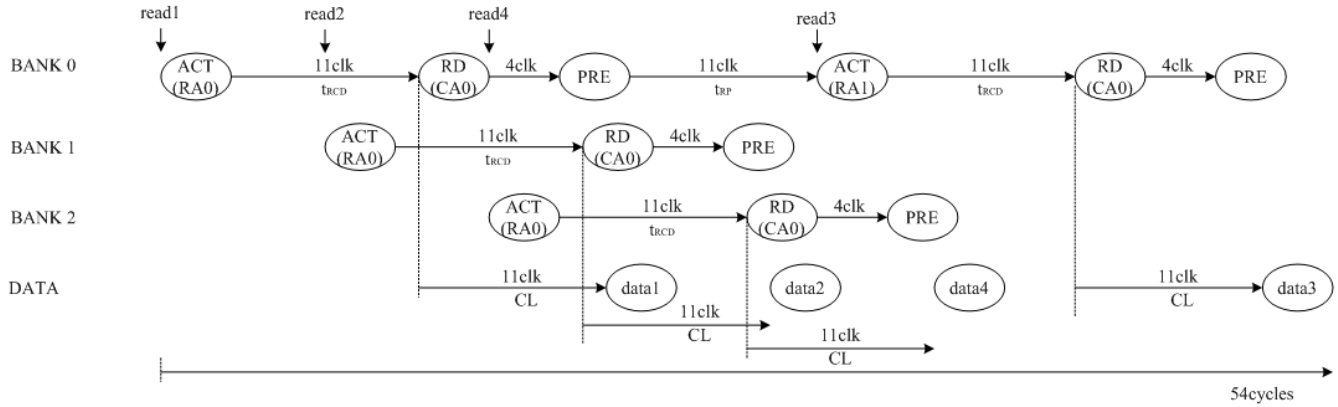
2.1.2.3 Short Turn-Around Bank Interleaving

A bank interleaving approach as a solution of bank conflict is the efficient technique. Hence, high memory utilization and short memory latency can be achieved as explained in Section 2.1.2.1. However, bank interleaving may achieve little improvement, in particular, in high performance SDRAM even if bank interleaving is performed completely. In Table 2.1, as an operating clock of SDRAM is faster and faster, activation delay time (t_{RCD}), deactivation delay time (t_{RP}) and read/write latency (CL/WL) are also longer and longer. The long delay times let the benefit of bank interleaving critically degraded since a bank interleaved may not get sufficient time to be deactivated or reactivated after the bank is accessed by the previous request with different RA.

For example, in Figure 2.4, there are two SDRAM schedulers reordering four read requests, i.e., read 1 (RA 0, BA 0, CA 0), read 2 (RA 0, BA 1, CA 0), read 3 (RA 1, BA 0, CA 0), and read 4 (RA 0, BA 2, CA 0). We assume that all schedulers work for DDR



(a) Scheduler 1: read1 (RA0, BA0, CA0), read2 (RA0, BA1, CA0), read3 (RA1, BA0, CA0) and read 4 (RA0, BA2, CA0)



(b) Scheduler 2: read1 (RA0, BA0, CA0), read2 (RA0, BA1, CA0), read 4 (RA0, BA2, CA0) and read3 (RA1, BA0, CA0)

Figure 2.4: Examples showing short turn-around bank interleaving in DDR III SDRAM @800MHz.

III SDRAM at 800MHz clock frequency. In Figure 2.4(a), let them be scheduled in the order, read 1, read 2, read 3, and read 4 by scheduler 1 such that all read requests are performed without bank conflict. After performing read 1, bank 0 is deactivated and read 2 starts to receive data 2. Then, read 3 waits until all data 2 are received. However, read 3 accessing bank 0 cannot be performed even if read 2 is done and the relation between read 2 and read 3 is bank interleaving. The reason is that bank 0 accessed by read 1 is not deactivated due to too long t_{RP}, i.e., operations for read 3 such as deactivation, reactivation, and read/write cannot be hidden behind the process of read 2. Hence, while bank 0 is deactivated, reactivated with data of RA 1, and ready to transfer data 3, any data cannot be transferred or received from other banks, which makes memory utilization and latency degraded.

On the contrary, scheduler 2 changes the execution order of read 3 and read 4 as shown in Figure 2.4(b). As a result, read 4 accessing bank 2 can be hidden behind the process of executing read 2 and even read 3 accessing bank 0 can be hidden behind the process of executing read 4. If there is another read 5 accessing bank 3 and it is performed between read 4 and read 3, data may be transferred more continuously with no loss of clock cycle. Consequently, memory utilizations by scheduler 1 and scheduler 2 are 6.7% (= 4 data/60 clock cycles) and 7.4% (= 4 data/54 clock cycles), respectively. Since this problem is more serious in high performance DDR SDRAM, a memory subsystem should check when banks get active again even if bank interleaving is performed completely.

2.1.3 NoC Design with SDRAM

2.1.3.1 Problem Description

Bank conflict and data contention frequently happen in the conventional NoC design due to limited resources such as an input buffer in a memory subsystem. Moreover, short turn-around bank interleaving also happens in high performance DDR SDRAM. Figure 2.5 shows a simple example of bank conflict in a 2×3 NoC design under the limited resources. This NoC includes a single memory subsystem that consists of an input buffer, a memory scheduler and an SDRAM interface signal generator. The memory scheduler reorders packets stored in the input buffer to avoid bank conflict, data contention and short turn-around bank interleaving. In this figure, *RxBy* means that a row address (RA) and a bank address (BA) of packet are *x* and *y*, respectively. An arrow indicates that a packet will move to the direction at the next clock cycle. We assume that a length of all packets is 1, the memory subsystem includes a two-depth input buffer to store two packets and the scheduler makes one of two stored packets executed every cycle (although execution time is actually longer than one cycle).

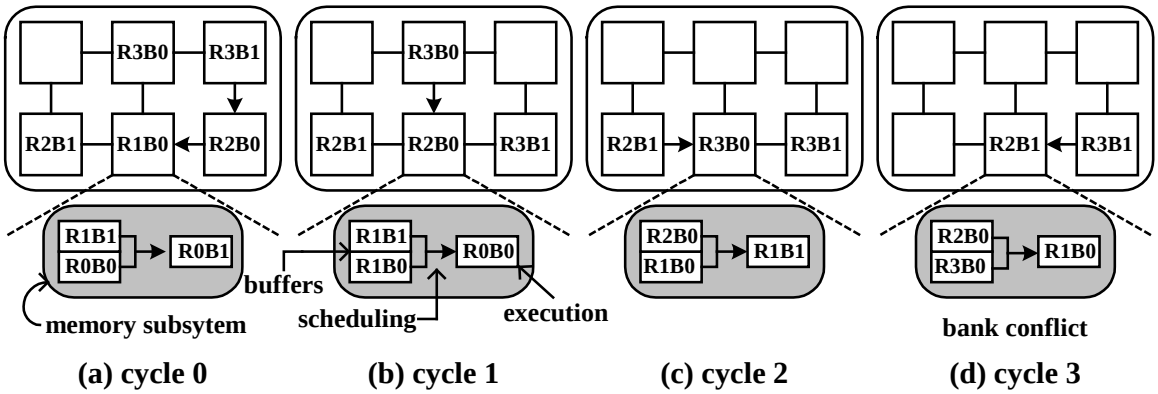


Figure 2.5: Bank conflict in 2×3 NoC with conventional round-robin flows controller although an effective memory subsystem.

In Figure 2.5, round-robin arbitration [19] is adopted as a flow control mechanism of NoC routers to assign a channel and an input buffer of the next node to one packet among several competing packets. At cycle 0, three packets, R2B0, R2B1 and R3B0 get a competition for an advance to the router interconnected to the memory subsystem and we assume that R2B0 wins. R0B1 is executed in the memory subsystem. At cycle 1, R2B0 advances to the router interconnected to the memory subsystem and then R3B1 also advances to the empty router by the advance of R2B0. Then three packets, R2B1, R3B0 and R3B1 also get the competition such that R3B0 wins by round-robin arbitration. In the memory subsystem, R0B0 but not R1B1 is executed for avoiding bank conflict since R0B1 accessing bank 0 is performed at cycle 0. At cycle 2, R3B0 advances in the router interconnected to the memory subsystem and R1B1 is executed in the memory subsystem. Then, two packets, R3B1 and R2B1 get the competition such that R2B1 wins by round-robin arbitration. At cycle 3, bank conflict happens in the memory subsystem since current execution is a bank 0 request and two buffers are also stored with bank 0 requests, where all row addresses are different. Although the efficient memory subsystem is included in the NoC design, it is difficult to avoid bank conflict completely under the limited depth of a buffer and the dynamic SDRAM accesses of processing elements. Data contention and short turn-around bank interleaving can happen in the conventional NoC design by similar mechanism to this example.

2.1.3.2 Basic Idea of Our Approach

In our NoC design, scheduling SDRAM request packets is performed by multiple SDRAM-aware routers. This architecture makes the possibility of bank conflict lower since packets arrive at a memory subsystem in the order that is friendly to SDRAM

operations. Figure 2.6 shows how NoC with our SDRAM-aware router works well without bank conflict. At the first competition (cycle 0) for an advance to the router interconnected to the memory subsystem, the winner is R2B1 accessing bank 1 since the former packet (R1B0) passed in this router accesses bank 0. The rest of packet causes bank conflict since they read/write data in the same bank but different row addresses from the former packet. At cycle 1, R2B1 advances to the router interconnected to the memory subsystem and then R2B0 and R3B0 get the competition. Both can be a winner for the next advance since they access bank 0. In this example, R2B0 is chosen by our SDRAM-aware router. At cycle 2, R2B0 advances to the router interconnected to the memory subsystem and R3B1 avoiding bank conflict wins against R3B0 for the next advance. Finally, R3B1 advances to the router interconnected to the memory subsystem and R3B0 follows R3B1 at cycle 3. As a result, an NoC design with our SDRAM-aware router avoids bank conflict better than an NoC design with the conventional memory subsystem and router.

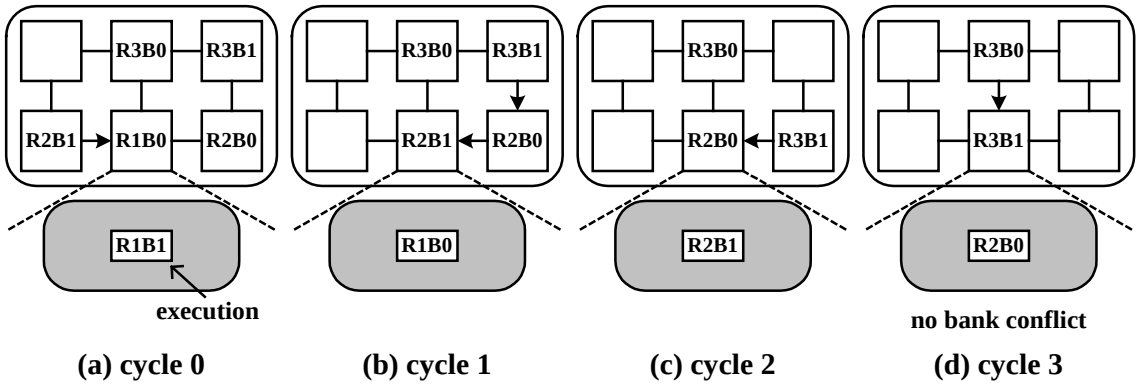


Figure 2.6: No bank conflict in 2×3 NoC with SDRAM-aware flow controller although a simple memory subsystem.

A single memory subsystem usually controls one channel of SDRAM in the conventional NoC, which means the same number of memory subsystem as the number of SDRAM channel is required. Whereas it is allowable to use multiple SDRAMs for high performance, it is not desirable to use a corresponding number of memory subsystems. The reason is that the memory subsystem as shown in Figure 2.5 is too high in terms of hardware cost due to the heavy input buffer and the complex scheduler. Furthermore, a depth of input buffer rapidly increases as a length of packet is longer and longer in a high definition graphics/video system. On the other hand, the proposed architecture saves the NoC design cost since any input buffer and any scheduler are not required in the memory subsystem as shown in Figure 2.6. Instead, a simple flow controller is included in multiple routers, which has a very low hardware cost compared to an input buffer and a scheduler in a memory subsystem. In the next section, we present a novel SDRAM-aware NoC router in detail.

2.1.4 SDRAM-Aware Router

For a wide range of applications, the proposed NoC router is about a novel paradigm for SDRAM-aware-NoC exploration, which has a flow-control mechanism improve memory utilization and memory latency with a cost-effective NoC platform. Indeed, based on our idea present in Section 2.1.3.2, any deterministic and adaptive routing scheme can be combined to implement our SDRAM-aware router. Another flow-control mechanism can be also combined to avoid deadlock and livelock [19], to make traffic load balanced on a network [83][89][94] and to manage buffers and channel bandwidth [58].

2.1.4.1 Router Description

Our NoC router consists of an input buffer, a routing logic, a flow controller and an output scheduler as shown in Figure 2.7. A packet is split into so-called flits (flow control digits) which are then routed and stored in a pipelined fashion. The input buffers are managed by a wormhole flow control mechanism or a virtual-channel flow control mechanism and backpressure is used to inform upstream nodes when they must stop transmitting flits because all of the downstream input buffers are full. For our experiment, the wormhole flow control mechanism is implemented due to its simplicity and wide popularity [19] and an on/off flow control mechanism for the backpressure is employed to avoid a loss of flits.

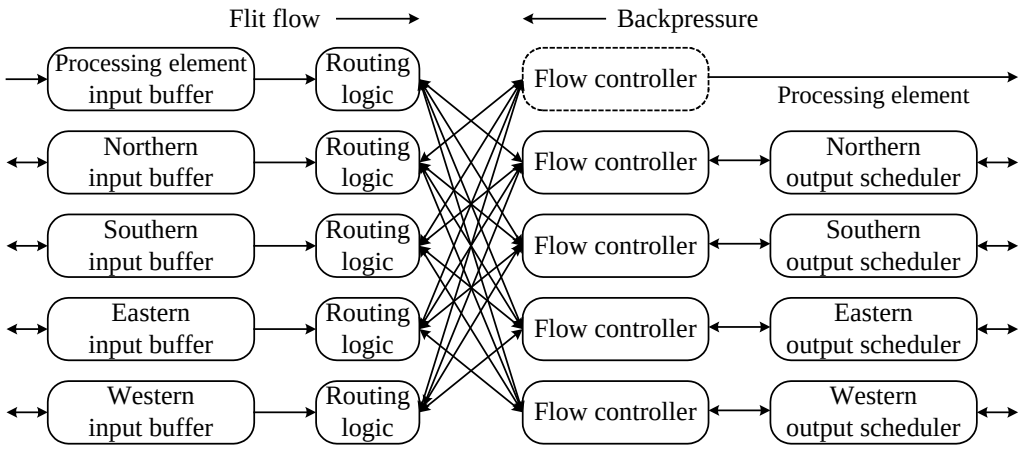


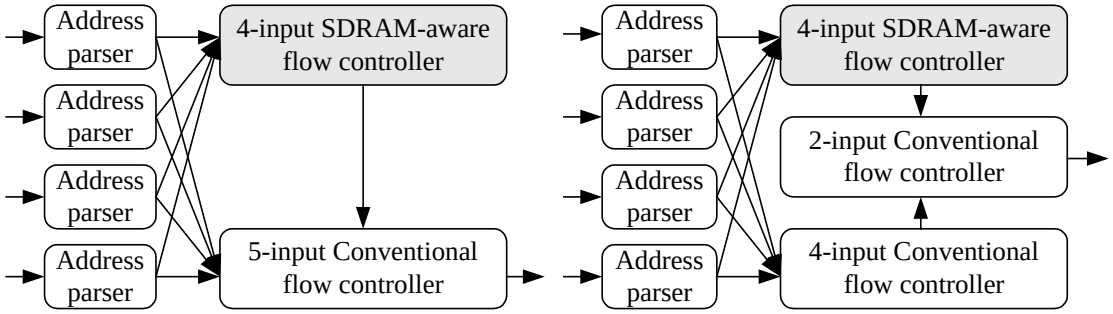
Figure 2.7: The architecture of an SDRAM-aware router consisting of input buffers, routing logics, flow controllers, and output schedulers for a mesh network.

Our SDRAM-aware router can be implemented to either deterministic or adaptive routers according to a routing logic that guarantees deadlock and livelock freeness. Virtual channels and deterministic dimension-ordered routings (e.g. XY routing, odd-even routing) are commonly used to prevent deadlock [19]. We implement XY routing that is a deterministic and minimal path routing algorithm such that it guarantees

deadlock- and livelock-free routing. In addition, we consider an ordering issue when a master core sends a read request to another slave core before the master core receives a read data from one slave core or when a master core requests another read data to a slave core in NoC employing an adaptive router before the master core receives one read data from the slave core. This ordering issue can be solved by [61] or under the following constraint: a master core can send a read request to a slave core only after the master core receives all data requested. The latter solution is employed in our implementation. In addition, since our SDRAM-aware flow control algorithm is performed with in-order buffers, the ordering problem does not happen in each SDRAM-flow control.

In this router, more than two flits arriving on different input buffers at the same time may both desire the same channel toward a memory subsystem. In this situation, our flow-control mechanism resolves this contention, allocating the channel to one packet and dealing with the others, blocked packets. Figure 2.8 shows our SDRAM-aware flow controller combined with the conventional flow controller. In Figure 2.8, an address parser sends an incoming memory request packet to our SDRAM-aware flow controller and an incoming normal packet to the conventional flow controller. Our SDRAM-aware flow controller schedules the memory packets in order to prevent bank conflict, data contention and short turn-around bank interleaving. In the next section, the SDRAM-aware flow-control algorithm using a priority-based arbitration is described minutely. Then, the resulting memory request packet competes with normal packets by the conventional flow control mechanism. Hence, normal packets can reach their destination with no additional communication delay.

Figure 2.8(a) shows its serial implementation. This architecture causes a timing path to be much longer since a 5-input conventional flow control algorithm is performed



(a) Serial implementation (b) Parallel implementation

Figure 2.8: The architecture of an SDRAM-aware flow controller combined with a conventional flow controller for a mesh network.

after performing our 4-input SDRAM-aware flow control algorithm. On the other hand, in Figure 2.8(b), our 4-input SDRAM-aware flow controller for memory packets and a 4-input conventional flow control algorithm for normal packets are parallelly performed. Finally, two resulting packets are scheduled by a 2-input conventional flow controller. This parallel implementation can minimize an increase of timing path whereas its design cost is more expensive than the design cost of the serial implementation. We adopt this parallel implementation in our experiment. In addition, our flow controllers adopt winner-take-all bandwidth allocation that allocates all of the bandwidth to just one packet until it is finished or blocked before serving the other packets [19].

An output scheduler either detects if an input buffer of the next router is available or expects when the input buffer is available. When an input buffer of the next router is full and a deterministic routing logic is implemented, an output scheduler lets the corresponding SDRAM-aware flow controller stop scheduling packets. On the other hand, packets given multiple routing paths performed by an adaptive routing logic can be scheduled to other flow controller less busy.

2.1.4.2 SDRAM-Aware Flow Control for Avoiding Bank Conflict and Data Contention

Our flow control acts to allocate a channel to one of competing flits which destination is a memory subsystem interfacing with SDRAM. Therefore, our flow-control mechanism performs arbitration to determine which flit gets the channel it has requested. After the arbitration, a winning flit advances over this channel. Our arbitration algorithm also decides how to dispose of any flits that do not get their requested channel.

In Algorithm 1 called SP, our arbitration is a priority-based algorithm, where a priority is determined by SDRAM awareness. The priority is assigned to all head flits which destination is a memory subsystem. Let $h(n)$ be a head flit of a packet, which is already allocated a channel by the SDRAM-aware flow control at the nth arbitration. Body and tail flits are assigned the same channel as their head flit. Let $h_i(n+1)$ be one of all competing head flits (I) which should be allocated to the same channel as $h(n)$ by the SDRAM-aware flow control at the $(n+1)$th arbitration, where $i \in I$. The head flits, $h(n)$ and $h_i(n+1)$ contain address and command information to access SDRAM, denoted by (RA_n, BA_n, R/W_n) and ($RA_{n+1,i}$, $BA_{n+1,i}$, $R/W_{n+1,i}$), respectively, where the notations are (row address, bank address, read/write command). At the $(n+1)$th arbitration, all $h_i(n+1)$

Algorithm 1 Scheduling Packet to Avoid Bank Conflict and Data Contention

Input: $h(n)$, $h_i(n+1)$ and Table 2.2

1: **for** each $h_i(n+1)$, $i \in I$ **do**
2: **if** $h_i(n+1)$ is a new packet entering to the router **then**
3: $w_i = 0$;
4: **else**
5: $w_i = w_i$ + waiting cycles from previous arbitration(n);
6: **end if**
7: d_i = delay cycle between $h(n)$ and $h_i(n+1)$ from Table 2.2;
8: $p_i = w_i - d_i$;
9: **end for**
10: $h_i(n+1)$ with maximum(p_i) is allocated to a channel;

Output: $h(n+1)$

are compared to *h(n)* and then are given a delay penalty from Table II (line 7) that is composed from DDR I, II and III SDRAM working at 133MHz to at 800MHz clock frequency (with 266MHz to 1.6GHz data rate) [24].

Table 2.2 shows how many clock cycles waste by bank conflict and data contention or a combination thereof when $h_i(n+1)$ accesses SDRAM after *h(n).* If bank conflict and data contention happen simultaneously, bank conflict is commonly prioritized since bank conflict wastes more clock cycles than data contention. According to a read/write command, a bank address and a row address, there are twelve cases as shown in Table 2.2. Twelve cases are also classified into eight delay types that are described as follows:

Delay a: Case 1 and case 10 have no clock cycle loss since $h_i(n+1)$ is the same read/write command, bank address and row address as *h(n).* These cases indicate that the same row data of the same bank are again accessed by the same command. Thus, the bank does not need to be deactivated and reactivated, which causes the clock cycle loss. In addition, read/write latency of $h_i(n+1)$ can be hidden while *h(n)* is accessed. In Figure 2.3(b), the relation between read 2 and read 4 is case 1 and the relation between write 1 and write 3 is case 10.

Delay b: Case 2 is the read-to-read bank conflict explained in Section 2.1.2.1. Before executing the latter read accessing the same bank but a different row, the bank must be deactivated, i.e. data in the row buffer move to the corresponding row of the bank. Then, the bank must be activated again, which indicates that the row buffer should be again filled with new data for the latter read. Thus, it takes $t_{RP}+t_{RCD}+CL$ to receive data of the latter read after receiving data of the former

Relation of $h(n)$ with RW_n, BA_n, and RA_n and $h_i(n+1)$ with $RW_{n+1,i}$, $BA_{n+1,i}$, and $RA_{n+1,i}$				DDR I SDRAM (MHz)			DDR II SDRAM (MHz)				DDR III SDRAM (MHz)				cas e
				133	167	200	200	267	333	400	400	533	667	800	
RW_n=R	$RW_{n+1,i}$=R	$BA_n=BA_{n+1,i}$	$RA_n=RA_{n+1,i}$	0	0	0	0	0	0	0	0	0	0	0	1[a]
			$RA_n \neq RA_{n+1,i}$	6	8.5	9	9	12	12	18	18	24	30	33	2[b]
		$BA_n \neq BA_{n+1,i}$		0	0	0	0	0	0	0	0	0	0	0	3[c]
	$RW_{n+1,i}$=W	$BA_n=BA_{n+1,i}$	$RA_n=RA_{n+1,i}$	1	1	1	1	1	1	1	2	2	2	2	4[d]
			$RA_n \neq RA_{n+1,i}$	5	7	7	8	11	11	17	17	22	27	30	5[e]
		$BA_n \neq BA_{n+1,i}$		1	1	1	1	1	1	1	2	2	2	2	6[d]
RW_n=W	$RW_{n+1,i}$=R	$BA_n=BA_{n+1,i}$	$RA_n=RA_{n+1,i}$	3	3.5	5	5	6	7	9	10	12	15	17	7[f]
			$RA_n \neq RA_{n+1,i}$	8	11.5	12	12	16	20	24	24	32	40	45	8[g]
		$BA_n \neq BA_{n+1,i}$		3	3.5	5	5	6	7	9	10	12	15	17	9[f]
	$RW_{n+1,i}$=W	$BA_n=BA_{n+1,i}$	$RA_n=RA_{n+1,i}$	0	0	0	0	0	0	0	0	0	0	0	10[a]
			$RA_n \neq RA_{n+1,i}$	7	10	10	11	15	15	23	23	30	37	42	11[h]
		$BA_n \neq BA_{n+1,i}$		0	0	0	0	0	0	0	0	0	0	0	12[c]

[a] Accessing the same row data one more time.
[b] $t_{RP} + t_{RCD} + CL$.
[c] Bank interleaving.
[d] One clock cycle interval between input and output in DDR I/II SDRAM. Two clock cycle intervals between input and output in DDR III SDRAM.
[e] $t_{RP} + t_{RCD} + WL$.
[f] $t_{WTR} + CL$.
[g] $t_{WR} + t_{RP} + t_{RCD} + CL$.
[h] $t_{WR} + t_{RP} + t_{RCD} + WL$.

Table 2.2: SDRAM data input/output delay between $h(n)$ and $h_i(n+1)$.

read. This case is shown in the relation between read 1 and read 2 and in the relation between read 3 and read 4 in Figure 2.2(a).

Delay c: Case 3 and case 12 have no clock cycle loss since bank interleaving is completely performed as mentioned in Section 2.1.2.1. Case 3 is the read-to-read bank interleaving as shown in Figure 2.2(b) and case 12 is the write-to-write bank interleaving. Since a bank address of the latter request is different from that of the former request, the bank accessed by the latter request can be activated while data of the former request are transferred to or from SDRAM. Then, when data of the former request complete to transfer, data of the latter request can be accessed with no loss of clock cycle.

Delay d: Case 4 and case 6 have at least one clock cycle interval between the former read data and the latter write data to avoid data contention in DDR I/II SDRAM as shown in Section 2.1.2.2. In DDR III SDRAM, the latter write command can be executed internal read-to-write command delay time (t_{RTW}) after the former read command. Then, write data can be transferred to SDRAM after write latency (WL). Thus, actual write data are transferred to DDR III SDRAM, two clock cycles after receiving the last read data. Thus, t_{RTW} lets data contention hidden naturally. In Figure 2.3(a), the relation between read 2 and write 3 is case 4. Case 6 is data contention with bank interleaving.

Delay e: In case 5, data contention and bank conflict happen at the same time since it is a read-to-write access and bank addresses of the read request and the write request are same but their row addresses are different. As mentioned

before, bank conflict should be considered preferentially since it wastes more clock cycles than data contention. Before writing data to the different row in the same bank, the row buffer should be idle after reading data and then active. It takes t_{RP} to be idle and t_{RCD} to be active again. Then, data can be written after write latency. Data contention hides naturally behind this bank conflict.

Delay f: Case 7 is the write-to-read data contention when the latter read request accesses data placed in the same bank and row as the former write. Case 9 is also the write-to-read data contention with bank interleaving since the latter read has a different bank address. To read data after writing data placed in the same bank and row or in a different bank, a read command is accepted internal write-to-read command delay time (t_{WTR}) after writing the last data to SDRAM. Then read data are transferred from the SDRAM after read latency.

Delay g: Case 8 causes the longest delay time due to the write-to-read bank conflict. Data contention is ignored since the bank conflict is more critical. The latter read request accesses data placed in the same bank but a different row. Thus, after the former write request, the bank should be idle. t_{WR} is required to accept a precharge command for deactivation after writing the last data and it takes t_{RP} to complete the precharge command. Furthermore, t_{RCD} is required to activate the row buffer for the latter read request. Then, data can be received read latency after accepting a read command. Thus, total delay time is $t_{WR}+t_{RP}+t_{RCD}+CL$.

Delay h: Case 11 is the write-to-write bank conflict since the latter write request accesses the same bank but a different row from the former write request. As presented in the previous case, *delay g*, it takes a bank t_{WR} and t_{RP} to be idle after writing the last data. Then, its row buffer gets active with row data for the latter write request. It takes t_{RCD} to be active and write data are finally transferred to SDRAM after write latency.

Our priority-based arbitration guarantees the upper bound latency even if a high delay penalty of packet given from Table 2.2 lasts for a long time. For example, let a packet with case 11 lose a competition against a packet with case 10. If it meets another packet with case 10 at the next competition, the defeated packet keeps losing the competition since the delay penalty is not changed. Thus, the defeated packet is required to escape from this competition after several defeats. To solve this starvation problem, our flow control counts the number of clock cycle passed from the first competition to the current competition (line 5) for each defeated packet. Then, this waiting clock cycle is subtracted by the delay clock cycle obtained in Table 2.2 (line 8). By this operation, any packet delayed for the amount of the worst delay (case 8) does not have a lower priority than a new packet entered in the router. For example, in DDR III SDRAM working at 533MHz and 800MHz clock frequency, the packets waiting for 32 and 45 clock cycles get higher priority than any new packet entered in the router respectively.

Finally, the packet with the maximum p_i is allocated to a channel (line 10). In our SDRAM-aware flow control, the packet with longer waiting cycle and shorter delay cycle gets a higher priority. Then, the rest of packet that is blocked waits for the next competition or get another competition at a different SDRAM-aware flow controller if

multiple routing paths are allocated by a routing logic. Thus, if an adaptive router instead of a deterministic router is employed in a routing logic, the performance would be better.

2.1.4.3 SDRAM-Aware Flow Control for Avoiding Short Turn-Around Bank Interleaving

As mentioned in Section 2.1.2.3, a short turn-around bank interleaving problem is not critical for low-performance SDRAM like DDR I SDRAM since a bank has sufficient time to be deactivated or reactivated until the bank is accessed again The reason is that short deactivation ($t_{WR}+t_{RP}$ for writing and t_{RP} for reading) or reactivation time (t_{RCD}) is hidden behind the process of accessing a different bank. On the other hand, deactivation, reactivation and read/write latency time are so long in high-performance SDRAM that it is difficult for them to hide behind the process of accessing a different bank. For example, in DDR III SDRAM working at 800MHz clock frequency, it takes a bank 23, 11 and 11 clock cycles to deactivate, reactivate and output data, respectively after writing data. Thus, before the written bank is again read with a different row address, a scheduler should let different banks accessed for at least 23 clock cycles to improve memory utilization.

The proposed SP algorithm just schedules memory request packets to prevent bank conflict and data contention. Hence, it should check whether a bank accessed by $h_i(n+1)$ is given sufficient deactivation time before the bank is activated. It is well explained together with hardware/architecture of an SDRAM interface signal generator. Figure 2.9 is an SDRAM interface signal generator commonly used, where an input packet passes three buffers to generate SDRAM commands such as a PRE command, an ACT command and an R/W command. First, an input packet arriving at an SDRAM interface signal generator is stored in a deactivation buffer but not an activation buffer as

shown in Figure 2.9. That means a bank keeps activating after accessing data, called open page mode. It can be useful for high memory utilization since most of the cores access data placed in continuous memory addresses, i.e., the same bank and row addresses but different column addresses. Then, if the input packet accesses the bank previously activated with a different row address, a PRE command is output to an SDRAM interface signal controller to deactivate the bank and then the packet moves to an activation buffer. On the other hand, if the input packet accesses the bank previously activated with the same row address or if the input packet accesses the bank already deactivated, the packet just passes a deactivation buffer with no PRE command. If a packet stored in an activation buffer accesses the bank deactivated, the packet lets an ACT command generated to an SDRAM interface signal controller and then moves to a read/write buffer. Finally, a packet stored in a read/write buffer always lets an R/W command generated to an SDRAM interface signal controller. An SDRRM interface signal controller receives a PRE command, an ACT command and an R/W command from those buffers and then generates the final interface signals to SDRAM.

To solve the short turn-around bank interleaving problem happening in this SDRAM interface signal generator, a packet that is output from a deactivation buffer should pass $(t_{WR})+t_{RP}$ until the packet is output from an activation buffer. In addition, a packet that is output from an activation buffer should pass t_{RCD} until the packet is output from a read/write buffer. Since the deactivation time is longer than the activation time and read/write latency, the interval between packets accessing the same bank and a different row should be at least t_{RP} or $t_{WR}+t_{RP}$ depending on a read request or a write request that the previous packet accessing each bank is.

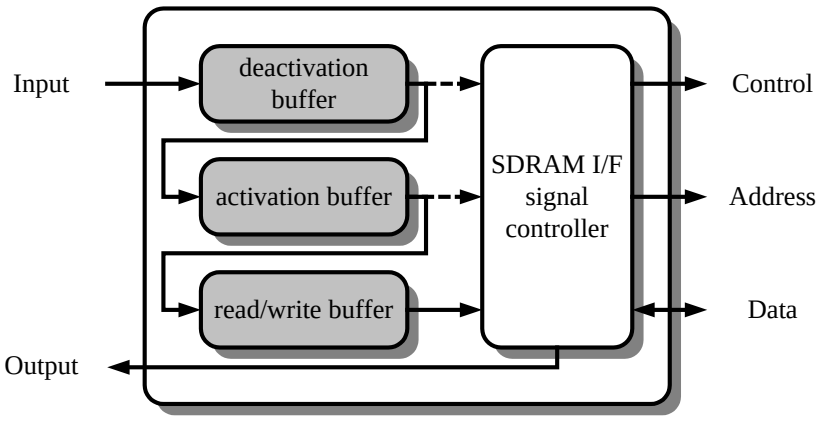


Figure 2.9: The architecture of an SDRAM interface signal generator with a deactivation buffer, an activation buffer, and a read/write buffer which packets pass through.

Algorithm 2 called AP is executed instead of line 8 in our SP algorithm to solve the short turn-around bank interleaving problem. In the AP algorithm, a clock cycle (d_{is}) required to deactivate each bank is recorded after a read/write operation is completed. Thus, the same number of counter as the number of bank is required to save and count d_{is}. If the packet *h(n)* performed is a write request, d_{is} of bank that *h(n)* accesses is set to $t_{WR}+t_{RP}$ in line 4. If the packet *h(n)* performed is a read request, d_{is} of bank that *h(n)* accesses is set to t_{RP} in line 6. Then, all d_{is} are reduced by 1 every clock cycle in line 9. If any packet, $h_i(n+1)$ is in case 3, 6, 9, and 12 of Table 2.2 with *h(n)*, our AP algorithm checks if the bank accessed by the $h_i(n+1)$ has sufficient deactivation time (line 11-15). For this operation, d_{id} captures d_{is} in line 12 if the relation between *h(n)* and $h_i(n+1)$ is case 3, 6, 9, and 12. Otherwise, d_{id} is 0 in line 14. Then, d_{id} is compared to d_i obtained from Table 2.2. Finally, larger delay time is chosen as effective delay time and then subtracts a waiting clock cycle as shown in line 16. Our solution makes banks accessed as uniformly as possible such that the banks get the sufficient time to be deactivated for the next request.

Algorithm 2 Assigning Priority to Avoid Short Turn-Around Bank Interleaving

Input: w_i, d_i, $h(n)$ and $h_i(n+1)$

```
for every clock cycle do
    if h(n) is done then
        if h(n) is write request then
            d_is of bank(h(n)) = t_WR + t_RP;
        else
            d_is of bank(h(n)) = t_RP;
        end if
    end if
    d_is = d_is – 1 for all d_is;
end for
if relation of h(n) and h_i(n+1) is case 3, 6, 9 and 12 then
    d_id = d_is of bank(h_i(n+1));
else
    d_id = 0;
end if
p_i = w_i – max(d_i, d_id);
```

Output: p_i

2.1.4.4 Hardware Complexity

Memory scheduling is performed by our SDRAM-aware flow controller included in multiple NoC routers instead of a single memory subsystem. Thus, simple logics are added for our SP algorithm to compute SDRAM access delay (d_i) and waiting time (w_i) whereas a buffer and a scheduler of memory subsystem are removed as shown in Figure 2.6. A buffer in a memory subsystem is used to store several packets and then to reorder the packets for successive delivery of SDRAM data. However, as the massive size of packet is recently generated in graphics processing units (GPU) and a high-definition video system, the size of buffer gets larger. The proposed NoC design does not require any buffers in a memory subsystem since memory scheduling is performed in multiple NoC routers and the maximum four input buffers per router in a regular mesh network substitute for a buffer in a memory subsystem. In addition, the size of input buffer in the router does not increase according to the size of packet since the input buffer is managed

by wormhole flow control. Consequently, a distinguished hardware decrease by a buffer in a memory subsystem exceeds a hardware increase by the SDRAM-aware flow controller in multiple routers such that total gate count is reduced.

2.1.5 Experimental Results

Our SDRAM-aware NoC router is implemented with Verilog hardware description language (HDL). We implement a memory subsystem operating for DDR I SDRAM working at 133MHz and 200MHz clock frequency, DDR II SDRAM working at 266MHz, 333MHz and 400MHz clock frequency and DDR III SDRAM working at 533MHz and 800MHz clock frequency [24] which all consist of four banks. The memory subsystem is implemented with a design concept of Sonics MemMax [75] and Denali Databahn [23]. MemMax offers a sophisticated thread-based pipeline and advanced arbitration schemes which prevent bank conflict and data contention conditions. Because there are no ordering requirements between threads, requests from different threads can be freely reordered. Different bandwidths and the qualities of service (QoSs) may be allocated to different threads to effectively support system data flow requirements. In MemMax, users can choose the depth of buffers, operation modes, and QoS settings that best suit various applications. Since MemMax supports OCP where request signals and data signals are separated, MemMax requires both a request buffer and a data buffer per thread. We use 4-thread MemMax where each thread requires a 32-flit request buffer and a 32-filt data buffer. The Databahn is an SDRAM controller that optimizes RAS, CAS, PRE, and refresh operation. Since the Databahn employs command look-ahead to prepare pages in memory in advance of when commands execute, it can give class-leading performance even if the pattern of traffic is not known at design time. Both are included

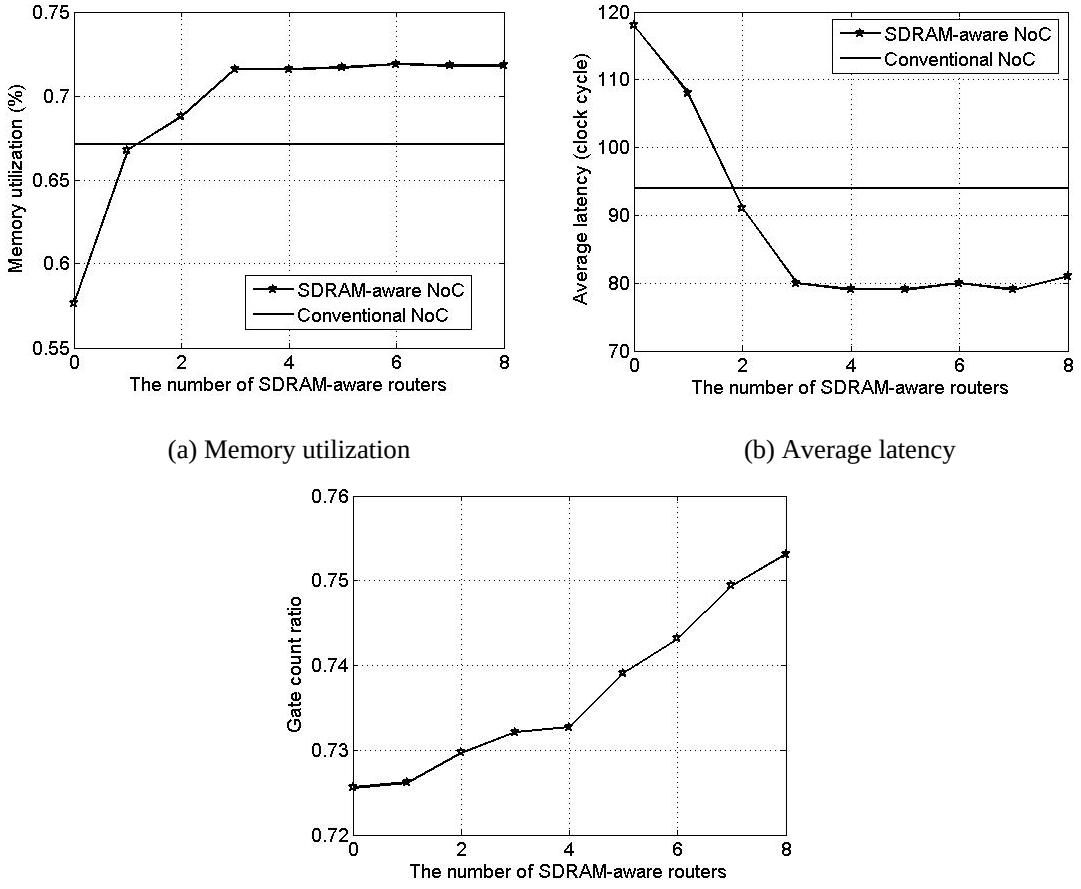
in the conventional NoC design with a round-robin flow control based router. This is compared to our NoC design including multiple SDRAM-aware routers and an SDRAM interface signal generator instead of a full memory subsystem. Applications are mapped to mesh grid by A3MAP [51] and each simulation runs for one million clock cycles.

2.1.5.1 Digital Television Application

The conventional NoC design and our SDRAM-aware NoC design are applied to a Samsung DTV system that consists of nine subsystems, i.e., a central processing unit (CPU) that consists of ARM and several peripherals, a moving picture experts group (MPEG) decoder, a digital natural image engine (DNIE), GPU, an audio decoder, a transport stream (TS) decoder, an audio/video (AV) format converter, a channel decoder and a memory subsystem that interfaces with DDR II SDRAM working at 333MHz clock frequency. In the conventional NoC design, a router using a round-robin flow control algorithm is gradually replaced with our SDRAM-aware router using the proposed SP algorithm in the order where the router that is the closest to a memory subsystem is replaced firstly and where the router that is the farthest away from a memory subsystem is replaced lastly. Figure 2.10 shows the results depending on the number of SDRAM-aware router placed in the order.

In Figure 2.10(a) and (b), memory utilization and memory latency achieved by the conventional NoC design are 67.2% and 94 cycles, respectively. Memory utilization and memory latency performed by our SDRAM-aware NoC design is just 57% and 119 cycles, respectively, in case that there are no input buffer and no memory scheduler in a memory subsystem and no SDRAM-aware router. However, whenever our SDRAM-aware router is substituted for the conventional router in the DTV system, memory

utilization and memory latency improves rapidly. As a result, when three SDRAM-aware routers are substituted for three conventional routers, memory utilization increases up to 72% (that is 7.1% better than the conventional NoC design). However, more than four SDRAM-aware routers do not improve memory utilization any more since the solvable bank conflict and data contention are almost prevented by three SDRAM-aware routers.



(a) Memory utilization

(b) Average latency

(c) Gate count ratio

Figure 2.10: The comparisons of memory utilization, latency, and design complexity in DTV application according to the number of SDRAM-aware routers, where our NoC design achieves the best tradeoff between performance and cost when three conventional routers are replaced to SDRAM-aware routers.

Similarly, in Figure 2.10(b), memory latency is also shortened by three SDRAM-aware routers up to 79 cycles (that is 16% shorter than the conventional NoC design) since high memory utilization makes a packet performed as fast as possible and our SDRAM-aware flow controller manages the upper bound latency.

Our SDRAM-aware NoC design and the conventional NoC design are synthesized by Synopsys Design Vision with a TSMC130LV library. The gate count of our SDRAM-aware NoC design is 26.8% smaller when three round-robin routers are replaced with our SDRAM-aware routers in Figure 2.10(c). In addition, its gate count is 24.8% smaller even if all round-robin routers are replaced with our SDRAM-aware routers. The reason is that a large buffer and a complex scheduler in a memory system are removed whereas an additional hardware increased by our SDRAM-aware flow controller is minimal.

We also implement the SDRAM-aware NoC based a DTV system interfacing with a variety of DDR SDRAMs working at 133Mhz to 800MHz clock frequency. Our DTV system works for real-time computing when it interfaces with DDR II SDRAM working at 333MHz clock frequency. However, to show the benefit of our SDRAM-aware NoC design in various DDR SDRAMs, we let a packet injection rate of each IP changed similar to a change of the SDRAM clock speed. Table 2.3 shows memory utilization and latency in our NoC design including three SDRAM-aware routers compared to the conventional NoC design. Our SDRAM-aware NoC design proves more merits on high-performance DDR SDRAM in Table 2.3. For example, our SDRAM-aware NoC improves more 3.7% memory utilization and 14.3% memory latency than the conventional NoC when they all interface with DDR I SDRAM working at 133MHz clock frequency. On the other hand, our SDRAM-aware improves more 26% memory utilization and 30.8% memory latency than the conventional NoC when they all interface

with DDR III SDRAM working at 800MHz clock frequency. Since timing constraints caused by bank conflict is about six times longer in DDR III SDRAM working at 800MHz clock frequency than in DDR I SDRAM working at 133MHz clock frequency, our SDRAM-aware NoC design achieves better improvement of memory utilization and latency in DDR SDRAM operating at a fast clock frequency.

The SDRAM-aware NoC design implemented by our SP algorithm is also applied into dual DTV model [99] containing dual MPEG decoders and dual memory subsystems. Consequently, the improvement of memory utilization and latency is similar to a single memory subsystem. However, it saves more than 42% gate count compared to dual DTV model implemented by the conventional NoC design since our SDRAM-aware NoC design does not need eight 32-flit request and data buffers and two complex memory schedulers in a dual memory subsystem.

2.1.5.2 Synthetic Benchmarks

We evaluate the improvement of memory utilization and memory latency obtained from several randomly generated applications on industrial intellectual properties (IP) with DDR II SDRAM working at 333MHz clock frequency. The SDRAM-aware router adopts the SP algorithm and all of the conventional routers are replaced with our SDRAM-aware routers. The IPs are mapped into 3×3 to 6×6 mesh network by A3MAP [51] and generate 4 to 32 flits per packet at dynamic intervals. Table 2.4 shows our SDRAM-aware NoC improves 11.8% memory utilization and 18% memory latency on average compared to the conventional NoC. In particular, the improvement of memory utilization and memory latency is higher in 6x6 NoC than in 3x3 NoC since packets passing through more SDRAM-aware routers have more

Measure	Router	DDR I SDRAM		DDR II SDRAM		DDR III SDRAM	
		133MHz	200MHz	266MHz	400MHz	533MHz	800MHz
Memory utilization	Conventional NoC	73.1%	72.2%	70%	61.4%	52.6%	43.5%
	SP	75.8%	74.5%	74.8%	68%	62.3%	54.8%
	Difference (improvement)	**2.7% (3.7%)**	**2.3% (3.2%)**	**4.8% (6.9%)**	**6.6% (10.7%)**	**9.7% (18.4%)**	**11.3% (26%)**
Memory latency	Conventional NoC	84 cycles	85 cycles	96 cycles	110 cycles	126 cycles	146 cycles
	SP	72 cycles	76 cycles	76 cycles	83 cycles	90 cycles	101 cycles
	Improvement	**14.3%**	**10.6%**	**20.8%**	**24.5%**	**28.6%**	**30.8%**

Table 2.3: Memory utilization and latency comparison in DTV application according to various DDR SDRAMs.

Measure	Router	3x3 NoC	4x4 NoC	5x5 NoC	6x6 NoC	average
Memory utilization	Conventional NoC	59.4%	58.7%	52.9%	53.2%	56.1%
	SP	63.7%	65.5%	60.3%	61.2%	62.7%
	Difference (improvement)	**4.3% (5.6%)**	**6.8% (11.6%)**	**7.1% (13.4%)**	**8.0% (15%)**	**6.6% (11.8%)**
Memory latency	Conventional NoC	65cycles	79 cycles	94 cycles	99 cycles	84 cycles
	SP	59 cycles	66 cycles	80	71 cycles	69 cycles
	Improvement	**9.2%**	**16.4%**	**14.9%**	**28.3%**	**18%**

Table 2.4: Memory utilization and latency comparison in synthetic benchmarks according to network size.

opportunities to be scheduled well for SDRAM operations. Therefore, we can expect that the improvement of memory utilization and memory latency would be greater in larger or complex NoC.

2.1.5.3 Comparison of SP and SP+AP

We evaluate the improvement of memory utilization and latency of SP+AP algorithm that considers the short turn-around bank interleaving problem. For this experiment, we use a 4x4 mesh network including three SDRAM-aware routers and execute several randomly generated applications on industrial IPs. Table 2.1 shows the SP+AP algorithm achieves better memory utilization and memory latency than the SP algorithm in particular in high-performance DDR SDRAM. As shown in Table 2.5, the short turn-around bank interleaving problem is not critical in low-performance DDR SDRAM since the improvement of memory utilization and latency achieved by the SP+AP algorithm is just around 1% compared to the SP algorithm. On the other hand, it causes memory performance critically degraded when high-performance DDR SDRAM is adopted in an NoC design. For example, in DDR III SDRAM working at 800MHz clock frequency, the SP+AP algorithm achieves 9.2% higher memory utilization and 9.2% shorter memory latency than the SP algorithm. The proposed SP+AP algorithm requires an additional hardware such as four counters, one comparator and some control logics to check each bank state. However, the additional circuitry is tiny.

2.1.6 Summary

This section presented an SDRAM-aware NoC design where multiple NoC routers adopting our SDRAM-aware flow control algorithm allocate an SDRAM access

Measure	Router	DDR I SDRAM		DDR II SDRAM		DDR III SDRAM	
		133MHz	200MHz	266MHz	400MHz	533MHz	800MHz
Memory utilization	Conventional NoC	67%	69.3%	61.8%	51.9%	38.8%	30.9%
	SP	68.2%	69.9%	64.3%	54.9%	47.8%	39.2%
	SP+AP	69.1%	70.5%	66.1%	58.3%	51.4%	42.8%
	Difference (improvement)*	**0.9% (1.3%)**	**0.6% (0.9%)**	**1.8% (2.8%)**	**3.4% (6.2%)**	**3.6% (7.5%)**	**3.6% (9.2%)**
Memory latency	Conventional NoC	85	87 cycles	96 cycles	115 cycles	151 cycles	186 cycles
	SP	83 cycles	84 cycles	92 cycles	106 cycles	126 cycles	152 cycles
	SP+AP	83 cycles	84 cycles	90 cycles	100 cycles	114 cycles	138 cycles
	Improvement*	**0%**	**0%**	**2.2%**	**5.7%**	**9.5%**	**9.2%**

* It is the difference (improvement) between SP and SP+AP.

Table 2.5: Memory utilization and latency comparison of SP and SP+AP in DDR I/II/III SDRAM.

packet to a channel for the efficient SDRAM operation. Our SDRAM-aware flow control algorithms solve three memory scheduling problems, such as bank conflict, data contention, and short turn-around bank interleaving to improve memory utilization and latency. The proposed SP algorithm solves the bank conflict problem and the data contention problem and the proposed SP+AP algorithm solves the short turn-around bank interleaving problem. Experimental results show that our SDRAM-aware flow controller adopting the SP algorithm delivers superior memory utilization and latency with the small design cost compared to the conventional NoC design. In addition, our SP+AP algorithm achieves higher memory performance than the SP algorithm in particular in high-performance DDR SDRAM. Our SDRAM-aware router actives better performance improvement when it is employed in complex NoC or its routing scheme is adaptive. In conclusion, the proposed SDRAM-aware router provides more opportunities to support bandwidth-hungry NoC designs with the small hardware cost.

2.2 APPLICATION-AWARE NOC DESIGN

In Section 2.1, our NoC design provides a best-effort memory service as each SDRAM-aware NoC router equally manages all memory request packets. However, since the latest real-time applications request a memory service with short latency, a priority memory service should be also provided for cores sensitive to memory latency. Furthermore, different applications request various sizes of memory data. In the state-of-the-art multimedia system, the length of memory request packets requested by a video encoder/decoder like H.264 [115] gets shorter whereas the length of memory request packets requested by a video enhancer/format converter gets longer. The long best-effort packets cause a priority packet to be further delayed. If any long best-effort packet is

already scheduled in a router, a priority packet may wait until the best-effort packet is completely transferred to the next router. On the contrary, the short packets cause SDRAM utilization to be severely deteriorated. Since most SDRAMs receive or transmit fixed-length data per read/write command, SDRAM data unnecessarily acquired may be thrown away. Therefore, a NoC design should also consider the access granularity of diverse applications for an efficient SDRAM access.

Since such guaranteed throughput and bounded latency are essential for NoC designs, many researchers have developed various approaches [71]. Æthereal NoC proposed in [31] provided a guaranteed service combined with a best-effort service employing variants of time division multiplexing. In [76], Nostrum NoC was implemented with the service of guaranteed bandwidth and latency in addition to the existing service of best-effort. In [5], MANGO using clockless circuit techniques was implemented. It exploited virtual channels to provide connection-oriented service guarantees and connection-less best-effort routing. Kim et al. proposed router architecture which utilized adaptive routing while maintaining low latency [58]. BiNoC supporting a self-configuring bidirectional channel mechanism for better bandwidth utilization and lower packet delivery latency was proposed in [63]. Das et al. in [21] proposed efficient prioritization policies and architectural extensions to NoC routers that improved the overall application-level throughput, while ensuring fairness in the network. The prioritization policies were application-aware, distinguishing applications based on the stall-time criticality of their packets. In [22], they also proposed router prioritization policies that exploited the available slack of interfering packets in order to accelerate performance-critical packets and thus improved overall system performance. However, these approaches are not optimized for SDRAM request packets that cause the most critical latency.

In this section, we propose an application-aware NoC design to efficiently access shared SDRAMs [52][55]. Our key motivations are twofold. First, some cores request a guaranteed SDRAM service to an on-chip network and a memory subsystem. For example, a demand request generated by a microprocessor is usually served as a priority packet since the microprocessor may halt until the demand request is served. However, the priority packet causes overall memory latency and utilization to be severely degraded. Since an on-chip network first serves the priority packet without any consideration of SDRAM operations, there exits strong possibility to meet bank conflict, data contention and short turn-around bank interleaving in SDRAM, which all make memory performance deteriorated. Therefore, the priority memory service should be considered not only in a memory subsystem but also in an on-chip network. In addition, since long best-effort packets may interfere with the fast service for the priority packet, they should be split to several short packets and then served. Second, different cores request various-length SDRAM data whereas DDR I/II SDRAMs always generate fixed-length data. Even if DDR III SDRAM can generate variable-length data, it has few advantages due to CAS to CAS delay time (t_{CCD}) [24]. If the length of data requested by cores is not either same as the length of data served by SDRAM or a multiple of the length of data served by SDRAM, unnecessary data may be accessed and then thrown away. Therefore, the access granularity mismatch problem resulting in low memory performance is considered in our application-aware NoC design. Based on these motivations, the major novelties and contributions of this section include the following.

- We propose a guaranteed SDRAM service (GSS) router. It provides an efficient priority service for cores sensitive to memory latency.

- We propose an SDRAM access granularity matching (SAGM) NoC design. Since a packet is split to several short packets of which the size is equal or less than SDRAM access granularity and then served by our GSS router and memory subsystem without a memory scheduler and a number of buffers, unnecessary SDRAM data can be less accessed.
- We show the hardware architecture of our GSS router and memory subsystem working with a partially open-page policy and an auto- precharge (AP) operation.
- We show the GSS router significantly improves memory latency for a priority packet with few penalties of overall memory utilization and latency. In addition, the SAGM NoC design not only recovers the penalties but also further improves overall memory performance.

To the best of our knowledge, this is the first work that addresses a NoC design improving the quality of memory service through application-aware manners. The rest of this section is organized as follows. In Section 2.2.1, we introduce two problems of conventional application-unaware NoC designs. Section 2.2.2 presents the detail description of the proposed application-aware NoC design. Experimental results are shown in Section 2.2.3. Finally, Section 2.2.4 summarizes Section 2.2.

2.2.1 Problem Description and Our Basic Idea

2.2.1.1 Priority SDRAM Service in NoC

A microprocessor including a general processor, a cache and a prefetcher commonly generates a demand request and a prefetch request. The demand request should be served as soon as possible since the microprocessor may stall until it receives a

service of the demand request. On the contrary, the prefetch request does not need to be served with such a priority since it may be useless or not promptly used by the microprocessor. Memory requests of multimedia processors and peripherals are commonly handled similarly to the prefetch request in the latest video/graphics systems.

Most of the conventional memory scheduler or NoC router takes two different approaches as to how to treat a priority request with respect to others. Figure 2.11(b)-(c) show the operation of three different memory schedulers when two demand memory requests, two prefetch memory requests and two memory requests by specific video processors are filled in their input buffer as shown in Figure 2.11(a). In the figure, BA means a bank address and all requests are read operations. In addition, the RAs of all requests are different except prefetch 2 and request 2.

A memory scheduler providing a best-effort service as shown in Figure 2.11(b) regards a priority memory request to have the same priority as others and then schedules all memory requests to avoid bank conflict, data contention, and short turn-around bank interleaving and to encourage row-buffer hit and bank interleaving. As a result, all memory requests are successively executed with no bank conflict whereas the execution of demand 2 is considerably delayed, which may cause the microprocessor generating demand 2 to halt for a long time. On the contrary, in Figure 2.11(c), the demand requests are executed with a priority. This approach makes the demand requests executed early. However, since demand 2 accesses the same bank as demand 1 access with a different RA, bank conflict happens. It causes any data not to be delivered while the row buffer of bank 1 becomes deactivated and then is filled with the data of demand 2. Consequently, since total execution time of six requests is longer, memory utilization gets deteriorated. Therefore, a memory scheduler providing a priority service without the loss of memory utilizaton is required.

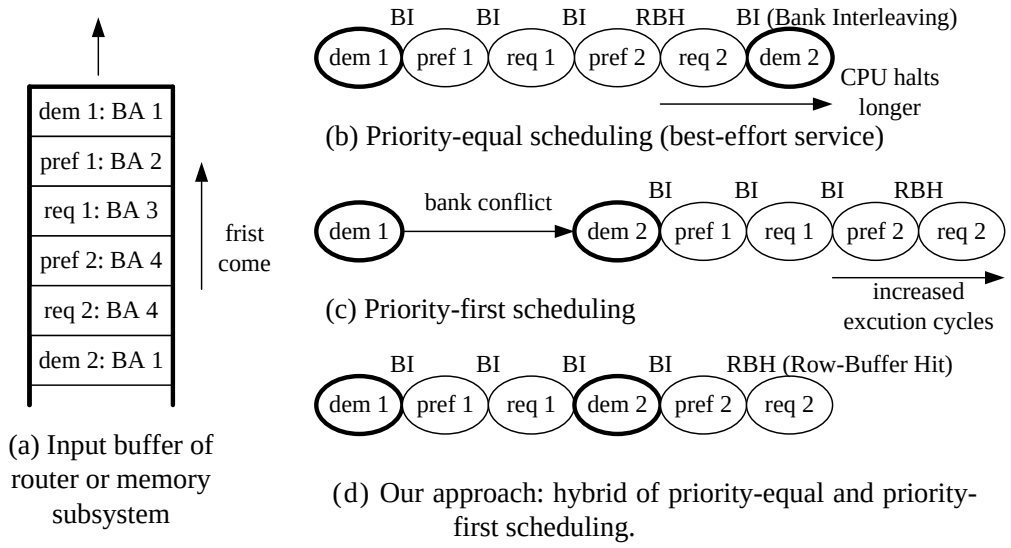


Figure 2.11: Examples of scheduling memory requests, where priority-equal and priority-first schedulers show long latency for priority packets and low memory utilization, respectively.

Figure 2.11(d) is the most desirable scheduler that achieves the same memory utilization as the best-effort scheduler and the same memory latency for the demand requests as the priority-first scheduler. In order to achieve this performance, we propose a hybrid flow control algorithm that gets the advantage of the priority-equal and priority-first scheduler, which is fully described in Section 2.2.2. There may be strong possibility to meet bank conflict, data contention, and short turn-around bank interleaving if a demand request is separately considered on an on-chip network and in a memory subsystem. Therefore, this scheduling is performed by multiple NoC routers which are similar to the SDRAM-aware NoC design proposed in Section 2.1.

Moreover, we consider a long best-effort packet interfering with the fast service of priority packets. In the advanced video/graphics system, the length of a packet is longer and longer to provide a high-quality image. For example, the length of a packet generated by an industrial video enhancer/format converter reaches 64 burst lengths

(BLs), which means that it takes at least 64 clock cycles to transfer the packet to the next router [99]. It is usually served as a best-effort packet. Let a router employing winner-take-all bandwidth allocation [19] schedule the long best-effort packet to any channel and then a priority packet reach in this router. If the router allocates the priority packet to the same channel as the long best-effort packet, the priority packet must wait until the long best-effort packet is completely delivered. In order to solve this problem, we split all packets to several short packets and then served. As a result, the priority packet can get more opportunities to be allocated to the channel. In the video/graphics system with 64-BL packets, if the best-effort packet is split to several packets with 4 BLs, a priority packet will wait for the maximum 4 clock cycles and then get the next competition. In the proposed application-aware NoC design, the length of a packet split is determined by an SDRAM access granularity introduced in the next subsection.

2.2.1.2 SDRAM Access Granularity Mismatch

SDRAMs transfer/receive fixed-length data (= the number of data bit × BL) per CAS command, called SDRAM access granularity. DDR I SDRAM has a BL 2, BL 4 and BL 8 mode and DDR II/III SDRAM has a BL 4 and BL 8 mode. In addition, since DDR III SDRAM has a selectable BL 4 or BL 8 on-the-fly (OTF) mode, it can deliver data with 4 or 8 BLs, depending on address 12 pin without any BL mode change. For example, if SDRAM with 16-bit data bus is set to a BL 8 mode via mode register set (MRS), it always generates 16 bytes per CAS command as shown in Figure 2.12. On the contrary, any cores may request data with various lengths to SDRAMs. For example, an MPEG-1/2 and H.264 [115] encoder/decoder requests 8 or 16 bytes and 4, 8 or 16 bytes for motion estimation/compensation to SDRAM, respectively. If the MPEG-1/2 or H.264

encoder/decoder requests just 8 bytes as shown in Figure 2.12, the rest of data unnecessarily accessed are thrown away, which seriously degrades memory performance.

Simple solutions are to reduce the number of data bits or to use a short BL mode in DDR SDRAM. If the number of data bits is changed to 8 bits, there exists no wasteful data. However, the overall system interfacing with SDRAM with 8-bit data bus does not have sufficient memory bandwidth to feed all cores. If more SDRAMs are interfaced with the entire system in order to increase the memory bandwidth, additional memory subsystems and pins/wires that are the limited resources are required. On the contrary, when short BL modes such as BL 2 and BL 4 are used in DDR SDRAM, command bandwidth exceeds data bandwidth such that it is difficult to hide commands behind data input/output time. The reason is that BL 2 and BL 4 have just one and two spaces where commands can be executed, respectively, whereas three spaces per SDRAM access are always required to execute three commands such as RAS, CAS, and PRE, except for a row-buffer hit condition. If there exists no row-buffer hit condition, memory utilization cannot exceed 33.4% and 66.7% in BL 2 and BL 4.

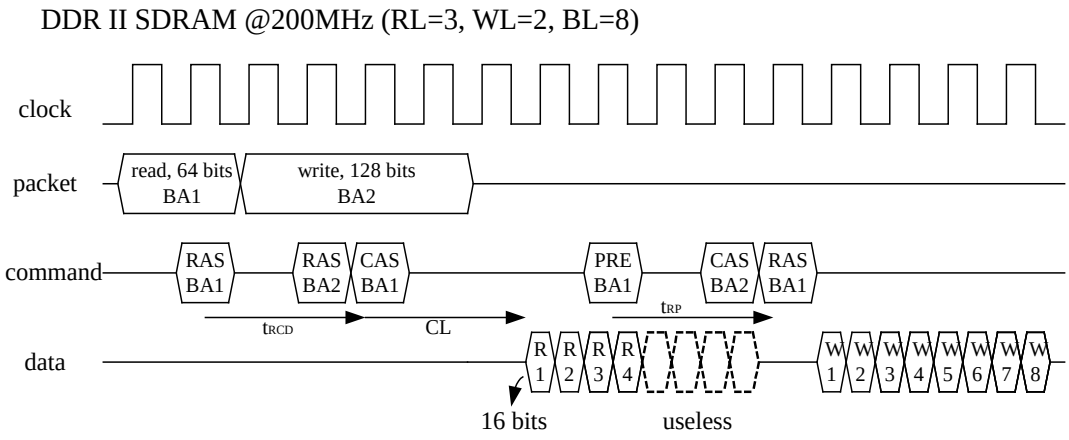


Figure 2.12: Example of memory access granularity mismatch in DDR II SDRAM @200, where four bursts read are thrown away.

For this SDRAM access granularity mismatch problem, we focus on the latter approach using a BL 4 mode. In order to overcome the shortage of a command execution space, we use an auto-precharge (AP) operation in a memory subsystem. When AP is executed with a CAS command, the row buffer of an accessed bank automatically becomes idle without a PRE command after finishing transferring or receiving SDRAM data. In addition, a packet is split to several short packets with the same BL as SDRAM or less BL of SDRAM and then served by an on-chip router and a memory subsystem in our application-aware NoC design. As mentioned in Section III.B, splitting a packet to several short packets is also helpful to a priority service when a best-effort packet is too long. The detail approach is described in Section 2.2.2.

2.2.2 Application-Aware NoC Design

Even with a perfect network routing algorithm and a perfect flow control algorithm mentioned in [71], a priority memory request may be significantly congested and delayed in a memory subsystem if it reaches the memory subsystem with the order unfriendly to SDRAM operations. In addition, if the length of data requested by cores is different from that of data served by SDRAM, data unnecessarily accessed are thrown away. In this regime, our attention shifts to an application-aware NoC design to improve not only the overall memory performance but also the quality of memory service.

2.2.2.1 Architecture of GSS Router

The proposed GSS router with p input/output ports consists of an input buffer, a routing logic, a flow controller and an output scheduler as shown in Figure 2.13. Typically, p is 5 and 7 for 2D and 3D mesh networks, respectively. The input buffers are

managed by a wormhole flow control mechanism or a virtual-channel flow control mechanism. For our experiment, the wormhole flow control mechanism is implemented due to its simplicity and wide popularity [19]. The routing logic is responsible for determining the next router for each packet. Our GSS router can be implemented to either deterministic or adaptive routers according to a routing logic that guarantees both deadlock and livelock freeness. For our experiment, we implement XY routing that is a deterministic and minimal path routing algorithm such that it guarantees deadlock-free and livelock-free routing.

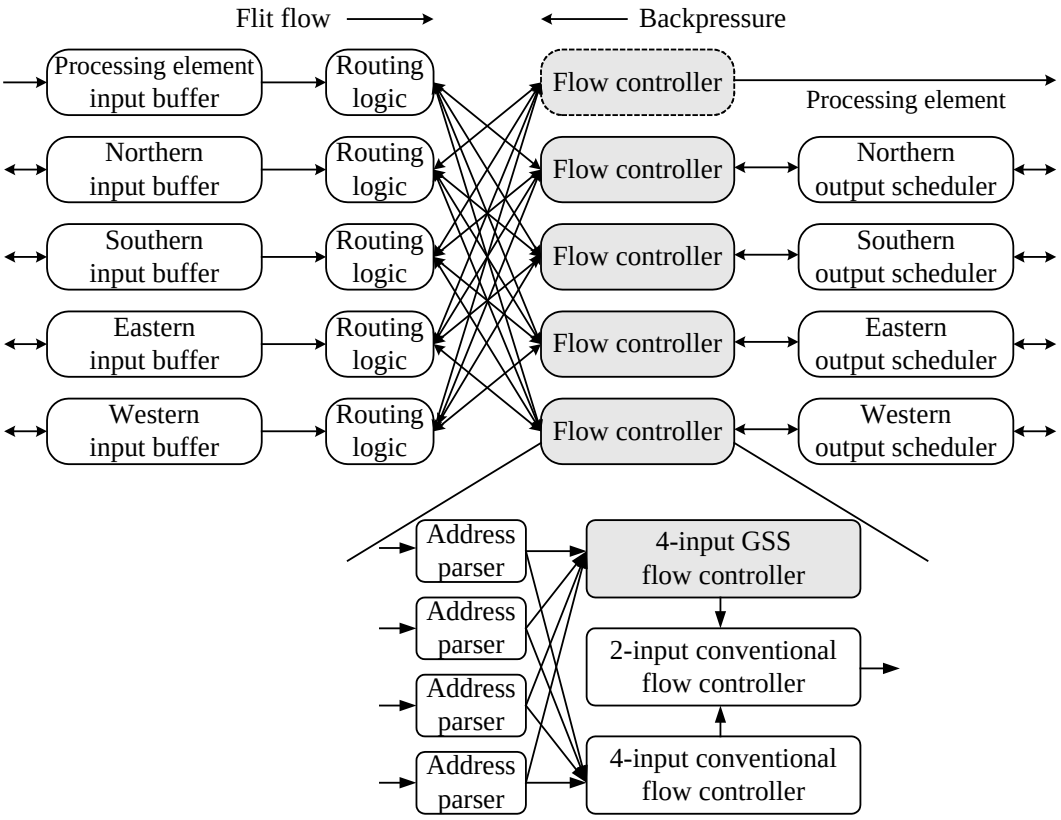


Figure 2.13: The architecture of an NoC router and a GSS flow controller for a 2D mesh network.

In this router, more than two different packets arriving on input buffers at the same time may desire the same channel toward a memory subsystem. In this situation, our GSS flow-control mechanism resolves this contention, allocating the channel to one packet and dealing with the others, blocked packets. In Figure 2.13, our GSS flow controller is parallelly performed with the conventional flow controller. Each address parser sends an incoming memory request packet to our GSS flow controller and an incoming normal packet to the conventional flow controller. Our GSS flow controller schedules the memory request packets in order to prevent bank conflict, data contention, and short turn-around bank interleaving and provide a priority service at the same time. Then, the resulting memory request packet again competes with a normal packet by the conventional flow control mechanism. Hence, normal packets can reach their destination with no additional communication delay and interference. This parallel implementation can minimize an increase of timing critical path whereas its design cost is slightly expensive. In addition, our flow controllers adopt winner-take-all bandwidth allocation that allocates all of the bandwidth to just one packet until it is finished or blocked before serving the other packets [19].

An output scheduler either detects if an input buffer of the next router is available or expects when the input buffer is available. When the input buffer of the next router is full and a deterministic routing logic is implemented, an output scheduler makes the corresponding GSS flow controller stop scheduling packets. On the contrary, packets given multiple routing paths by an adaptive routing logic can be scheduled to other GSS flow controllers which is not busy.

In addition, we consider an ordering issue when a master core sends a read request to another slave core before the master core receives a read data from one slave core or when a master core requests another read data to a slave core in NoC employing

an adaptive router before the master core receives one read data from the slave core. This ordering problem can be solved by various previous works including [61] or a following constraint: a master core can send a read request to a slave core only after the master core receives all requested data. The latter solution is employed in our implementation for simplicity. In addition, since our GSS flow control algorithm is performed with in-order buffers, the ordering problem does not happen in each GSS flow control.

2.2.2.2 GSS Flow Control Algorithm

In this section, we minutely present our flow control algorithm providing short latency for a priority memory request packet and similar overall memory utilization and latency. Let *h(n)* be a packet already allocated any channel by our GSS flow control at the *n*th arbitration. Let $h_i(n+1)$ be any packet *i* of all completing packets, *H(n+1)*, which may be allocated the same channel as *h(n)* by our flow controller at the (*n+1)*th scheduling. The packets, *h(n)* and $h_i(n+1)$ contain an address and a command to access SDRAM, denoted by (RA_n, BA_n, R/W_n) and ($RA_{n+1,i}$, $BA_{n+1,i}$, $R/W_{n+1,i}$), respectively, where the notations are (row address, bank address, read/write). Thus, bank conflict, data contention, bank interleaving and row-buffer hit conditions are defined as (BA_n=$BA_{n+1,i}$ and $RA_n \neq RA_{n+1,i}$), ($RW_n \neq RW_{n+1,i}$), ($BA_n \neq BA_{n+1,i}$) and (BA_n=$BA_{n+1,i}$ and RA_n=$RA_{n+1,i}$), respectively. Based on these notations and definitions, Algorithm 3 shows how our flow controller works for a guaranteed memory service, which consists of two parts.

First, a memory request packet (*i*) is given some tokens (t_i), depending on its input order and priority (line 1-13). Let a new packet come in a router. All of the old packets are given to one additional token to avoid starvation (line 3). Then, if the new packet has a priority, old best-effort packets accessing the same bank as the priority packet are

Algorithm 3 GSS Flow Control

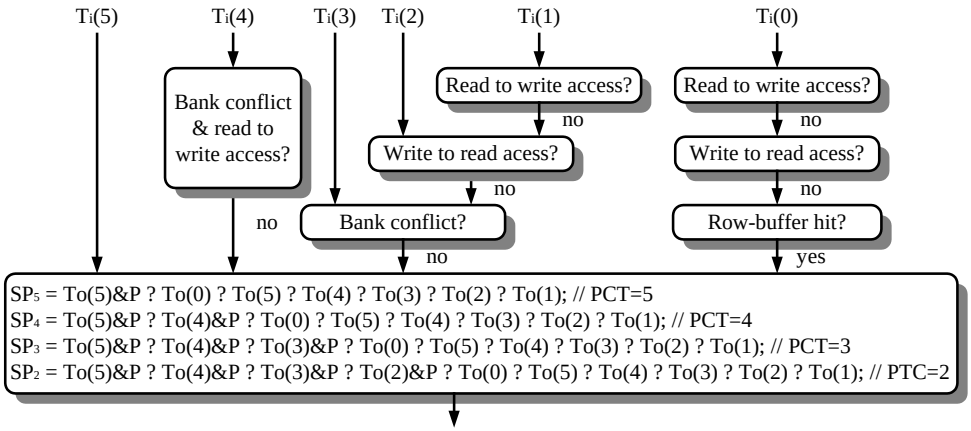
```
1:  if new packet h_k(n+1) comes in each router then
2:    for h_i(n+1) ∈ H(n+1) do
3:      t_i ← t_i+1;
4:      if h_k(n+1) is priory packet and its BA is equal to that of h_i(n+1) that is best-
        effort packet then
5:          h_i(n+1) is except from H(n+1);
6:      end if
7:    end for
8:    if h_k(n+1) is priority packet then
9:      t_k ← 2 to 5 (or 6); // PCT for Figure 2.14(a) (or (b))
10:   else
11:     t_k ← 1; // best-effort packet
12:   end if
13: end if
14: if h(n) finishes being delivered to the next router then
15:   for h_i(n+1) ∈ H(n+1) do
16:     T_i(t_i) in Figure 2.14 ← h_i(n+1);
17:     T_i(0) in Figure 2.14← h_i(n+1);
18:   end for
19:   if SP_PCT = ∅ then
20:     for h_i(n+1) ∈ H(n+1)   do
21:       t_i ← t_i+1;
22:     end for
23:     Go to line 14;
24:   end if
25: end if
```

except from *H(n+1)* (line 5). It means that old best-effort packets that access the same bank as any priority packet are not scheduled until the priority packet is scheduled. Then, the new packet gets an initial token. If it is a best-effort packet, one token is given (line 11). Otherwise, more than two tokens are given (line 9) to a priority packet by a user, called a priority control token (PCT). If a single token is given to the priority packet, it is equal to a priority-equal scheduler and if the maximum tokens are given to the priority packet, it is equal to a priority-first scheduler. Therefore, we can control the service speed of a priory packet by PCT.

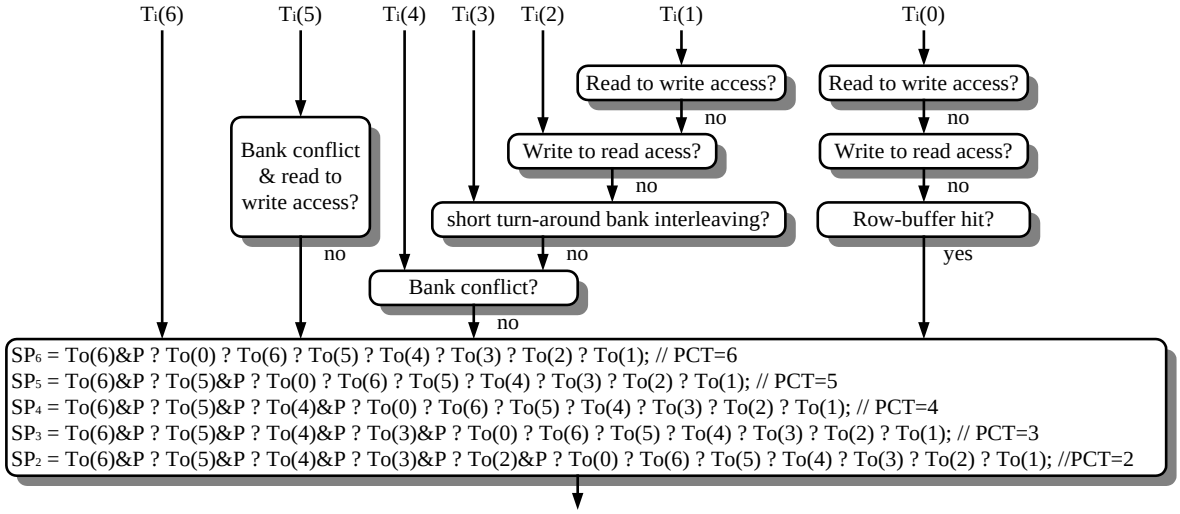
Second, when *h(n)* finishes being delivered, the rest of packets, *H(n+1)* in the router are scheduled (line 14-25). They all are input to Figure 2.14, according to the number of tokens each packet has. That is, if any packet has 1, 2, 3, 4, 5, and 6 tokens, the packet is input to $T_i(1)$, $T_i(2)$, $T_i(3)$, $T_i(4)$, $T_i(5)$, and $T_i(6)$, respectively (line 16). All of the packets are also input to $T_i(0)$ in line 17. As mentioned in Section III.A, a short turn-around bank interleaving problem is not critical for DDR SDRAM working at a low clock frequency since a short deactivation clock cycle and a reactivation clock cycle can be hidden behind the process of accessing a different bank. For such DDR SDRAMs, our flow controller just resolving bank conflict and data contention is shown in Figure 2.14 (a). On the contrary, since it takes a number of clock cycles to finish deactivation and reactivation in DDR SDRAM working at a high clock frequency, it is difficult for them to hide behind the process of accessing different banks. For example, in DDR III SDRAM working at an 800MHz clock frequency, it takes 23 clock cycles to deactivate any bank after writing data [24]. Thus, until the written bank finishes being deactivated, a flow controller should make different banks accessed for 23 clock cycles to improve memory performance. Therefore, a flow controller working for such DDR SDRAMs should consider not only bank conflict and data contention but also short turn-around bank interleaving as shown in Figure 2.14(b).

In order to check whether each bank finishes being idle, our flow controller has the same number of a counter as the bank of DDR SDRAM. After the last data are transferred to SDRAM, a counter corresponding to a bank written is set to $t_{WR}+t_{RP}$, where t_{WR} and t_{RP} are write recovery (WR) time and row precharge (RP) time, respectively [24]. On the contrary, after the last data are received from SDRAM, a counter corresponding to a bank read is set to t_{RP}. Then, the delay cycle stored in the counter is reduced by 1 every clock cycle. Thus, in Figure 2.14(b), the short bank turn-around bank interleaving

condition is defined as the counter corresponding to a bank accessed by $h_i(n+1)$ is greater than 0. If it is true, the bank is not ready to be activated again. Otherwise, the bank finishes being deactivated.



(a)



(b)

Figure 2.14: Scheduling memory request packets for guaranteed SDRAM service considering (a) bank conflict and data contention, and (b) bank conflict, data contention and short turn-around bank interleaving.

Finally, the packets are differently filtered in Figure 2.14, depending on the number of token and the priority. If any packet has a few token, which means an old packet or a priority packet, it is easy to pass this filter. After filtering all packets, if there is no packet passing the filter (line 19), all packets are given one additional token (line 21) and then go to the input of the filter again (line 23). Finally, if there are any packets passing the filter, one among the packets is output to SP_{PCT} (Scheduled Packet). If PCT is *n* in line 9, SP_n is used in Figure 2.14 where $T_o(t_i)$ is the filtered output of $T_i(t_i)$. SP_n=A?B?C means A is chosen if A is not 0. If A is 0 and B is not 0, B is selected. Finally, if both A and B are 0 and C is not 0, C is chosen. In Figure 2.14, a packet with a priority (P) and the most tokens is first selected. Next, a packet with $T_o(0)$ is selected. Lastly, a best-effort packet with the most tokens is selected. The reason that the packet with $T_o(0)$ is preferred to the best-effort packet with the most tokens is that there is strong possibility that $h(n)$ and $h_i(n+1)$ is split from the same packet. Why they are split from the same packet will be explained in the next section.

2.2.2.3 NoC Design for SAGM

It is useful to split a long packet into several short packets since on-chip network resources can be efficiently reserved and an SDRAM access granularity mismatching problem can be easily solved. That is, the optimal length of packets can improve memory/network utilization/latency. We split a packet to several short packets, depending on an SDRAM access granularity. Since our GSS routers communicate through a famous open core protocol (OCP) [86] or an AMBA AXI/AHP [2] protocol, packets consist of body flits but not head and tail flits including routing information. Instead, more controls and address buses include the routing information. Therefore, even

if a packet is split to several short packets in each core and then is injected on a network, network loads do not increase.

As mentioned in Section 2.2.1.2, DDR I/II SDRAMs always transfer/receive fixed-length data per read/write command after any BL mode is set in MRS. Most of the memory subsystems prefer a BL 8 mode in DDR I/II SDRAM because a BL 2 mode and a BL 4 mode can cause command bandwidth to be severely limited. As DDR SDRAMs transfer/receive two data burst per clock cycle, data are transferred/received for one and two cycles in the BL2 and BL4 mode, respectively. However, without any row-buffer hit, SDRAM needs three commands such as RAS, CAS and PRE to obtain the short data. Therefore, the commands are so congested that the execution of commands is delayed. As shown in Figure 2.15, we assume that a PRE command for BA 1 and a CAS command for BA 2 are issued at the same time. In Figure 2.15(a), the PRE command is performed earlier than the CAS command. Consequently, the data of the second packet are written with some delays. In Figure 2.15(b), the CAS command is performed earlier than PRE command. Consequently, the bank 1 gets idle and active with some delays. Therefore, such command congestion should be solved when short BL modes are used.

Fortunately, SDRAMs can omit a PRE command if a CAS command is executed with an auto-precharge (AP). The AP is enabled to provide a self-timed row precharge that is initiated at the end of burst access. As a result, both the PRE command and the CAS command are not delayed due to AP, as shown in Figure 2.15(c). Under this consideration, it is useful that the BL (granularity) of packets is 2 and a BL mode in DDR I/II SDRAM is set to 4. Now that DDR III SDRAM has a selectable BL4 or BL8 OTF mode, it is useful that the BL of packets is 4 and a BL mode in DDR III SDRAM is set to 8. For example, if the BL of any packet is 9, it is split to five packets whose BLs are 2, 2, 2, 2 and 1 for DDR I/II SDRAM and it is split to three packets whose BLs are 4, 4 and 1

for DDR III SDRAM. It is efficient not only to match the access granularity but also to manage network resources. That is, a priority packet can be served faster in a winner-take-all bandwidth allocation policy. If the length of any best-effort packet is 9, a priority packet waits until all 9 bursts of the best-effort packet are transferred. If it is split like our approach, a priority packet wait until the maximum 2, 2 and 4 bursts of the best-effort packets are transferred in DDR I, II and III SDRAM, respectively and then get more opportunities to be allocated to a channel.

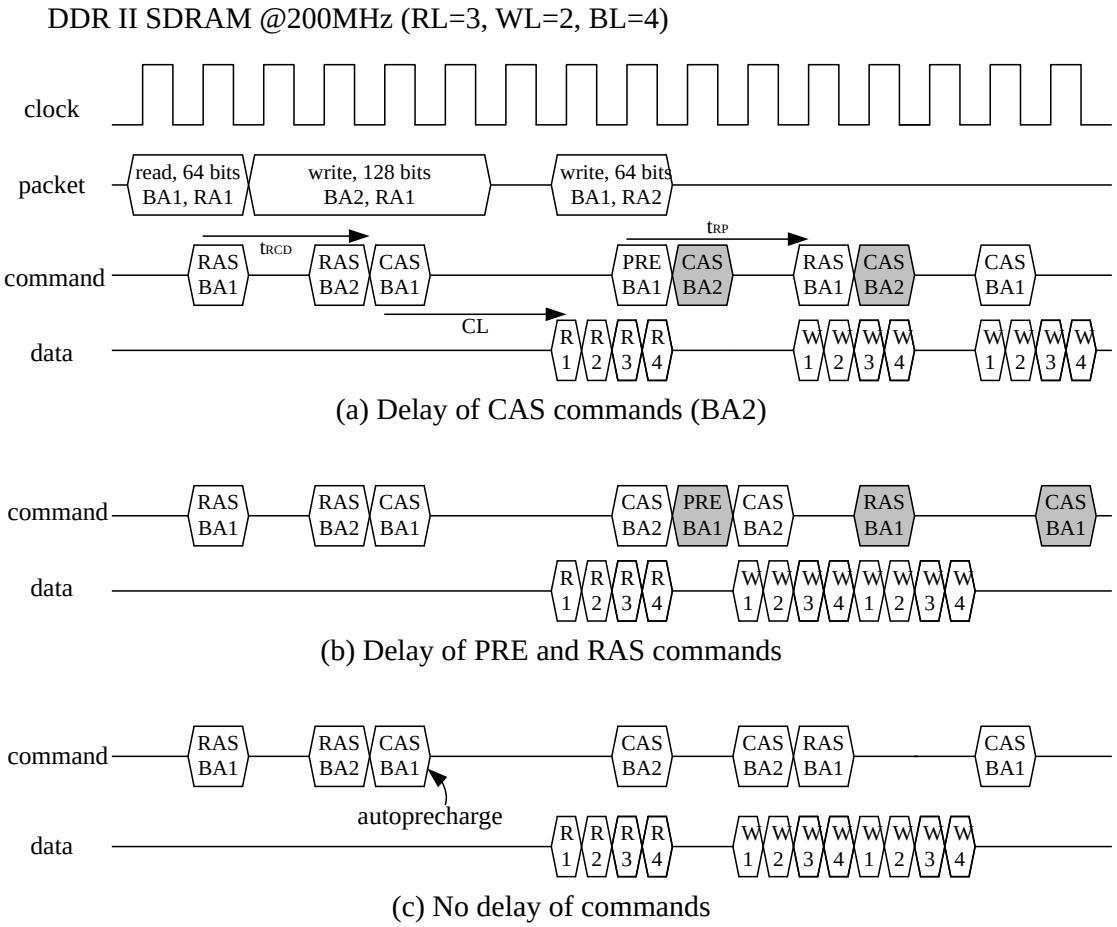


Figure 2.15: SDRAM Operations when BL is set to 4 in DDR II SDRAM @300MHz, where the read command with authoprecharge does not need any precharge command.

To implement this idea, we make a core generate short packets whose granularity is 2, 2 and 4 in DDR I, II and III SDRAM, respectively and the last packet has a tag to execute AP. Since the relation of packets split is row-buffer hit, there is not any loss of memory performance. As explained in IV.B, our GSS router prefers the row-buffer hit condition to the bank interleaving condition even if both do not cause any loss of memory performance. Therefore, if split best-effort packets do not meet any priority packet, they are scheduled successively. On the contrary, a priority packet is always scheduled without any interference.

Figure 2.16 shows our memory subsystem. Since memory scheduling is performed in multiple GSS routers, our memory subsystem consists of an SDRAM controller, but not a complex memory scheduler and a number of buffers. Our SDRAM controller makes DDR SDRAMs work for a partially open-page mode. Each bank keeps an active state (open-page) after being accessed by a packet without any tag indicating the last packet split from a long packet. However, if a bank is accessed by a packet with a tag, the bank is deactivated (closed-page) by AP. In addition, when a priority packet meets bank conflict relation with the previous best-effort packet, the bank is closed even if the previous best-effort packet has no tag. Our SDRAM Controller works by this concept.

A memory request packet that is input to our SDRAM controller is decoded to extract SDRAM access information such as BA, RA, column address (CA), the length of data, the type of a command, write data (if the command is a write request) and a master address. Then, the master address of read requests is stored in an output buffer and then used for building a memory service packet when requested data are received from SDRAM. The write data is stored in a data buffer and then used for generating an SDRAM interface signal for a write operation. The rest of SDRAM access information is

stored in a PRE buffer. Then, the PRE buffer issues a PRE command only if a priority packet has any bank conflict relation with the previous best-effort packet without any tag. Since AP performing with a CAS command can be substituted for the PRE command, a number of PRE buffers are not required. The information stored in the PRE buffer is again stored to a RAS buffer. The RAS buffer issues a RAS command only if a packet does not have any row-buffer hit relation with the previous packet. The information stored in the RAS buffer is again stored in a CAS buffer. The CAS buffer always issues a read/write command. If a tag is attached to any information, its command is executed with AP. Next, all PRE, RAS and CAS commands are scheduled by a command scheduler with a round-robin policy. Finally, an SDRAM interface signal generator builds SDRAM interface signals for each command and then sends them to SDRAM.

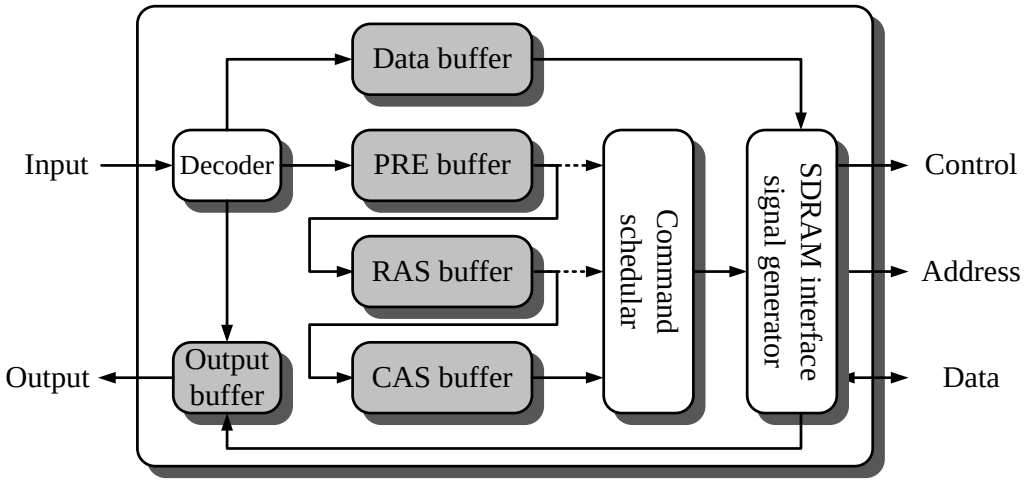


Figure 2.16: The architecture of our memory controller where small PRE and RAS buffers are required thanks to authoprecharge operations.

2.2.3 Experimental Results

The proposed application-aware NoC is implemented in Verilog HDL. The memory subsystem interconnected to DDR I/II/III SDRAM with 32-bits data bus [24] employs the design concepts from Sonics' MemMax [75] and Denali's Databahn [23]. Both the MemMax and the Databahn are employed in the conventional NoC design with a round-robin flow control based router, called CONV. The conventional NoC design and the SDRAM-aware NoC design set DDR SDRAMs to a BL 8 mode via MRS. They are compared to our application-aware NoC design where DDR I/II SDRAM are set to a BL 4 mode and DDR III SDRAM is set to a selectable BL 4 or BL 8 OTF mode.

We use a Blu-ray model [115], a DTV model and a dual DTV model [99] as applications, which consist of 9, 9 and 16 cores, respectively. A memory subsystem is placed in a upper left corner and the applications are mapped to 3×3, 3×3 and 4×4 mesh network, respectively by A3MAP [51] as shown in Figure 2.17. The multimedia systems can work for various video sizes to measure memory performance in different DDR SDRAMs. For example, let dual DTV work for two video streams with 1920×1088 pixels, interfacing with 400MHz DDR II SDRAM for real-time computing. If the dual DTV interfaces with 200MHz DDR I SDRAM and 800MHz DDR III SDRAM, it works for two video streams with 1280×720 pixels and 2560×1600 pixels, respectively. All simulations run for one million cycles.

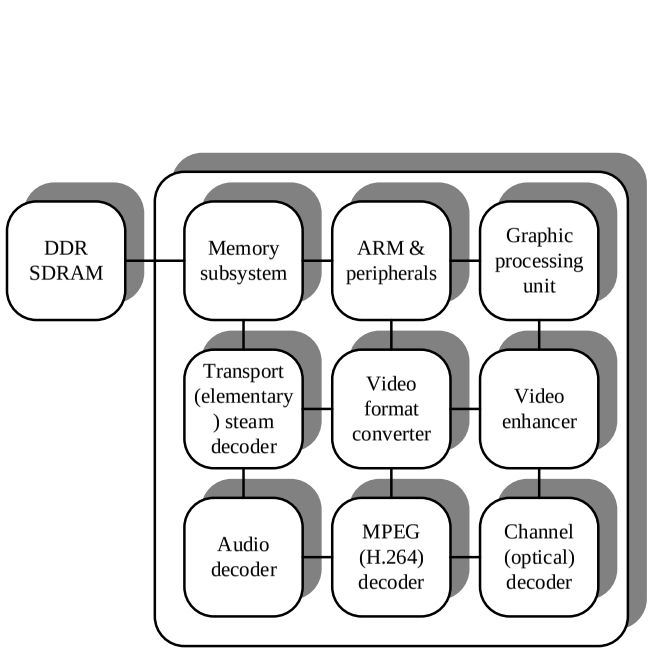
2.2.3.1 No Priority Memory Request

Our application-aware NoC design is first experimented when there is not any priority packet. Since a demand packet generated by a microprocessor or a cache is not assigned to a priority packet, all packets receive a best-effort service. We implement the proposed application-aware NoC design to two versions. One is that only a GSS router is

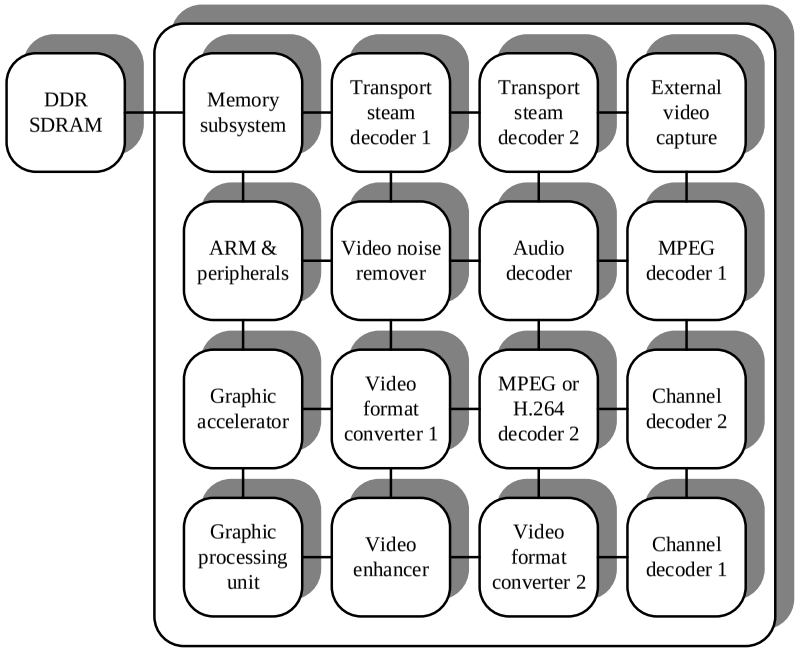
employed and the other is that both a GSS router and an SAGM design are employed in our NoC design, called GSS and GSS+SAGM, respectively.

Table 2.6 shows their memory performance, where the performance ratio is based on the SDRAM-aware NoC design presented in Section 2.1, called SANoC. The GSS router achieves slightly better overall memory utilization and latency than SANoC even if it is optimized for the latency of priority memory requests. On the contrary, the GSS router shows slightly worse latency of demand packets, compared to SANoC. However, the latency of demand packets is not important since the demand packets are not assigned to a priority packet. Our NoC design employing both the GSS router and the SAGM design achieves not only higher memory utilization and shorter memory latency for overall requests, but also much shorter latency for the demand requests than SANoC and GSS designs.

As shown in Table 2.6, the proposed application-aware NoC design with SAGM is the most useful for DDR II SDRAM where a read operation cannot be interrupted by any write and a write operation cannot be interrupted by any read and precharge operations. In DDR I SDRAM, a read operation can be interrupted by a burst stop command to support a short-burst data. However, since a write operation cannot be still interrupted, our SAGM design can improve memory performance in DDR I SDRAM. Now that DDR III SDRAM has a selectable BL4 or BL8 on-the-fly (OTF) mode, it looks perfect for the SAGM. However, in DDR III SDRAM, a CAS command can be performed only 4 clock cycles after the previous CAS command due to t_{CCD} (CAS to CAS delay time). It makes DDR III SDRAM similarly works for a BL8 mode even if the BL mode is not set to 8. Therefore, our performance improvement in DDR III SDRSAM is less than that in DDR I/II SDRAM.



(a) Single DTV/Blue-ray model in 3×3 mesh network



(b) Dual DTV model in 4×4 mesh network

Figure 2.17: Single DTV/blue-ray and dual DTV application mapping results by A3MAP in 3x3 and 4x4 mesh networks.

Application	Clock speed	Memory utilization				Memory latency of all packets (cycle)				Memory latency of demand packet(cycle)			
		CONV	SANoC	GSS	GSS+ SAGM	CONV	SANoC	GSS	GSS+ SAGM	CONV	SANoC	GSS	GSS+ SAGM
Blu-ray	133MHz[a]	0.755	0.763	0.771	0.774	121	81	74	69	111	63	65	60
	266MHz[b]	0.651	0.691	0.717	0.761	157	109	101	86	153	91	89	74
	533MHz[c]	0.505	0.592	0.600	0.619	216	134	140	131	216	113	124	113
Single DTV	166MHz[a]	0.717	0.737	0.766	0.776	144	101	86	71	140	80	74	61
	333MHz[b]	0.625	0.673	0.715	0.756	173	120	108	91	171	96	94	77
	667MHz[c]	0.463	0.554	0.577	0.596	244	154	143	140	248	126	127	119
Dual DTV	200MHz[a]	0.696	0.707	0.708	0.712	154	104	89	80	128	73	67	57
	400MHz[b]	0.555	0.627	0.627	0.682	246	149	141	115	196	107	104	85
	800MHz[c]	0.426	0.559	0.531	0.547	364	191	195	184	266	133	144	128
Average		0.599	0.656	0.668	0.691	202	127	120	107	181	98	99	86
Ratio[d]		0.914	1.000	1.018	1.054	1.591	1.000	0.942	0.846	1.847	1.000	1.007	0.878

[a] DDR I SDRAM
[b] DDR II SDRAM
[c] DDR III SDRAM
[d] Ratio is based on SANoC

Table 2.6: Memory performance comparison on industrial benchmarks without priority memory requests.

2.2.3.2 Priority Memory Request

We test the proposed application-aware NoC design on priority packets. Since a demand packet generated by a microprocessor or a cache is assigned to a priority packet, it is served earlier than a best-effort packet. We also implement the conventional NoC design and the SDRAM-aware NoC with a priority-first service (PFS), called CONV+PFS and SANoC+PFS, respectively.

Table 2.7 shows their memory performance, where the ratio is based on SANoC in Table 2.6. Our application-aware NoC design proves more merits when there exists a priority packet on NoC. SANoC+PFS improves, on average, the latency of priority memory request packets up to 20.7%, compared to SANoC. However, the memory utilization and latency of all packets are 8.3% and 23.3% worse than SANoC. On the contrary, our GSS router improves, on average, the latency of priority memory request packets up to 23.7%, compared to SANoC. The memory utilization and latency of all packets are just 1.7% and 2.9% worse than SANoC. Compared to SANoC +PFS, our GSS router improves, on average, 7.7% memory utilization, 16.5% latency of all packets and 3.7% latency of priority packets. This result shows our GSS router has fewer penalties of memory performance than SANoC +PFS to support a priority service.

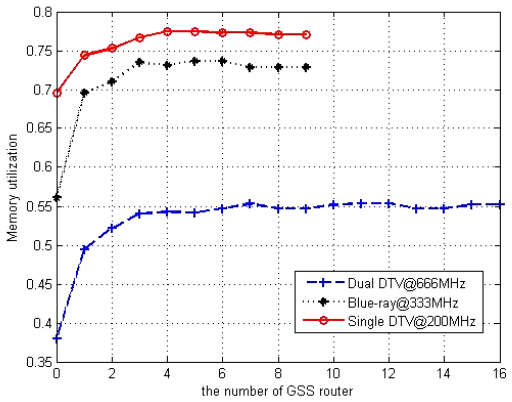
Furthermore, GSS+SAGM further improves the memory performance since it accesses few SDRAM data unnecessary. GSS+SAGM achieves, on average, 4.7% higher memory utilization, 10.2% shorter memory latency of all packets and 9.1% shorter memory latency of priority packets than GSS. Consequently, GSS+SAGM improves, on average, not only 32.7% latency of priority packets but also 3.4% memory utilization and 7.8% latency of all packets, compared to SANoC. Compared to SANoC+PFS, GSS+SAGM improves, on average, 12.7% memory utilization, 25.2% latency of all packets and 15.2% latency of priority packets.

Application	Clock speed	Memory utilization				Memory latency of all packets (cycle)				Memory latency of demand packet(cycle)			
		CONV +PFS	SANoC + PFS	GSS	GSS+ SAGM	CONV +PFS	SANoC + PFS	GSS	GSS+ SAGM	CONV +PFS	SANoC + PFS	GSS	GSS+ SAGM
Blu-ray	133MHz[a]	0.729	0.742	0.77	0.774	141	106	77	72	97	59	42	38
	266MHz[b]	0.612	0.621	0.699	0.745	176	134	112	96	123	73	72	60
	533MHz[c]	0.454	0.517	0.561	0.608	248	166	151	138	179	88	98	90
Single DTV	166MHz[a]	0.676	0.699	0.755	0.779	163	124	96	76	105	64	57	41
	333MHz[b]	0.58	0.613	0.684	0.738	192	143	116	107	128	74	72	66
	667MHz[c]	0.387	0.489	0.534	0.559	309	182	158	151	213	94	98	95
Dual DTV	200MHz[a]	0.655	0.675	0.7	0.709	183	124	103	80	131	62	55	36
	400MHz[b]	0.521	0.577	0.608	0.657	280	178	153	127	156	81	78	68
	800MHz[c]	0.405	0.481	0.518	0.53	389	252	210	207	198	104	101	99
Average		0.558	0.602	0.648	0.678	231	157	131	117	148	78	75	66
Ratio[d]		0.85	0.917	0.987	1.034	1.821	1.233	1.029	0.922	1.508	0.793	0.763	0.672

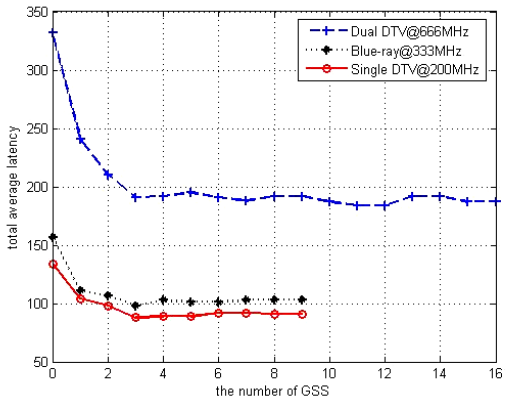
[a] DDR I SDRAM
[b] DDR II SDRAM
[c] DDR III SDRAM
[d] Ratio is based on SANoC

Table 2.7: Memory performance comparison on industrial benchmarks with priority memory requests.

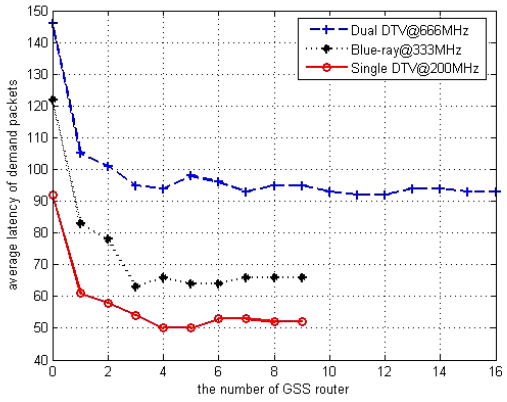
Figure 2.18 shows the memory performance of our application- aware NoC design according to the number of GSS routers when a single DTV model (3×3), a Blue-ray model (3×3) and a dual DTV model (4×4) work with DDR I SDRAM at 200MHz, DDR II SDRAM at 333MHz and DDR III SDRAM at 666MHz, respectively. In the



(a) Average memory utilization



(b) Average latency for all packets



(c) Average latency for demand packets

Figure 2.18: The memory performance of our application-aware NoC design according to the number of GSS routers, where our NoC design achieves the best tradeoff between performance and cost when three conventional routers are replaced to GSS routers.

conventional NoC design, a router employing a priority-first and round-robin flow control algorithm is gradually replaced with our GSS router in the order where a router that is the closest to a memory subsystem is replaced first and where a router that is the farthest away from a memory subsystem is replaced last.

When any input buffer and any memory scheduler are not adopted in a memory subsystem and the conventional router is placed on a network, its memory utilization is just 69%, 56% and 38% in a single DTV model, a Blue-ray model, and a dual DTV model, respectively as shown in Figure 2.18(a). However, whenever our GSS router is substituted for the conventional router, its memory utilization improves rapidly. As a result, when three GSS routers are substituted for three conventional routers, the memory utilization increases up to 77%, 73% and 54% in a single DTV model, a Blue-ray model and a dual DTV model, respectively. However, more than four GSS routers achieve little improvement of memory utilization since the solvable bank conflict and data contention are almost prevented by three GSS routers.

Figure 2.18(b) shows the memory latency of all packets including both a priority packet and a best-effort packet. The memory latencies of all packets are initially 134 cycles, 157 cycles and 332 cycles in a single DTV model, a Blue-ray model and a dual DTV model, respectively. However, whenever the GSS router is substituted for the conventional router, the memory latency of all packets also improves rapidly. As a result, when three GSS routers are substituted for three conventional routers, the memory latency of all packets decreases up to 88 cycles, 98 cycles and 191 cycles in a single DTV model, a Blue-ray model and a dual DTV model, respectively.

Figure 2.18(c) shows the memory latency of priority packets. The memory latencies of priority packets are 92 cycles, 122 cycles and 146 cycles in a single DTV model, a Blue-ray model and a dual DTV model, respectively when any input buffer and

memory scheduler are not adopted in a memory subsystem and the conventional router is placed on a network. However, when three GSS routers are substituted for three conventional routers, the memory latency of priority packets decreases up to 54 cycles, 63 cycles and 95 cycles in a single DTV model, a Blue-ray model and a dual DTV model, respectively. Therefore, three GSS routers placed around a memory subsystem shows the most efficient result in terms of hardware cost and memory performance.

We also evaluate the improvement of memory performance when a short turn-around bank interleaving problem is considered in our application-aware NoC design, called GSS+SAGM+STI. For this experiment, we use three GSS routers employing Figure 2.14(b) and execute a Blue-ray model, a single DTV model and a dual DTV model with DDR III SDRAM at 533MHz, 667MHz, and 800MHz, respectively. The short turn-around bank interleaving problem is not critical in DDR SDRAM working at a low clock frequency. This is because a bank can be sufficiently deactivated and reactivated while any different bank is accessed. On the contrary, the short turn-around interleaving problem causes memory performance to be critically degraded in DDR SDRAM working at a high clock frequency. This is because the deactivation, activation, and WL/CL delay time are too long, compared to the length of data accessed. Table 2.8 shows that GSS+SAGM+STI achieves, on average, 9.4% higher memory utilization, 11.2% shorter memory latency of all packets and 12.9% shorter memory latency of priority packets than GSS+SAGM.

CONV, SANoC, and the proposed NoC design are synthesized by Synopsys Design Vision with OSU PDK 45nm CMOS standard cell library [107]. Table 2.9 shows their gate count in case that they are optimized at 400MHz clock speed. Our flow controller is 8.9% smaller than SANoC even if it provides effective QoS and high throughput. This is because our GSS flow control mechanism for scheduling memory

App.	Clock speed	Utiliz ation	Imp.	Latency of all packets	Imp.	Latency of priority packet	Imp.
Blue-ray	533MHz	0.674	10.9%	119 cycles	4%	79 cycles	12.2%
Single DTV	667MHz	0.590	5.5%	140 cycles	7.3%	87 cycles	8.4%
Dual DTV	800MHz	0.593	11.9%	161 cycles	22.2%	81 cycles	18.2%
Average		0.619	9.4%	140 cycles	11.2%	82 cycles	12.9%

Table 2.8: The memory performance comparison of GSS+SAGM+STI and GSS+SAGM on industrial benchmarks.

Module	CONV		SANoC		GSS+SAGM+STI	
	Gate count	Ratio	Gate count	Ratio	Gate count	Ratio
Flow controller	3,310	0.539	6,732	1.097	6,136	1
Router	56,683	0.904	62,949	1.003	62,721	1
Memory subsystem	489,898	3.283	158,874	1.065	149,245	1
3x3 NoC with memory subsystem	966,250	1.511	661,645	1.035	639,481	1

Table 2.9: The comparison of gate count synthesized at 400MHz clock speed.

requests and avoiding starvation are optimized by event driven architecture. On the contrary, the gate count of our flow controller is 85.4% greater than that of a conventional flow controller due to the additional GSS flow control mechanisms. However, since the flow controllers are commonly tiny, the gate count increased or decreased by our GSS flow control mechanism has little impact on the area of whole NoC design. In addition, routers can be equipped with the minimum GSS flow controllers according to a routing policy. That is, any conventional flow controller through which a packet goes to a memory subsystem can be just substituted for the proposed flow controllers. Moreover, the GSS flow controllers can have fewer input ports. For example, if a memory subsystem is placed in the upper left corner on NoC as shown in Figure 2.17, a router located in (2, 2) can have two 3-input GSS flow controllers. The GSS flow controllers schedule memory requests from processing element, south, and east inputs

and are attached to a north and west output scheduler, respectively. As a result, the gate count of our router is just 10.7% greater than a conventional router and 0.4% less than SANoC, as shown in Table 2.9.

Our memory subsystem has great impact on the area of whole NoC design since it does not require any reordering buffers and any complex memory scheduler. Since memory requests are already scheduled by multiple routers with GSS flow controllers, the memory requests arrive at our memory subsystem with the order friendly with memory operations. In addition, our memory controller has fewer PRE buffers than a conventional memory controller and SANoC due to effective AP operations. Thus, our memory subsystem is 69.5% and 6.1% smaller than a conventional memory subsystem and SANoC, respectively. Such a distinguished gate count decrease by removing reordering buffers and a memory scheduler in our memory subsystem far exceeds a gate count increase by GSS flow controllers in multiple routers. As a result, NoC with our memory subsystem and three routers with GSS flow controllers is 33.8% and 3.3% smaller than a conventional one and SANoC, respectively, as shown in Table 2.9.

We compute their power consumption by Synopsys Prime Time PX after gate-level simulation. As shown in Table 2.10, our application aware NoC design consumes on average 28.5% and 0.3% less power than the conventional one design and SANoC, respectively.

Application	Clock speed	CONV		SANoC		GSS+SAGM+STI	
		Power	Ratio	Power	Ratio	Power	Ratio
Single DTV	200MHz	179.0mW	1.550	116.0mW	1.004	115.5mW	1
Blue-ray	400MHz	351.6mW	1.550	227.8mW	1.004	226.8mW	1
Dual DTV	800MHz	961.9mW	1.328	726.0mW	1.003	724.1mW	1
Average		497.5mW	1.399	356.6mW	1.003	355.5mW	1

Table 2.10: The comparison of power consumption ruing at 400MHz clock speed.

2.2.4 Summary

In NoC, a microprocessor and a specific core that perform various applications request not only a best-effort memory service but also a priority memory service. In addition, they request memory data with various sizes which do not match an SDRAM access granularity. Therefore, we proposed an application-aware NoC design for an efficient SDRAM access. The proposed GSS router schedules a priority packet as fast as possible with the consideration of bank conflict, data contention, and short turn-around bank interleaving which all make memory performance severely degraded. Furthermore, the proposed SAGM NoC design splits a packet to several short packets, based on the BL of SDRAM and then serves them with a partially open-page mode and an AP operation in our memory subsystem. Experimental results showed our application-aware NoC design improved not only the memory utilization and latency of all packets but also the memory latency of priority packets in famous industrial multimedia systems. In conclusion, our application-aware NoC provides more opportunity for bandwidth-hungry system-on-chip designs with the high quality of a memory service.

Chapter 3

Power Optimization for Advanced NoC

Power has become a major concern in NoC as more and more IPs are integrated to a single chip. NoC itself is not efficient for power consumption and even may consume higher power than shard bus interconnects due to increased communication between IPs. However, the combination of NoC and other techniques efficient for power has the potential to easily reduce power to allowable levels.

In NoC, it is not necessary for all IPs and links to run at a single voltage level and clock speed. Voltage-frequency island (VFI) enables fine-grained core-level power optimization by utilizing a unique voltage and clock for each island. Thus, VFI can be one of the most desirable solutions for reducing power consumption in NoC. This is possible because static complementary metal–oxide–semiconductor (CMOS) logic used in the vast majority of current processors has a voltage-dependent maximum operating frequency. Thus, when used at a reduced frequency, the processors can operate at a lower supply/higher threshold voltage. The power is supplied by an off- or on-chip source and can be controlled independently for each VFI. This may be achieved by using either on-chip voltage regulators or multiple power grids. The communication across different VFIs is achieved through a mixed clock first input, first output (MCFIFO) buffer and a voltage level converter (VLC) [12][16]. They provide the flexibility to scale the frequency and voltage of various VFIs in order to minimize power consumption. A number of modern processors such as Intel's XScale [47], AMD's Athlon [77], and IBM's CU-08, -45HP and -65HP [45] are employed with the VFI concept. The use of multiple voltages and clocks in NoC provides better performance-power tradeoffs than that of a single voltage and clock. In Section 3.1, we present a systematic VFI-aware

energy optimization framework that performs partitioning, mapping and routing together to improve the power efficiency of VFI-based NoC designs [49][53].

Application mapping that decides how to topologically place the selected set of cores onto the tiles of a network can greatly reduce both application latency and power consumption. NoC designers or programmers commonly favor a regular mesh network consisting of regular rectangle tiles on which homogeneous processors are placed since the regular mesh network makes application mapping manageable easily. On the contrary, most industrial SoC platforms consist of heterogeneous cores with different design areas, thus they may be structured with an irregular mesh network or even a custom network. Therefore, since previous works have just optimized their application mapping on the regular mesh architecture network, a novel application mapping algorithm is required to reduce application latency and power consumption in various networks. In Section 3.2, we present architecture-aware analytic application mapping (A3MAP) algorithms that are analogous to analytical communication minimization in various network architectures [51].

3.1 VFI-AWARE ENERGY OPTIMIZATION FRAMEWORK FOR NOC

There are many existing works that address the problem of VFIs generation for core-based SoC designs. The design style based on multiple VFIs was proposed in [62], where synchronous IPs in an SoC design had different voltages and frequencies. Hu et al. considered voltage island partitioning, assignment and floorplanning in an SoC design [41]. By using a graph-based representation, the partitioning and floorplanning steps were modeled in an integrated fashion and solved by a simulated annealing-based algorithm. Wu et al. considered trade-off between power and design cost under timing requirement

for a VFI generation problem that was formulated as a voltage-partitioning problem and solved by a two-step heuristic algorithm [118]. In [119], the number of voltage islands determined by island partitioning was minimized after performing placement phase. Ching et al. considered non-slicing voltage-island partitioning to facilitate the floorplanning in [14].

As many cores have been recently interconnected by an on-chip network, the concept of the VFI design is being employed in NoC. Ogras et al. proposed a design methodology for partitioning NoC tiles into multiple VFIs and assigning supply/threshold voltages and corresponding clock speeds to each domain [84]. Leung et al. proposed an NoC design with voltage islands in [66]. The approach simultaneously solved three problems, i.e. tile mapping, routing path allocation and physical voltage island generation and voltage assignment. Seiculescu et al. proposed a synthesis approach to obtain customized application-specific NoC that can support the shutdown of voltage islands in [102]. Liu et al. proposed a simultaneous task and voltage scheduling algorithm for energy minimization in NoC based designs in [69]. The energy-latency tradeoff was handled by Lagrangian relaxation.

Such a VFI-based NoC concept fits very well with a globally asynchronous, locally synchronous (GALS) design style for global on-chip asynchronous communication. The problem of selecting voltages and clock speeds for voltage/frequency islands in GALS systems was addressed in [81]. The problem of both rate and latency constrained systems was considered and a practical solution for static and application adaptive, dynamic voltage and speed scaling is provided. The field programmable gate array (FPGA) prototype of GALS-based NoC with two synchronous IPs was presented in [95]. In [8], a method for reducing wire propagation delays in

GALS-based NoC is proposed. In [84], both VFI and GALS concepts were applied to an NoC design for minimum energy consumption.

However, there are several limitations in the previous VFI-based NoC designs although their powerful energy efficiency. First, an island partitioning process is only combined with a voltage and frequency (VF) assignment process. Such approach limits the flexibility of VFI optimization thus NoC energy efficiency. As shown in the experimental results of [84], more than three VFIs in NoC with less than 25 cores cannot improve overall energy consumption any more since the inflexibility of VFI optimization generates a lot of energy overheads. Second, a search for low energy consumption is carried out on a hard mesh network where both communication and computation components are pre-designed. Since the application mapping process is not optimized by a VFI-aware manner, the solution space is inevitably constrained. Third, VFI-based NoC needs a good routing strategy to bring down energy consumption. It may be inefficient to insert all links between different VFIs since MCFIFO and VLC required to interconnect different VFIs are too expensive. Therefore, pruning the links between VFIs and allocating efficient routing paths over the survived links are required to further improve VFI energy efficiency, where the routing path must guarantee deadlock and livelock freeness. Last, efficient VFI interfaces are required to easily satisfy the performance constraints. In [84], MCFIFO and VLC are simply placed between routers in different VFIs. However, if any packet generated in VFI operating with a fast clock passes VFI operating with a slow clock, it may be difficult for the packet to satisfy performance constraints. In addition, the better VFI interfaces can further reduce the number of MCFIFO and VLC required thus NoC energy consumption.

3.1.1 Motivation and Contributions

3.1.1.1 Motivational Example

VFI generation causes the chip design process severely to be complicated with respect to static timing, power routing and clock tree. In particular, the design complexity grows significantly with the number of allowed VFIs as shown in Figure 3.1. Since each VFI requires its own power grid, clock tree, MCFIFO buffer and VLC in order to communicate with other VFIs, those design overheads with respect to area, delay and energy are not avoidable. Therefore, NoC designs employing the concept of VFI design needs to cluster as many cores supplied by the same voltage level and clock speed as possible and ensure that the created grouping does not violate other design constraints such as performance, timing and wiring congestion [93].

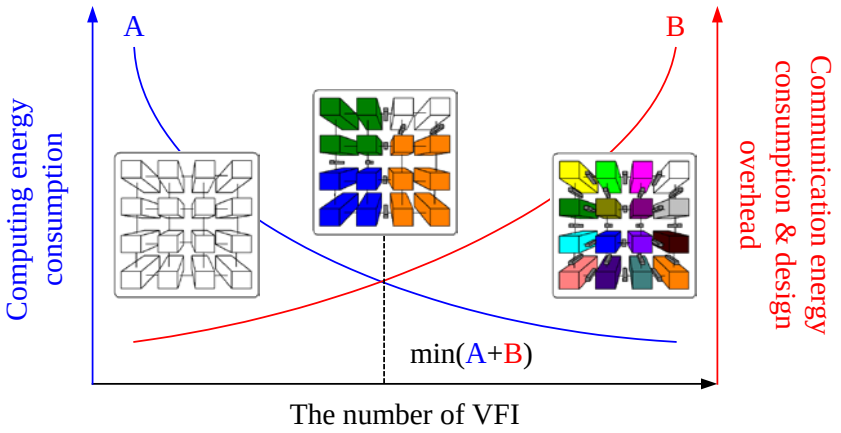


Figure 3.1: Computing and communication energy consumption and design overhead according to the number of VFIs. The goal of VFI based NoC designs is to minimize the sum of the computing and communication energy and the design overhead.

Figure 3.2, for instance, shows two VFI-based NoC designs with 16 tiles. Each tile operates at either voltage *A* or voltage *B*, depending on the computation complexity

of mapped cores. After cores are mapped to tiles for the purpose of reducing hop counts and thus communication energy consumption, two different mapping results are shown in Figure 3.2(a) and (b). Then, let us apply tile partitioning with VF assignment to the NoC designs, as proposed in [84]. Such approach may improve total energy consumption by running two VFIs in Figure 3.2(a) since its additional design cost is only four complex routers including MCFIFO and VLC. If the energy saved by operating two VFIs is lower than the energy consumed by four complex routers, it is regarded as a desirable solution. However, in Figure 3.2(b), any tile cannot operate together with other tiles at the same VF. Operating each tile as one VFI needs the complex wiring of power, ground and clock and even 24 complex routers that may be much more expensive than the energy saved by VFI separation. As a result, higher voltage of two voltages, *A* and *B*, will be used for meeting performance constraints in overall NoC such that their approach may fail to consume lower energy. This shows that tile partitioning with VF assignment alone may be misleading during NoC energy optimization. Our solution is to combine core partitioning with VF assignment, core mapping and routing path allocation together, which are considered by VFI-aware manner.

In addition, we implement various VFI interfaces and propose their insertion algorithm to minimize communication latency and further reduce energy consumption. If any packet passes VFI operating at very low clock frequency, it may be difficult for the packet to meet target communication performance. Even though a core operates at the same clock frequency as its VFI, its router and link should operate at different clock frequency satisfying performance constraints. To achieve this issue, we perform the VFI-aware mapping algorithm with specific constraints and insert a pair of MCFIFO and VLC between routers or a router and a core by our VFI interface insertion algorithm.

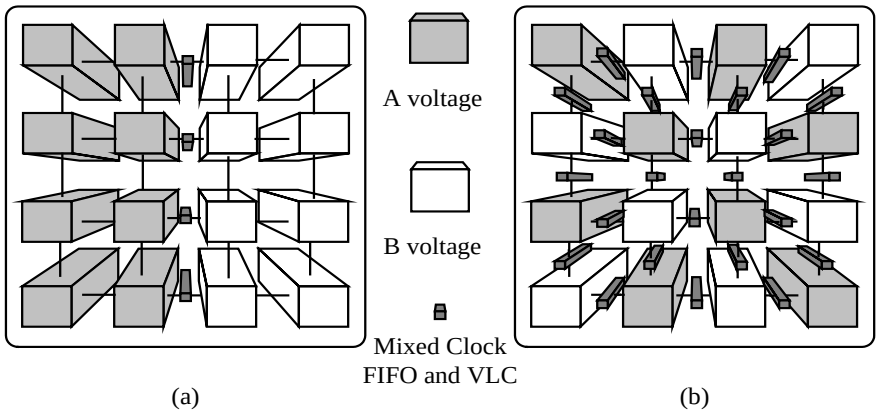


Figure 3.2: Motivational VFI based NoC designs.

3.1.1.2 Major Novelty

The main novelties and contributions of our VFI-aware optimization framework include:

- We propose an NoC design methodology that is aware of a voltage-frequency island. After partitioning cores with VF assignment, mapping the cores and allocating routing path provides more opportunities to efficiently build unified VFIs.
- VFI-aware mapping is performed, based on an effective region growing method. In addition, we add specific constraints for efficient VFI interface which provides short communication latency. Such VFI-aware mapping techniques fit the VFI-based NoC methodology well.
- VFI-aware routing path allocation seeks to further reduce VFI overheads such as MCFIFO and VLC.

- We implement various VFI interfaces and propose their insertion algorithm. Such approach achieves short communication latency and further reduces VFI overheads.
- We show that the proposed VFI-based NoC optimization framework makes cores running at the same voltage and clock frequency unified to single VFI with the slight increase of hop count such that it provides better energy-performance trade-offs.

3.1.2 Problem Formulations

We start to solve VFI-applied NoC issues from a core graph consisting of cores and their communication relation since a core can be one-to-one mapped onto a tile of NoC. Therefore, we assume to have an application that needs to be mapped onto SoC populated by cores as a starting point. We implement earliest deadline first (EDF) and a heuristic called energy aware scheduling (EAS) that are used for generating a core graph from a task graph [42].

3.1.2.1 Partitioning with VF Assignment Problem

In this stage, the object is to decide how cores are partitioned to minimize energy consumption except for communication energy consumption. We assume that the maximum number of allowable VFIs denoted by $max\{n(VFI)\}$ or m, a core graph G with a set of n cores where pairs of supply voltage and threshold voltage are (V_1, V_{t1}), (V_2, V_{t2}), …, (V_n, V_{tn}) and an NoC topology such as a mesh or a torus are given. Clock period (τ_i) for each core c_i, which can trade off with supply and threshold voltage, is defined as:

$$\tau_i\left(V_i, V_{ti}\right) = \frac{K_i V_i}{\left(V_i - V_{ti}\right)^{\alpha}} \tag{3.1}$$

where α is a technology parameter and K_i is a design specific constant [84][73][98]. The operating frequency (f_j) of VFI j is determined by a core including the longest path as:

$$f_j \le \min_{i \in S_j}\left\{\frac{1}{\tau_i\left(V_i, V_{ti}\right)}\right\} \tag{3.2}$$

where S_j is a set of tiles that belong to VFI j. Each core can be performed with a different supply and threshold voltage and the voltage level is regarded as a legal one as long as the performance constraints are satisfied. Based on these constraints, we partition n cores into the maximum number of allowable VFIs and assign a supply and threshold voltage to each core such that total energy consumption is minimized as:

$$\min\left\{\sum_{\forall i \in G}\left(R_i C_i V_i^2 + T_i k_i V_i \exp\left(-\frac{V_t}{S_t}\right)\right)\right\} \tag{3.3}$$

where G is a set of n cores, R_i is the number of active cycles, C_i is total switched capacitance per cycle, T_i is the number of idle cycles, k_i is a design parameter and S_t is a technology parameter [7].

3.1.2.2 VFI-Aware Mapping Problem

In this section, we determine which tile each core should be mapped to in order to minimize communication energy consumption under stringent performance constraints.

> *Definition 1*: A partitioned core graph *G´(V,E)* generated by Section 3.1.2.1 is a directed graph, where each vertex $v_i \in V$ represents a core, and each directed edge $e_{i,j} \in E$ represents communication relation from v_i to v_j. *vol($e_{i,j}$)* represents the communication volume between v_i to v_j.

Definition 2: An NoC topology graph *N(T,C)* is a directed graph, where each vertex $t_i \in T$ represents a tile, and each directed edge $c_{i,j} \in C$ represents candidate minimum paths from t_i to t_j. $bw(c_{i,j})$ represents the minimum bandwidth requirement from t_i to t_j.

The one-to-one mapping function *M()* of the partitioned core graph *G´(V,E)* onto the NoC topology graph *N(T,C)* is defined as:

$$M : V \to T,\ \ s.t. M\left(v_i\right) = t_j, \forall v_i \in V,\ \ \exists t_j \in T \tag{3.4}$$

This mapping function is only defined when $n(V) \leq n(T)$, where *n(X)* is the number of $x_i \in X$. In addition, our mapping function has two objectives, i.e. minimizing overall communication and building a convex region with cores using the same voltage on given NoC.

3.1.2.3 VFI-Aware Routing Problem

$E_{bit}(e_{i,j})$ is the energy consumption of sending one bit of data from $M(v_i)$ to $M(v_j)$. Assuming the bit energy values are observed at *VDD*, its energy consumption is defined as:

$$E_{bit}\left(e_{i,j}\right) = \sum\nolimits_{p \in L\left(e_{i,j}\right)} \left(E_{Lbit}\left(p\right) + E_{Bbit}\left(p\right) + E_{Sbit}\left(p\right)\right) \frac{V_i^2}{V_{DD}^2} \tag{3.5}$$

where $L(e_{i,j})$ is a set of links passed from $M(v_i)$ to $M(v_j)$ and E_{Lbit}, E_{Lbit}, and E_{Lbit} are the energy consumed by the link, buffer and switch fabric, respectively [84]. Therefore, finding a routing path from t_i to t_j is formulated as:

$$\min\left[\sum_{\forall e_{i,j}}\left\{vol\left(e_{i,j}\right)E_{bit}\left(e_{i,j}\right)\right\}+\sum_{q\le m}\left\{E_{CLK}(q)+E_{VLC}(q)+E_{MCFIFO}(q)\right\}\right] \quad (3.6)$$

where E_{CLK} is the energy overhead of generating additional clock signals and E_{VLC} and E_{MCFIFO} is the energy overhead of VLC and MCFIFO, respectively. This formulation is subject to performance constraints expressed as:

$$\frac{e_c}{f_c}+d_c \le deadline \quad (3.7)$$

where e_c is the number of cycles required to complete the function of core c and d_c is communication delay encountered when core c needs to communicate with a core mapped to a different tile.

3.1.3 VFI Optimization Framework

In this section, we present the proposed VFI-aware NoC methodology and detailed algorithms for core partitioning with VF assignment, VFI-aware core mapping onto a given NoC topology and VFI-aware routing path allocation. In addition, we show a VFI interface to easily satisfy performance constraints and further improve its energy efficiency. Figure 3.3 shows the overall flow chart of our VFI-aware NoC optimization framework. We first partition *n* cores but not tiles into *m* VFIs, where *m* is given as the maximum number of allowable VFIs. Based on the result of partitioning cores, novel VFI-aware mapping and routing path allocation algorithms are performed to minimize communication energy consumption. Then, we establish unique interconnection for key traffic paths between islands to minimize the overheads of VFI. After routing path allocation is carried out, the pairs of MIFIFO and VLC are placed between routers or a core and its router. Finally, we compute its energy consumption and check whether it

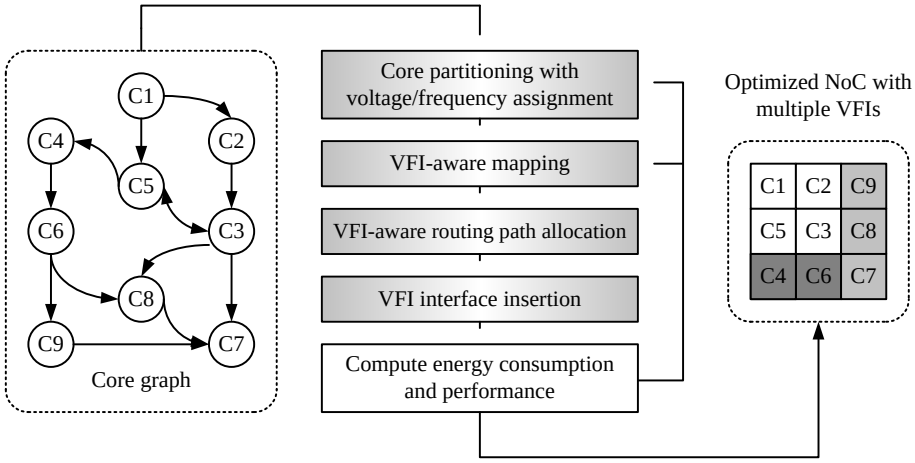


Figure 3.3: The proposed VFI-aware NoC methodology where VFI partitioning is first performed.

meets performance constraints. If the performance constraints are satisfied, an energy-efficient NoC platform with $q(\leq m)$ VFIs is obtained. Otherwise, we again perform the VFI-aware mapping with constraints descripted in Section 3.1.3.2 or all procedures with decreasing the maximum number of VFIs, m by 1.

3.1.3.1 Core Partitioning with VF Assignment

The proposed core partitioning algorithm is different from [84] that can partition tiles in a neighbor on NoC. Since our partitioning stage is performed before the stages of mapping and routing path allocation, any core unmapped to a tile can be clustered together to the same VFI. Thus, our methodology can make cores operating at the same voltage gathered as one VFI such that the number of a complex router requiring MCFIFO and VLC is minimized. Consequently, it further reduces energy consumption.

Algorithm 4 shows our core partitioning algorithm for a core graph $G(V,E)$ and the maximum number of allowable VFIs, m. Since the voltage of a core can trade off its

Algorithm 4 Core partitioning and VF assignment

Input: ***G(V,E), max{n(VFI)}=m***

1: Compute the lowest voltage of each core satisfied with performance constraints using Eq. (3.1);
2: **for** all cases that choose *m* VFs among *k* VFs used in each core **do**
3: Assign the lowest operable voltage among *m* to all *n* cores;
4: **if** chosen *m* VFs are satisfied with performance constraints of all *n* cores **then**
5: Compute overall energy consumption from Eq. (3.3);
6: **end if**
7: **end for**
8: Choose the best VF pair consuming minimum energy;

Output: ***G´ (V,E)*** **partitioned into VFI**

operating frequency, the lowest voltage of each core are computed from Eq. (3.1) in line 1, which must satisfy performance constraint of each core. If there are *k* VFs used by *n* cores and the maximum number of accepted VFIs is *m*, we can choose *m* VF among total *k* VF (where $m \leq k$), where there are total ${}_kC_m$ cases. Then, the lowest VF among the chosen *m* VFs is assigned to each core if the performance constraint of the core is satisfied in the VF level. When the chosen *m* VFs are satisfied with the performance of all cores, computation energy consumption is computed from Eq. (3.3). This procedure repeats all ${}_kC_m$ VF cases. After completing this procedure, we choose the best VF pair consuming the lowest energy.

For example, there are 4 cores and the lowest voltages of each core, which satisfy their performance constraints, are 1.0V, 1.2V, 1.1V and 1.0V, respectively (k=3). We assume that the maximum number of allowable VFI is 2 (m=2) and the operating frequency and area of all cores are same to make simple. In this example, we can choose 2 of 3 voltages, i.e. 1.0V, 1.1V and 1.2V. Thus, there are 3 cases (${}_3C_2$) we can choose: (1.0, 1.1), (1.0, 1.2) and (1.1, 1.2). Here, (1.0, 1.1) case cannot satisfy the performance constraint of the second core that must operate at least 1.2V. On the other hand, the rest of two pairs meet performance constraints since the cores can run at 1.0V, 1.2V, 1.2V and 1.0V in the second case and 1.1V, 1.1V, 1.2V and 1.1V in the last case.

Consequently, since the third pair consumes more energy than the second pair, (1.0, 1.2) case is chosen to compose two VFIs and assigned to all cores such that cores runs at 1.0V, 1.2V, 1.2V and 1.0V. Finally, a core graph *G´ (V,E)* of which the VF level is assigned is generated.

3.1.3.2 VFI-Aware Mapping Algorithm

Cores operating at the same VF level should be mapped to NoC tiles which build a convex region and thus, it reduces the overhead energy consumption caused by MCFIFO, VLC and clock/power routing. We already know which cores VFIs consist of because the core partitioning with VF assignment is performed in the previous section. Therefore, this information helps selecting a region which looks as convex as possible.

In the mapping step, we use a heuristic approach using the partitioned core graph, as shown in Algorithm 5. In line 1, cores are sorted in a decreasing order by the amount of their communication and then they are mapped in the order. We define a *VF_LIST()* indicating whether the VF level of a core being mapped is already used on NoC. From line 3 to 11, our initial mapping algorithm starts for the sorted v_i. In line 4, the proposed mapping algorithm checks whether the VF level of a core being mapped is used throughout *VF_LIST()*. If the VF level of the core being mapped does not exist in *VF_LIST()*, it is mapped on any empty NoC tile with the maximum neighbor tiles or the minimum hops (line 5). Then, the VF level of the core is recorded in *VF_LIST()* (line 6). If the VF level of the core being mapped exists in *VF_LIST()*, the core is mapped on any candidate tile with the same VF level (line 8). Next, additional candidates are selected in line 10, where *NSWE(t_i)* indicates north, south, west and east tile of the mapped t_i. The candidates are used for the next mapped cores that run at the VF level. If the core

Algorithm 5 VFI-Aware Mapping

Input: *G´(V,E)*, NoC topology

1: *Sort(vol(v_i))* in decreasing order;
2: *VF_LIST() = empty*;
3: **for** all sorted v_i **do** // initial mapping
4: **if** VF level of v_i does not exists in *VF_LIST()* **then**
5: *M(v_i)* on any empty tile t_j with maximum neighbors and minimum hops;
6: Add VF level into *VF_LIST()*;
7: **else then**
8: *M(v_i)* on any candidate tile t_j with minimum hops;
9: **end if**
10: Add unmapped *NSWE(t_j)* (or *NS(t_j)* or *WE(t_j)*) as candidate with VF of v_i;
11: **end for**
12: **for** all isolated island t_i **do** // moving of an isolated tile
13: *Pair-wise swapping(t_i,t_j)* to be clustered to main VFI using the same VF level under minimum traffic increase;
14: **end for**
15: **for** all t_i **do** // minimization of the overall traffics
16: *Pair-wise swapping(t_i,t_j)* within island for min. traffic;
17: **end for**

Output: *N(T,C)* mapped on NoC

mapping is again performed due to the dissatisfaction of performance constraints, either *NS(t_i)* or *WE(t_i)* is used as candidate tiles for cores working at the lowest VF level. This candidate constraint makes the cores mapped to tiles in a single row or column and thus, improves communication speed with our VFI interface insertion algorithm which is described in Section V.D. If it does not still satisfy the performance constraints, either *NS(t_i)* or *WE(t_i)* is also used as candidate tiles for cores running at the next lowest VF level. This procedure repeats until all cores are mapped on NoC.

Our initial mapping algorithm as a near convex region solution reduces the number of an isolated tile separating from the main group of tiles (VFI) running at the same VF level. However, we cannot completely remove an isolated tile around the edge of NoC. In order to combine an isolated tile with its main VFI using the same VF level, the isolated tile is moved to the VFI if the moving cost is less than the overhead of the

extra isolated tiles (line 13). Since our initial mapping does not generate an isolated tile around the center of NoC where communication is too heavy, the loss of performance and the increase of hop count caused by this pair-wise swapping of tiles in different VFIs are minimal. The procedure repeats until all isolated tiles disappear. Finally, the pair-wise swapping of tiles within each island is executed to find the best mapping solution with the minimum hop count until it is not improved (line 16).

Figure 3.4 shows the simple example of the initial mapping in Algorithm 5. In Figure 3.4(a) that is the partitioned core graph, the number is the mapping order by sorting cores into the amount of communication and two groups, i.e. grey and white denoted VFI 1 and VFI 2, respectively exist. Core 1 that has the maximum communication is placed onto the center of NoC with the maximum neighbors as shown in Figure 3.4(b). Four candidates, *a*, *b*, *c* and *d* are also marked as VFI 1 for the next mapped cores using the same VF as the core 1, i.e. core 3 and 4. Core 2 that has the next maximum communication is placed onto candidates minimizing hop count with other cores already mapped if candidates marked as VFI 2 exist. Otherwise, core 2 is only placed onto any unmapped tile that minimizes hop count with other cores already mapped. In this example, core 2 is mapped by the latter case as shown in Figure 3.4(c). Three candidates, *e*, *f* and *g* are also marked as VFI 2 for the next mapped core using the same VF as the core 2, i.e. core 5 and 6. Next, core 3 that has the next maximum communication is placed onto one of the VFI 1 candidates, minimizing hop count with other cores already mapped. In Figure 3.4(c), there are three candidates, *b*, *c* and *d* and candidate *b* is chosen because *b* generates fewer hops than *c* and *d*. Then, three candidates, *e*, *h* and *i* are also marked as VFI 1, where any core operating at VFI 1 and VFI 2 can be mapped to tile *e* in Figure 3.4(d). The procedure repeats until all cores are mapped as shown from Figure 3.4(e) to (f). Since this mapping algorithm makes the

region of each VFI grow toward its candidates, the VFI is prevented splitting into several VFIs using the same VF level.

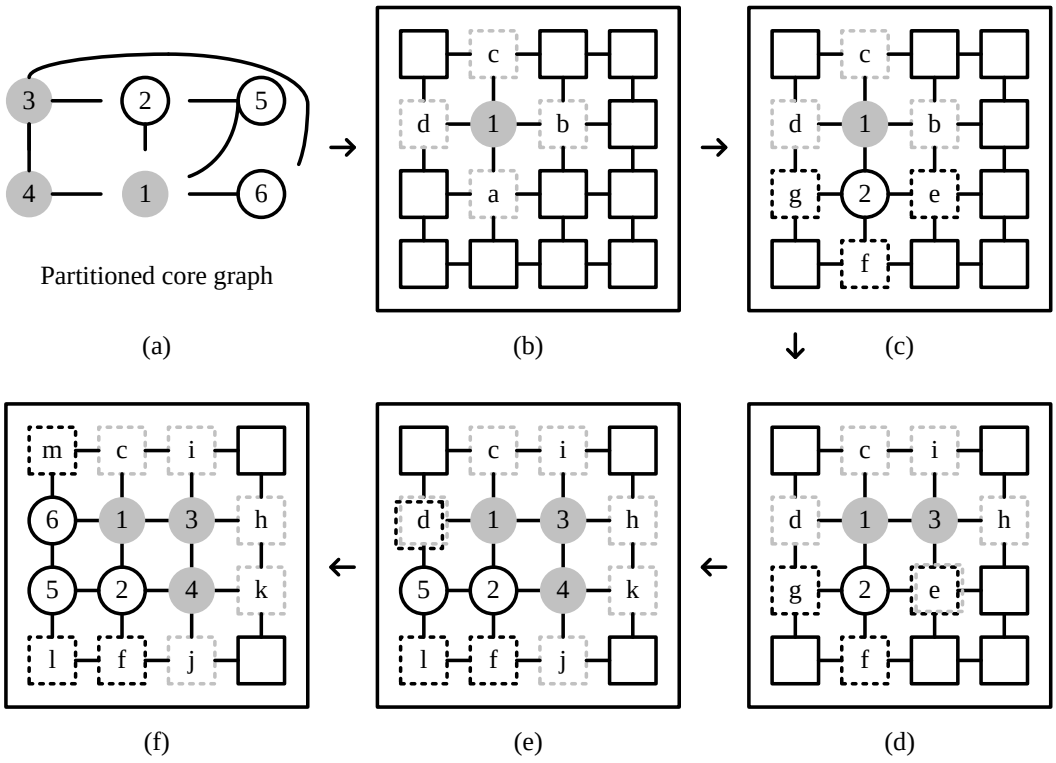


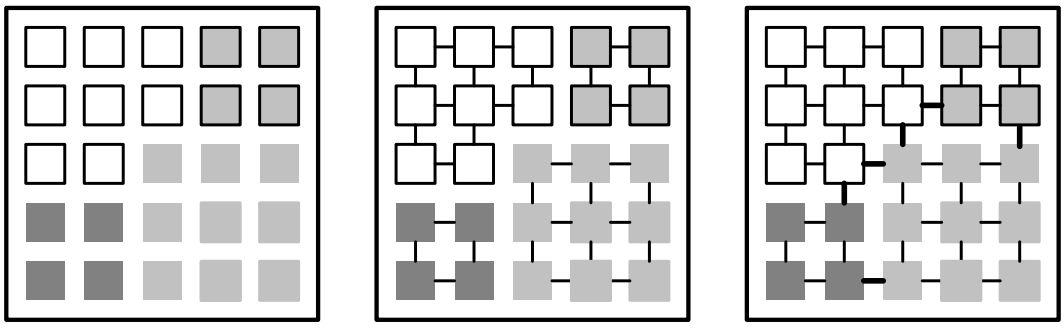
Figure 3.4: Incremental core mapping on NoC.

3.1.3.3 VFI-Aware Routing Path Allocation

In this section, we present a VFI-aware routing path allocation algorithm. In [84], more than three VFIs applied to less than 25 cores could not improve overall energy consumption any more since the VFIs also generate more overheads that degrade the energy efficiency of VFI separation. The key idea of our routing path allocation is to use the minimum links between VFIs. Since NoC based on a mesh or torus topology has a lot

of extra bandwidth, we can remove some links requiring MCFIFO and VLC if the latency of communication and the bandwidth of links satisfy performance constraints. In addition, since our VFI-aware mapping algorithm generates unified VFIs that mean any core is not split from the main group of VFI using the same voltage as the core, we use fewer MCFIFO and VLC. Consequently, the rate of energy saved by VFI separation becomes higher than the rate of energy consumed by MCFIFO and VLC as the number of VFI increases in our VFI-based NoC.

Figure 3.5 shows how links between tiles are inserted briefly. After the VFI-aware mapping in Section V.B, we assume that there is no link between any tiles as shown in Figure 3.5(a) including four VFIs. Next, as shown in Figure 3.5(b), all links within each VFI are inserted. Then, some links between VFIs are partially inserted, as shown in Figure 3.5(c). Under such an irregular NoC interconnection, routing paths allocation should minimize energy consumption and improve performance with livelock and deadlock freeness. As a result, our NoC-aware routing path allocation can achieve lower energy consumption with the tiny loss of performance and the tiny increase of hop count.



(a) No link (b) Inserting links within VFI (c) Inserting links between VFI

Figure 3.5: Link insertion within VFI and between VFIs, where all links between VFIs are not inserted.

In addition, communication congestion that causes the misrouting, dropping and delaying of packets is minimized.

Algorithm 6 minutely shows how links between tiles are inserted and how routing paths are allocated. Algorithm 6 consists of two parts, i.e. inserting links (line 1 to 3) that changes an NoC topology and allocating routing paths on such an irregular interconnection (line 4 to 12). First, we interconnect all tiles within each VFI (line 1) as shown in Figure 3.5(b) because routers required neither MCFIFO nor VLC havereasonable overhead. The optimal number of the complex routers with several MCFIFOs and VLCs for connecting two islands is computed as:

$$b = \left\lceil \frac{w_{i,j} vol(VFI_{i,j})}{bw(VFI_{i,j})} \right\rceil \tag{3.8}$$

Algorithm 6 VFI-Aware Routing Path Allocation

Input: ***N(T,C)***

1: Interconnect all tiles within each VFI;
2: Compute the optimal number of routers with MCFIFO or VLS between two adjacent VFIs from Eq. (3.8);
3: Insert *b* routers to any place between VFIs, where the minimum hops are generated;
4: *Sort(length($c_{i,j}$))* in increasing order; // Rule 1
5: **for** all $c_{i,j}$ **do**
6: Bounding box including source and destination is built;
7: Dijkstra's shortest path algorithm where energy consumption of communication is computed from Eq. (3.6); // Rule 2
8: **if** performance of Eq. (3.7) is not satisfied **then**
9: Increase $w_{i,j}$ of Eq. (3.8) more than 1;
10: Go to line 2;
11: **end if**
12: **end for**

Output: link insertion and deterministic, minimal and livelock/deadlock-free routing path

where $\lceil x \rceil$ is the smallest integer larger than x, $w_{i,j}$ is the weight of links between VFI i and VFI j and $vol(VFI_{i,j})$ and $bw(VFI_{i,j})$ are the total amount of communication volume and the minimum bandwidth requirement between VFI i and VFI j, respectively (line 2). If performance constraints are not satisfied, the weight $w_{i,j}$ increases more than 1, which means more links are inserted between adjacent VFIs. The b complex routers with MCFIFO and VLC are placed between two VFIs, where the minimum hops are generated.

Here is a simple example in Figure 3.6, where S1, S2 and S3 communicate with D1, D2 and D3 respectively. We assume that the amount of each communication is 1Mbit/s, each link between tiles can contain 5Mbit/s and $w_{i,j}$ is 1. Therefore, $vol(VFI_{i,j})$ is 3Mbit/s and $bw(VFI_{i,j})$ is 5Mbit/s such that b is equal to 1 from Eq. (3.8). Then, we can insert one link between two islands. Depending on the location of a link, hop counts computed by the minimum shortest path are 9Mbit/s, 11Mbit/s and 7Mbit/s in Figure 3.6(a), (b) and (c) respectively. Therefore, we insert one link between VFIs as shown in Figure 3.6(c) because it generates the minimum hops.

From now, we perform routing path allocation under such an irregular NoC interconnection. In line 4 of Algorithm 6, all $c_{i,j}$ are sorted in an increasing order by their minimum hop count. For example, in Figure 3.6 (c), S2-to-D2 is the shortest and S1-to-D1 and S3-to-D3 have the same length. Then, we allocate routing paths based on the following two rules.

> *Rule 1*: $c_{i,j}$ with few hops among C is allocated earlier to relieve communication congestion between VFIs.

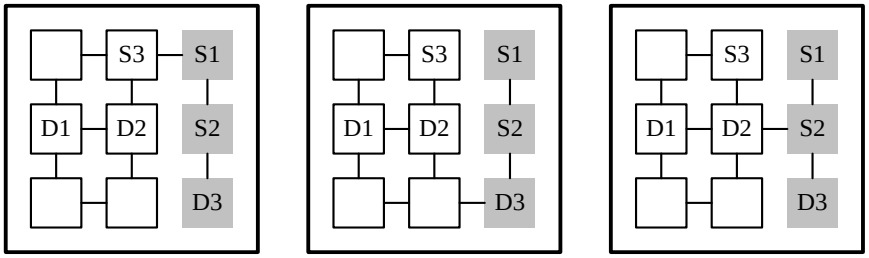
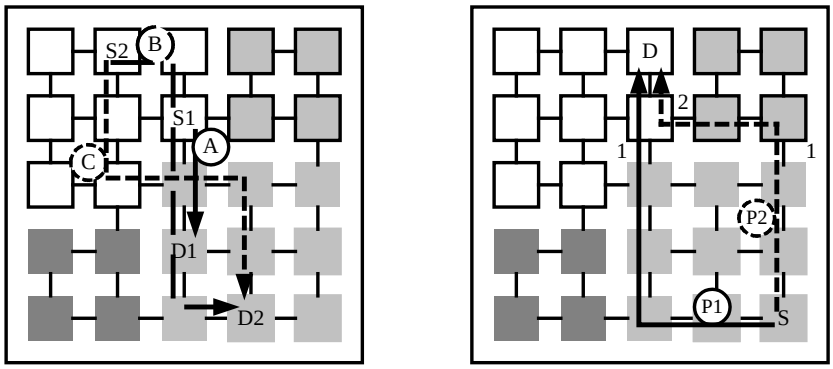


Figure 3.6: Finding the best interconnection between VFIs.

In Figure 3.7(a), there are two packets of which the directions are S1-to-D1 and S2-to-D2 and of which the hop counts are 2 and 6, respectively. From the *Rule 1*, the packet of which the direction is S1-to-D1 is allocated before the packet of which the direction is S2-to-D2 is allocated. The S1-to-D1 packet has only path A as the minimum shortest path whereas the S2-to-D2 packet has two paths, i.e. B and C as the minimum shortest path. If the S2-to-D2 packet is allocated to path B earlier than the S1-to-D1 packet, path A will overlap with path B since the S1-to-D1 packet has no choice. At a result, the congested packets are dropped and misrouted in a bufferless flow control mechanism like circuit switching or have long communication latency in buffered flow control mechanism like packet switching. However, if path A is allocated earlier than the S2-to-D2 packet, the path C but not the path B can be chosen as the routing path of the S2-to-D2 packet. Therefore, the routing path allocation order is important to reduce communication congestion and balance network load between VFIs. For example, in Figure 3.6(c), the S2-to-D2 packet should be allocated earlier than the S1-to-D1 packet or the S3-to-D3 packet according to the rule 1.

Our routing path allocation follows *Rule 2* in the line 6 of Algorithm 6 if the VFI of packet source is different from the VFI of its destination:



(a) Rule 1 (b) Rule 2

Figure 3.7: The proposed rules for allocating routing path in VFI-based NoC.

Rule 2: If the VFI of packet source is different from the VFI of its destination, the minimum shortest routing path that passes through fewer islands is selected.

For example, let any packet move from S to D in Figure 3.7(b). Even if several shortest routing paths can be chosen for the packet, let two routing paths, i.e. P1 and P2 considered. While both P1 and P2 are the minimum shortest paths, the routing path P1 meets one different island and the routing path P2 meets two different islands. As a result, the routing path P1 provides better performance and lower energy consumption than the routing path P2 passing two islands since the routing path P2 needs additional energy overheads.

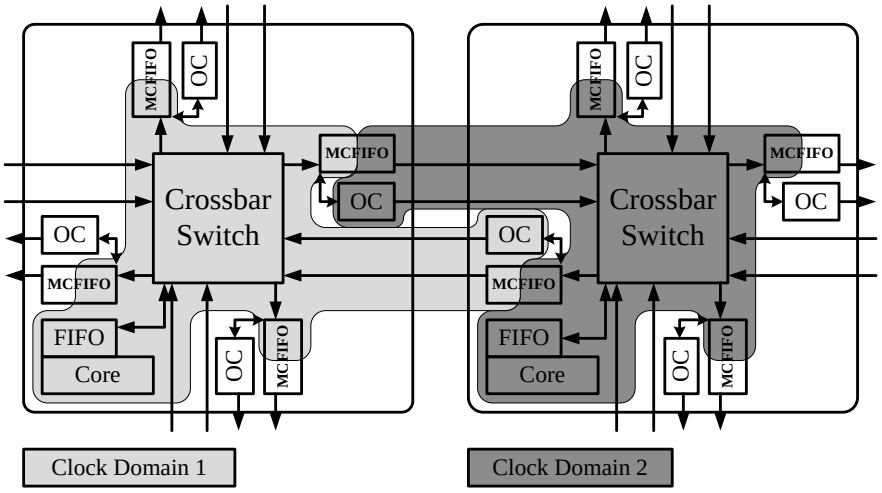
Based on two rules, we perform our routing path allocation algorithm. For each $c_{i,j}$, a bounding box is formed (line 7) and then, the path with the minimum energy consumption is obtained within the bounding box from the Dijkstra's shortest path algorithm, where its energy consumption is computed from Eq. (3.6). Since the routing path is allocated by a deterministic and minimal path router, both livelock and deadlock are free. If performance constrains computed from Eq. (3.7) is not satisfied, we increase a

link weight $w_{i,j}$ between VFIs, go to line 2 and then repeat the routing path allocation. Even if there is a tiny increase of hop count in our routing path allocation due to irregular link insertions between VFIs, the enormous energy saved by VFI separation covers such the tiny penalty.

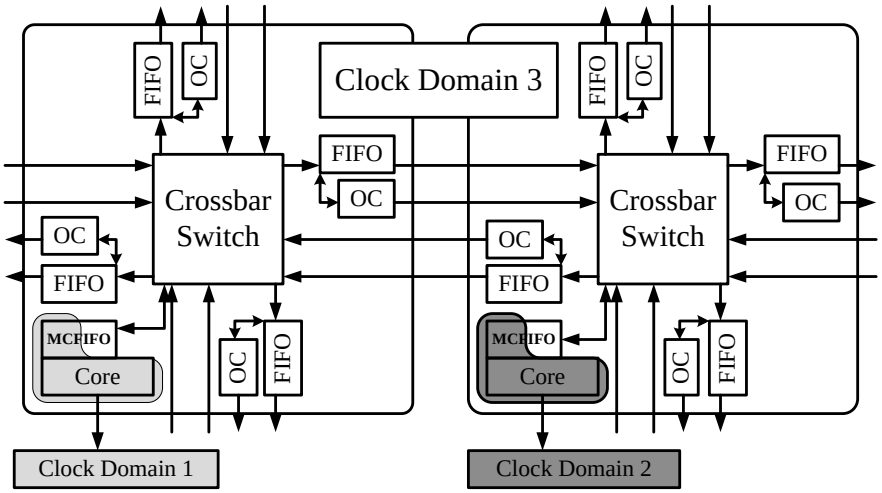
3.1.3.4 VFI-Aware Interface Planning

In a VFI-based NoC design, all data and control signals are required to be converted to a different voltage by VLC and synchronized to a different clock by MCFIFO whenever they pass through a boundary between different VFIs. In the conventional VF conversion proposed in [84], MCFIFO and VLC are simply inserted between routers in different VFIs as shown in Figure 3.8(a). Such a VFI interface may make it difficult to guarantee short communication latency under various VFI-based NoC design scenarios since the speed of on-chip communication depends on clock speeds used in VFIs. For example, let VFI 1 and 2 operate with 1GHz and 100MHz clock, respectively, in Figure 3.9. Then, let any packet generated in S go to D. If the routing path of the packet is allocated to P1 which is one of the shortest path as shown in Figure 3.9(a), it takes a lot of clock cycles to escape VFI 2 in terms of a VFI 1 viewpoint. The reason is that one clock cycle at VFI 2 is equal to ten clock cycles at VFI 1. Moreover, if the packet is blocked within VFI 2 due to any congestion, its latency may be severely long. This routing path is slower than another routing path P2 which is not the shortest path. Since the routing path P2 detours, it may consume more communication energy and be not free of deadlock and livelock. Actually, most microprocessors operate at several GHz whereas various co-processors such as peripherals, memories, specific-purposed

processors and IO interface logics operate at several hundred MHz. As described above, if any packet generated in the microprocessor operating at GHz clock speed traverses any

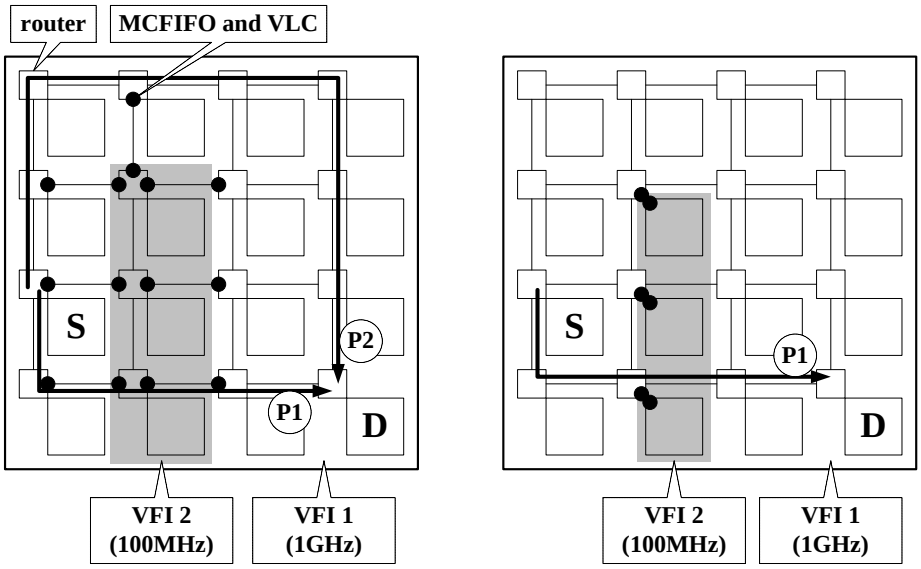


(a) NoC tiles including MCFIFO between routers [84]



(b) NoC tiles including MCFIFO between a core and a router

Figure 3.8: NoC tiles with MCFIFO or VLC placed (a) between routers and (b) a core and a router.



(a) Conventional VFI interface (b) Our VFI interface

Figure 3.9: NoC designs with (a) the conventional VFI interface and (b) the proposed VFI interface.

VFI consisting of the co-processors, it is too delayed at the microprocessor's viewpoint. Even if it is a critical problem that should be solved in VFI-based NoC designs, it is not considered in the previous works yet.

We propose a new VFI interface where MCFIFO and VLC are placed between a core and its router as shown in Figure 3.8(b). It is used together with the VFI interface proposed in [84]. Since the proposed interface makes the clock speed of a router choose one of two clocks, i.e. a clock used in VFI 1 and a clock used in VFI 2, the latency of packets can be greatly improved. In Figure 3.9(b), three tiles in VFI 2 employ the proposed interface with MCFIFO and VLC between a core and its router and the clock speed of their routers (clock domain 3 in Figure 3.8 (b)) is selected to the same as that of VFI 1. Even though any packet generated in VFI 1 traverses VFI 2, its communication latency is not affected from the clock speed of VFI 2. In addition, the proposed VFI-based interface can use fewer MCFIFOs and VLCs. In Figure 3.9, the number of a pair of

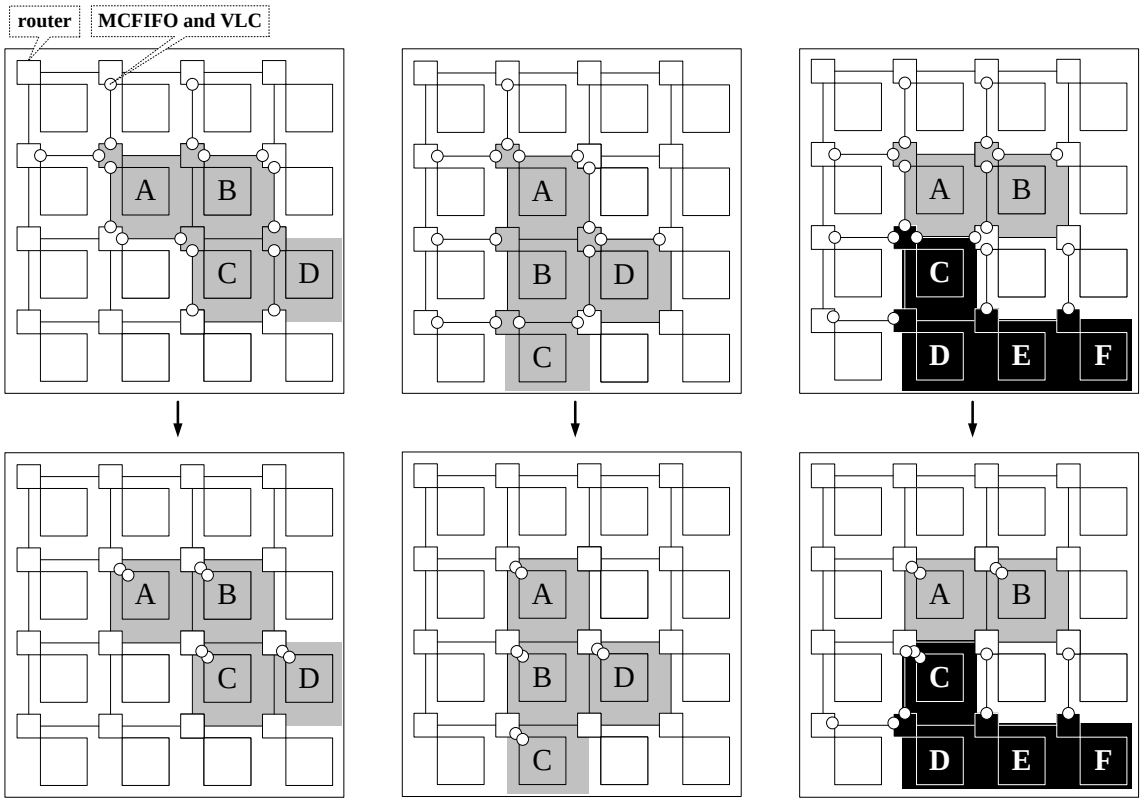
MCFIFO and VLC in the conventional VFI interface is 16 whereas that of the pair in our VFI interface is just 6. Thus, our VFI-aware NoC interface not only reduces communication latency but also further improves VFI energy efficiency.

However, this interface may be effective when cores included in slow VFI are mapped to NoC tiles in a single row or column as shown in Figure 3.9. In order to generate VFI with such a convex region, we used a different candidate selection policy in our VFI-aware mapping algorithm after mapping each core. If any VFI operates at too slow clock speed, candidates for cores in the slow VFI are just selected to either north and south tiles or east and west tiles, but not all north, south, east and west tiles. Moreover, when our VFI-aware NoC design does not meet performance constraints due to long communication latency, we repeat the VFI-aware mapping algorithm with the changed candidate selection policy as shown in Algorithm 5 (line 10). Finally, since cores in the slow VFI are mapped only to tiles in the candidates, the VFI with a convex region can be built with tiles in a single row or column. The mapping restriction may make not only hop count slightly increase but also routers operate at high clock frequency. However, the energy efficiency degraded by such overheads cancels out the energy efficiency improved by fewer MCFIFO and VLC and even the penalty is less than the benefit of our method with a fast on-chip communication speed.

In the proposed VFI-aware NoC design, both interfaces in Figure 3.8 are used together. If VFI 1 and VFI 2 inversely operate at 100MHz and 1GHz clock, respectively, the VFI interface in Figure 3.8(a) and the VFI interface insertion in Figure 3.9(a) are more desirable. Therefore, we need an efficient VFI interface insertion algorithm. As a starting point, we assume to have VFIs interconneccted by the interface with MCFIFO and VLC between routers and have routers located in the upper left corner of cores. Then, we start to replace the conventional VFI interface in Figure 3.8(a) to the proposed VFI

interface in Figure 3.8(b) from VFI operating with the lowest clock. The conventional interface in the upper or left tiles of any VFI can be replaced to the proposed interface if three conditions are satisfied as follows: 1) the upper or left tiles of the VFI must be contacted with different VFI operating at a faster clock speed, 2) the upper or left tiles of the VFI must be surrounded with both upper and lower or both left and right tiles in different VFI and 3) the interface replacement must repeat one time with a updated interface result.

Figure 3.10 shows various examples of our interface replacement, where VFI colored to black operates at the slowest clock speed and VFI colored to white operates at the fastest clock speed. In Figure 3.10(a), tile A, B, C and D satisfy condition 1) and 2) during the first interface replacement (condition 3)). Therefore, the conventional interfaces in tile A, B, C and D are replaced to the proposed interfaces. In Figure 3.10(b), tile A, C and D satisfy condition 1) and 2) whereas tile B does not satisfy condition 2) during the first interface replacement (condition 3)). However, during the second interface replacement after updating the result of the first interface replacement (condition 3)), tile B also satisfy condition 2). Therefore, the conventional interfaces in tile A, B, C and D are replaced to the proposed interfaces. In Figure 3.10(c) consisting of three VFIs, the conventional interfaces of black VFI operating with the slowest clock are first replaced and then the conventional interfaces of gray VFI operating with the next slowest clock are replaced. During the first replacement, the conventional interfaces in tile C and tile B are replaced to the proposed interfaces. During the second interface replacement after updating the result of the first interface replacement (condition 3)), the conventional interface in tile A is also replaced to the proposed interface since tile A meets condition 1) and 2). With such interface insertion, our VFI-aware NoC design can



(a) (b) (c)

Figure 3.10: Examples of the proposed VFI interface insertion.

make communication clock speed so selectable that it is more valuable for a platform- and socket-based NoC design with VFI.

We implement the VFI-based NoC routers consisting of MCFIFO, VLC, single clock FIFO (SCFIFO), an output control (OC) and a crossbar switch as shown in Figure 3.8. As described in Section 3.1.3.3, crossbar switches perform deterministic and minimal path routing to minimize communication energy consumption, based on our routing path allocation rules. The OC is responsible for determining the future departure time of each packet since a physical channel must be reserved and the OC must ensure that there will be sufficient buffer spaces in the next router to store the packets. Our flow control

mechanism adopts winner-take-all bandwidth allocation that allocates all of the bandwidth to one packet until it is finished or blocked before serving other packets. In our VFI-based NoC implementation, MCFIFO and VLC are placed between a core and its router or between routers. MCFIFO and SCFIFO are managed by wormhole flow control or virtual channel flow control mechanisms and a backpressure is used to inform the upstream nodes when they must stop transmitting packets because all of the downstream packet buffers are full. On/off flow control is adopted to avoid the loss of packets as the backpressure.

3.1.4 Experimental Results

In this section, we show experimental results on MPEG-4 video object plane decoder (VOPD) [111] and E3S benchmark suites [25]. As the first application consists of 16 cores, the cores are one-to-one mapped to tiles on 4x4 NoC. The second benchmark has several applications: office-automation, consumer, networking, auto-industry and telecom application containing 5, 12, 13, 24 and 30 tasks respectively. The benchmark also provides the information of 66 processing elements such as the size/cost of the processing elements, the maximum operating frequency, idle power consumption and task power consumption when the tasks are performed in any processor. The tasks are scheduled on to 4, 9, 9, 16 and 25 processors among the 66 processing elements, respectively by [42] to generate a core graph. Then, they are mapped to tiles on 2x2, 3x3, 3x3, 4x4 and 5x5 NoCs, respectively. We compare our VFI-aware NoC design with the previous state-of-the-art work, called VFI-P [84]. Since the previous work is assumed that VFI partitioning with VF assignment is performed in a hard NoC platform where communication and computation components are pre-designed, we implement NMAP

[79], one of the famous mapping and routing path allocation methods combined with VFI-P. We experiment our VFI-aware NoC methodology by three versions, i.e. our VFI-aware mapping algorithm combined with a conventional routing path allocation and a conventional interface, our VFI-aware mapping algorithm combined with the proposed VFI-aware routing path allocation and a conventional interface and our VFI-aware mapping algorithm combined with the proposed routing path allocation and interface to verify the performance of mapping, routing and interface apart, denoted as VFI-M, VFI-R and VFI-I, respectively.

Table 3.1 shows that VFI-M saves more MCFIFOs and VLCs on MPEG-4 VOPD benchmark due to our VFI-aware mapping algorithm generating a convex region with VFI partitioning results. In addition, VFI-R needs the fewest MCFIFOs and VLCs. On the contrary, the VFI-aware NoC approach commonly causes the slight increase of hop count due to the restrictions induced by VFI-aware mapping and routing path allocation. However, the maximum congestion is further relieved because a routing allocation order is considered for balancing network loads. The low congestion makes communication latency shorter and a communication clock speed lower.

Content	Algorithm	2-VF	3-VF	4-VF
# of pair of MCFIFO and VLC	VFI-P [84]	12	22	28
	VFI-M	10	14	20
	VFI-R	2	4	6
Hop count	VFI-P [84]	4309	4309	4309
	VFI-R	4353	4211	4211
Congestion (MB/s)	VFI-P [84]	923	923	923
	VFI-R	516	613	613

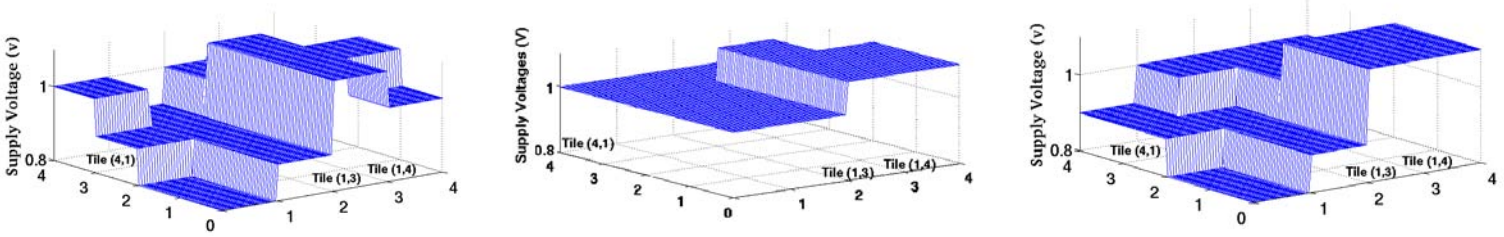
Table 3.1: The comparison of VFI overhead, hop count, and communication congestion on VOPD benchmark.

The thorough cross-comparison of E3S benchmarks is listed in Table 3.2. In this experiment, we assume that the changeable voltage range of cores from Eq. (3.1) is ±0.2V. For example, let three cores operate at 1.2V, 1.7 V, and 1.9V for the minimum energy consumption. Then, if two of three voltages, i.e. 1.7V and 1.9V are assigned for VFI-based NoC, the core operating at 1.2V cannot be scaled into 1.7V. The reason is that the core running at 1.2V can be scaled up to 1.4V by the constraint. Instead, if 1.2V and 1.9V are assigned for VFI-based NoC, the cores can be scaled into 1.2V, 1.9V and 1.9V, respectively. Since the changeable operating range of voltage and clock frequency is small in the most real cores, this constraint is reasonable. Under this condition, tile partitioning with VF assignment by VFI-P generates split VFIs operating at the same VF level since task/core mapping is already performed by a VFI-unaware manner. That is, it may be difficult to include all tiles using the same voltage and frequency in a convex region. As a result, a number of MCFIFOs and VLCs are required to interconnect the separated VFIs as shown in Table 3.2. On the contrary, our VFI-aware NoC design does not generate the split VFIs operating at the same VF level. As a result, fewer MCFIFO and VLCs are required to interconnect the VFIs. Similarly to the MPEG-4 VOPD benchmark, our VFI-aware NoC design makes hop counts increase on the E3S benchmark, yet its energy degradation is canceled out by fewer MCFIFOs and VLCs. Finally, VFI-I requires the fewest MCFIFO and VLC and thus improves energy efficiency by VFI separation. In addition, VFI-I improves 27% communication latency on average.

Figure 3.11 illustrates the visual comparison of VFI-P and our VFI-aware approach all performed on 4x4 NoC for the MPEG-4 VOPD application. Figure 3.11(a) is the result of core/task mapping by NMAP [79]. Since cores are not mapped by VFI-aware manner, tiles mapped to cores using the same VF level are split over NoC as

Content	Algorithm	Telecommunication				Auto-Industry			Networking			Consumer		
		2-VF	3-VF	4-VF	5-VF	2-VF	3-VF	4-VF	2-VF	3-VF	4-VF	2-VF	3-VF	4-VF
# of pair of MCFIFO and VLC	VFI-P	40	48	58	58	22	24	30	16	16	18	8	16	20
	VFI-M	26	28	44	40	16	20	26	12	12	14	6	10	14
	VFI- R	20	22	26	32	12	14	22	10	10	12	6	8	12
	VFI- I	14	20	24	32	8	12	20	8	8	10	6	6	12
Hop count	VFI-P	107	107	107	107	172	172	172	79692	79692	79692	30	30	30
	VFI- R	133	133	138	153	178	193	205	83886	83886	109051	33	35	39

Table 3.2: The comparison of VFI overhead and hop count on E3S benchmark.



(a) VF map after task/core mapping (b) NoC Partitioning by [84] (c) Proposed VFI-Aware NoC

Figure 3.11: Visual comparison of VFI based NoC designs on 4x4 NoC.

shown in Figure 3.11(a). Since tiles that operate at 1V are split to three VFIs, this VFI-based NoC design shows a VF map with six VFIs even if 4 VF levels are used. Based on the VF map, VFI partitioning with VF assignment proposed in VFI-P is performed and its result is shown in Figure 3.11(b). The VFI partitioning achieves the best energy consumption when two VFIs are built. However, some cores operate at higher voltage than the voltage which the cores require to satisfy performance constraints. Therefore, it is suboptimal solution since task/core mapping are already performed by VFI-unaware manner even if the result of VFI partitioning considerably depends on the mapping stage. On the other hand, our VFI-aware approach consisting of core partitioning with VF assignment, VFI-aware mapping and VFI-aware routing path allocation clusters tiles using the same VF level to single VFI such that it clearly provides better partitioned VFI as shown in Figure 3.11(c). As a result, all tiles operate at the optimal voltage and frequency they require to satisfy performance constraints and thus it is beneficial for low energy consumption as well as the global routing of power and clock.

Table 3.3 shows that our VFI-aware NoC design consumes less energy than VFI-P. The reason is that our approach needs fewer MCFIFOs and VLS since all tiles running at the same VF level can be clustered into single VFI. As a result, our VFI-aware NoC optimization further saves energy consumption as the number of VFI increases. On the other hands, the previous state-of-the-art approach VFI-P can improve energy consumption only when the number of VFI built is fewer than 2 or 3. In Network application, our VFI-aware NoC approach is worse than VFI-P since the amount of communication enormously increases when the number of VFI built is 4. However, the best energy consumption (0.76) of our approaches at 3-VFI still outperforms that (0.79) of the previous approach VFI-P. Moreover, our VFI interface achieves lower energy

Benchmark	Algorithm	Normalized Total Energy Consumption			
		1-VFI	2-VFI	3-VFI	4-VFI
Consumer	VFI-P [84]	1	0.56	0.53	0.54
	VFI_ R	1	0.55	0.51	0.50
	VFI_ I	1	0.55	0.50	0.50
Network	VFI-P [84]	1	0.8	0.79	0.79
	VFI_ R	1	0.78	0.76	0.89
	VFI_ I	1	0.77	0.76	0.89
Auto- industry	VFI-P [84]	1	0.69	0.65	0.67
	VFI_ R	1	0.63	0.59	0.58
	VFI_ I	1	0.61	0.58	0.57
Telecom	VFI-P [84]	1	0.58	0.57	0.58
	VFI_ R	1	0.53	0.51	0.49
	VFI_ I	1	0.50	0.50	0.48

Table 3.3: The comparison of energy consumption according to the number of VFI on E3S benchmarks .

consumption with shorter communication latency. Finally, the runtime of our VFI-aware NoC optimization ranges from a few seconds to a few minutes.

3.1.5 Summary

In this section, we proposed a systematic energy optimization framework, including core partitioning with VF assignment, VFI-aware mapping, VFI-aware routing and VFI-aware interface insertion for VFI-based NoC designs. The proposed VFI-aware NoC design makes tiles mapped cores using the same voltage and frequency level clustered to single VFI. In addition, our VFI interface further improves energy consumption with fast on-chip communication. Consequently, our VFI-aware NoC optimization framework reduces VFI design cost that degrades energy efficiency by VFI overheads. Compared to the recent state-of-the-art NoC design technique with VFI [84],

our VFI-aware optimization framework demonstrates an energy efficiency improvement of 10% and the overhead reduction of 82% under a variety of system constraints.

3.2 ARCHITECTURE-AWARE ANALYTIC APPLICATION MAPPING

As thousands of cores will be integrated to a single chip for enhanced performance and functionality, on-chip communication techniques and application mapping algorithms become key factors in the success of the multi- or many-core chips. So far, most of the NoCs have favored a regular mesh network consisting of regular rectangle tiles on which homogeneous processors are placed. The regular mesh network makes application mapping easy, increases routing efficiency, provides desirable electrical and physical properties and reduces the complexity of resource management. Hence, most previous works have optimized their application mapping on the regular mesh architecture as follows.

Murali et al. [79] present NMAP that is a fast algorithm, where tasks are mapped onto a regular mesh network under bandwidth constrains, aiming at minimizing average communication latency. In [39], a branch and bound algorithm is adopted for task mapping in a regular mesh-based NoC architecture, which minimizes the total amount of power consumed in communications. Shin et al. [103] explores the design space of NoC based systems, including task assignment, tile mapping, routing path allocation, task scheduling and link speed assignment using three nested genetic algorithms. The work presented in [15] proposes an efficient technique for runtime application mapping onto a homogeneous NoC platform with multiple voltage levels. Chen et al. in [13] proposes a complier-based application mapping algorithm that consists of task scheduling, processor mapping, data mapping and packet routing to reduce energy consumption. However,

since these solutions have been optimized only for a regular mesh network, they cannot be applied to various networks or their mapping performance gets severely deteriorated in irregular/custom networks.

However, industrial SoC platforms, e.g. Nexperia [28], Nomadik [82] and OMAP [117], consist of various PEs such as a general processor, a digital signal processor (DSP), a specific memory and a peripheral. Since such physically different sized processing elements cannot be floorplanned with a regular mesh topology, the resulting NoCs get an irregular mesh network or even a custom network [11]. The irregular mesh networks are also found in a regular mesh network when some links become faulty or degraded by process and temperature variation. Application mapping and routing path allocation should deal with the abnormal links and compensate for the loss of yield and performance [72]. In addition, since VFI based NoCs have links with different bandwidth [49][53][84], it is no longer a regular mesh network. However, the previous application mapping algorithms are inefficient in performing application mapping in an irregular/custom network since they are not adaptive to various network architectures. As a result, specific mapping algorithms may be required for different network architectures.

Recently, such heterogeneous cores have been considered for low energy consumption. Smit et al. solved the problem of run-time task assignment on heterogeneous processors with task graphs restricted to the small number of vertices or the large number of vertices within degree no more than two [105]. Carvalho et al. investigated the quality of several mapping heuristics promising for run-time use in NoC-based multiprocessor SoCs (MPSoCs) with dynamic workloads, targeting NoC congestion minimization [9]. Chang et al. proposed ETAHM to allocate tasks on a target multiprocessor system [10]. It mixed task scheduling, mapping and dynamic voltage scaling utilization in one phase and couples an ant colony optimization algorithm.

ADAM presented in [29] was run-time application mapping in a distributed manner, targeting for adaptive NoC-based heterogeneous MPSoCs. However, the previous application mapping solutions have not considered the irregularity of NoC tiles and links which are caused by different-sized heterogeneous PEs. Since the irregularities cause long detoured packets on a network, a lot of communication energy may be consumed or a quality-of-service requirement may not be guaranteed. Recently, Tornero et al. proposed a communication-aware topological mapping technique for irregular NoCs, which matched the communication requirements of the application running on the cores with the existing network resources [109]. However, its mapping quality was not still satisfactory since it did not provide the efficient solution searching algorithm. Therefore, an application mapping algorithm that can be applied to various networks should be required. This problem was also addressed as an open problem (P2) in [71].

Such different-sized PEs is only considered in the latest application-specific NoC methodologies. Chatha et al. in [11] present the design methodology and synthesis of application-specific NoC architecture. It employs a three-phase synthesis approach consisting of core-to-router mapping, custom topology decision, and route generation. In [100], an adaptive deadlock free routing algorithm is proposed to handle NoC layouts with embedded different-sized cores. Authors in [6] propose hardware-efficient routing in irregular mesh NoCs and routing table size minimization based on static shortest path routing. Holsmark et al. in [38] list the issues that a designer would encounter while designing a heterogeneous mesh topology for NoC using multi-port or multi-access point cores and present two deadlock-free routing algorithms for irregular mesh networks.

In this section, we propose novel and global architecture-aware analytic mapping (A3MAP) algorithms. The proposed approach can be employed in most networks

including regular/irregular mesh and custom networks. The main novelties and contributions include:

- We propose a simple yet efficient metric space to easily capture the architecture of NoC and the communication of cores. Then, an application mapping problem is exactly formulated to MIQP based on a metric embedding technique.
- We propose two effective heuristics solving the MIQP, based on a successive relaxation algorithm providing short runtime and a genetic algorithm providing high mapping quality. They fit well our formulation and provide better trade-off between mapping quality and runtime for a small-scale network.
- We propose a partition-based application mapping approach for large-scale networks and show that it provides short runtime with little loss of mapping quality.
- We show that the proposed A3MAP algorithms achieve excellent application mapping quality not only in regular networks but also in irregular/custom networks.

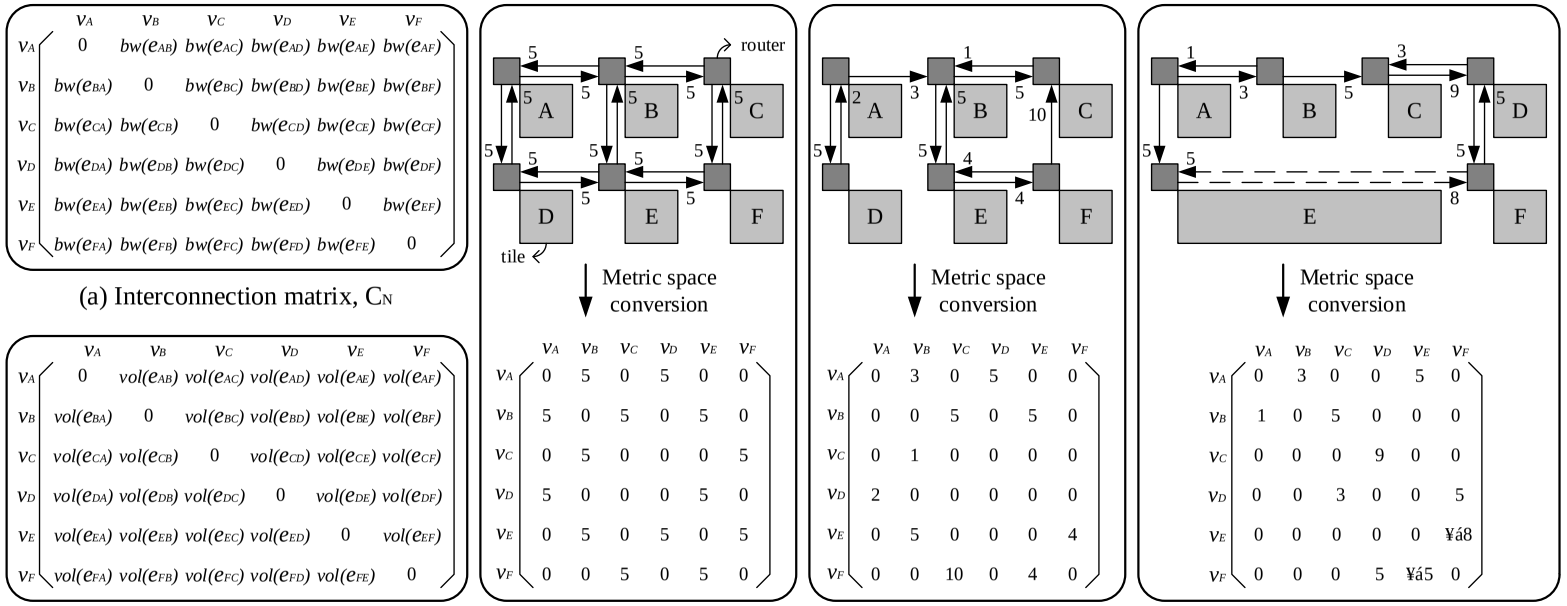
3.2.1 Problem Formulation

In this section, we formulate an application mapping problem to MIQP using metric embedding. As inputs, we take a core graph and a network. A graph $G(V,E)$ with n vertices is a directed graph, where each vertex $v_i \in V$ represents a core or a tile and where each directed edge $e_{i,j} \in E$ represents communication between v_i to v_j. $vol(e_{i,j})$ represents communication volume between v_i to v_j in a core graph and $bw(e_{i,j})$ represents a bandwidth requirement between v_i to v_j in a network. We construct an $n \times n$

interconnection matrix, C_N corresponding to a network, where $c_{Ni,j} \in C_N$ is equal to $bw(e_{i,j})$ as shown in Figure 3.12(a). Each row in C_N represents interconnection relation with respect to a single tile on NoC. Thus, C_N contains interconnection relations for an entire network, representing the metric space of a network. Similarly, we construct an $n \times n$ interconnection matrix C_C, corresponding to a core graph, where $c_{Ci,j} \in C_C$ is equal to $vol(e_{i,j})$ as shown in Figure 3.12 (b).

For example, Figure 3.12 (c), (d) and (e) show three network graphs and their metric spaces using the proposed interconnection matrix. In Figure 3.12 (c) that is a regular mesh, all routers are interconnected by a bidirectional network link with the same bandwidth. Its interconnection matrix is symmetrically composed as shown under the network graph. In case of an irregular mesh network in Figure 3.12(d), interconnections between tile A and tile B or between tile C and tile F are unidirectional and tile D is not interconnected to tile E. Since the bandwidth of links is also different, its interconnection matrix is asymmetrically composed. The irregular mesh network can be observed in VFI based NoC where each PE operates with its own voltage and frequency [49][53][84] and in NoC with faulty and degraded links by process and temperature variation [72].

In case of a custom network, there is slightly difference in the composition of its interconnection matrix. In Figure 3.12(e), wirelength between tile E and tile F is different from other wirelengths due to tile E with a larger area. Since a packet have to cross each link within one cycle, a link between tile E and F may have more repeaters to accommodate a fast transmission time resulting in significantly higher energy consumption. The composition of an interconnection matrix for the custom network is similar to regular/irregular mesh networks except weight α is added in the matrix in order to consider efficient communication energy consumption. The hop count based on the assumption that all links consume the same communication energy is no longer suitable



(a) Interconnection matrix, C_N

(b) Interconnection matrix, C_C

(c) regular mesh network

(d) irregular mesh network

(e) custom network

Figure 3.12: Various graphs and their interconnection matrices.

since links with different wirelength consume different communication energy. Let the energy consumption of each link, E_{link} computed as:

$$E_{link} = E_{driver} + E_{repeaters} \tag{3.9}$$

where E_{driver} and $E_{repeaters}$ are the energy consumed by the output driver of routers and repeaters on a link respectively. If E_{link1} and E_{link2} is the energy consumption of sending one bit in a solid line and a dotted line respectively, α is the ratio of E_{link1} to E_{link2} ($=E_{link1}/E_{link2}$) where $E_{link1} < E_{link2}$. The weigh α ($0<\alpha<1$) reduces the available bandwidth of a long dotted link in a network such that our formulation makes the long dotted link less used. In Figure 3.12(e), let's suppose that dotted lines are three times longer than solid lines and a packet generated in tile *A* goes to tile *D*. The packet can choose either *A-B-C-D* or *A-E-F-D* as a routing path. Since two routing paths include the same hop counts, it takes the packet the same clock cycle to reach tile *D* while the total wirelengh of path *A-E-F-D* is longer than that of path *A-B-C-D*. Thus, the path *A-E-F-D* may consume more communication energy than the path *A-B-C-D* since more repeaters may be inserted on the long dotted links or the router attached to tile *E* and *F* may be required to equip a stronger output driver. Therefore, it is good to assign a core with little communication to a tile with the long link or a core with a lot of communication to a tile with the short link for low dynamic energy consumption. If the energy consumption is linearly proportional to the length of wires in Figure 3.12(e) due to more repeaters and a stronger output driver, α is 1/3. The weight α lets a core with a lot of communication mapped into a tile with short wires such that communication energy consumption can be further minimized. Similarly, our interconnection matrix easily accommodates other general cases.

Graph embedding [74] maps the vertices of graph *G(V,E)* into a chosen metric space by minimizing distortion. Thus, application mapping has a natural correspondence with graph embedding into a given two-dimensional metric space representing NoC. Thus, we seek to embed a core graph into the metric space of a network based on the interconnection matrices. The goal is that a core is mapped to each tile, satisfying the performance constraints in a core-mapped network while the number of communication generated between routers is minimal. If a network is exactly same as a core graph, graph embedding does not cause any distortion of the edges in the core graph. As a result, it always produces the best possible mapping quality on the network. However, since most core graphs are generally different from a network, some distortion is not evitable in a network. Then, the mapping quality is measured by the total distortion of embedding. By minimizing the extent by which edges in a core graph are stretched or distorted with intermediate tiles when embedded into a network, we seek to reduce the total amount of communications and obtain a better global application mapping solution in terms of energy consumption under performance constraints. Based on this concept, our concrete application mapping algorithm is formulated as follows.

With two interconnection matrices, C_N for a network and C_C for a core graph, we exactly formulate an application mapping problem to mixed integer quadratic programming (MIQP). It is similar to a field programmable gate array (FPGA) placement problem proposed in [32]. However, a crucial difference in our work is the use of metric space that accurately captures the interconnections of a network and a core graph. The application mapping problem is equivalent to determining the assignment of a core to each tile with low energy consumption under performance constraints. This core assignment action can be mathematically presented by an $n \times n$ permutation matrix P. Column indices and row indices in P represent core identifiers and tile identifiers,

respectively. For example, if *P(i,j)=1*, then core *j* is mapped to tile *i*. Thus, only one element in each row and each column of *P* can be 1; all others must be 0. The action of *P* on a core graph is represented by $P^T C_C P$. Finally, *P* minimizing the difference between the permuted interconnection matrix of a core graph $P^T C_C P$ and the interconnection matrix of a network C_N for generating little communication between routers and minimizing the distortion of C_C for a short routing path can be found. For *P* that is orthogonal, we formulate the application mapping problem mathematically by our objective as:

$$\min f_{obj} = \left\| P^T C_C P - C_N \right\|_F^2 = \left\| C_C P - P C_N \right\|_F^2 \tag{3.10}$$

where $\|X\|_F = \sqrt{\sum_i \sum_j x_{i,j}^2}$, $x_{i,j} \in X$, i.e., the Frobenius norm of the matrix *X* and $x_{i,j} \leq 0$ to satisfy bandwidth constraints, subject to integrity and linearity constrains as follows:

$$\sum_{i=1}^{n} P(i, j) = 1, \forall j = 1, 2, ..., n \tag{3.11}$$

$$\sum_{j=1}^{n} P(i, j) = 1, \forall i = 1, 2, ..., n \tag{3.12}$$

$$P(i, j) \in \{0, 1\} \tag{3.13}$$

The constraints indicate that just one element in each row and each column is 1 and other elements are 0 in the permutation matrix *P*.

While our formulation has a convex quadratic object function, the binary constraints on the elements of P restrict the solution space to a non-convex set. Thus, convex optimization techniques like gradient descent cannot be directly applied to solve this problem. Actually, this type of formulation is well known as MIQP that is NP-hard [97]. Algorithms we take in MIQP are successive relaxation to quickly find an application mapping solution and a genetic algorithm to achieve a high mapping quality.

In the next section, we describe how they are applied in the proposed A3MAP formulation minutely.

3.2.2 A3MAP Algorithms

We present an effective heuristic based on successive relaxation of MIQP to a sequence of quadratic programming (QP), called A3MAP-SR, to quickly find the permutation matrix P that minimizes our objective f_{obj} in Eq. (3.10). In addition, we apply a genetic algorithm to find a better mapping solution, called A3MAP-GA even though it takes a longer runtime than A3MAP-SR. A genetic algorithm is an efficient random searching algorithm based on a cycle crossover and a mutation operation.

3.2.2.1 A3MAP-SR

In this section, we solve our A3MAP formulated to MIQP based on a successive relaxation algorithm [33]. The optimal MIQP formulation can become QP if we relax the discrete constraint of Eq. (3.13) to a continuous constraint as follows.

$$0 \le P(i, j) \le 1 \tag{3.14}$$

Then, the key idea behind this algorithm is to use this QP as a subroutine. QP is solved much faster and scaled much better. Then, continuous values obtained by a QP solver are guided to 0 or 1 depending upon a predefined threshold. If any continuous value is less than the threshold, it is guided to 0. Otherwise, it is guided to 1. The proposed concrete successive relaxation algorithm employed in our A3MAP formulation is shown in Algorithm 7.

Algorithm 7 A3MAP-SR

Input: ***MIQP formulation***

1: Relax $P(i,j) \in \{0,1\}$ to $0 \leq P(i,j) \leq 1$;
2: Set all $P(i,j)$ to a variable;
3: $i_{th} = n(V_N)$;
4: **repeat**
5: Solve relaxed MIQP only for variables $P(i,j)$;
6: **repeat**
7: Find $max\{P\}$ and store its location to (i_{max},j_{max}) for $\forall P(i,j)$ that is a variable;
8: **if** $max\{P\} \geq 1/ i_{th}$ **do**
9: $P(i_{max},j_{max}) = 1$ and a non-variable;
10: $P(i,j_{max}) = 0$ and a non-variable, $\forall i=1,2,.., n(V_N)$;
11: $P(i_{max},j) = 0$ and a non-variable, $\forall j=1,2,.., n(V_N)$;
12: i_{th} decreases by 1;
13: **end if**
14: **until (**$max\{P\} < 1/ i_{th}$)
15: **until** (all $P(i,j)$ are a non-variable)

Output: ***Permutation matrix P***

After relaxing the constraint of Eq. (3.13) to Eq. (3.14) in line 1, we set all $P(i,j)$ to a variable since any $P(i,j)$ is not guided to a permanent value, 0 or 1. Initial i_{th} is set to the number of tiles in a network and then as an initial threshold, we use $1/i_{th}$ to guide continuous $P(i,j)$ solved by a QP solver to 1, where the threshold indicates the expected average that variable $P(i,j)$ can get. On executing the successive relaxation, i_{th} decreases by 1 whenever any $P(i,j)$ is set to 1, which means the threshold gets increased. The rest of algorithm 1 attempts to constraint continuous values solved by a QP solver to binary values inversely. We look for the maximum $P(i,j)$ and compare it to the threshold. If it is greater than the threshold, it is set to 1 and a non-variable. In addition, all elements on the same row or column as the maximum $P(i,j)$ are also set to 0 and a non-variable since the sum of elements on a single row or a single column in the permutation matrix P should be 1 from the constraints in Eq. (3.11) and (3.12). This procedure repeats if the next maximum $P(i,j)$ is also greater than the updated threshold. Otherwise, we again solve the relaxed MIQP for the rest of variables $P(i,j)$ by a QP solver and continue to guide

continuous values to binary values. If all *P(i,j)* are guided to 0 or 1, we get the near-optimum permutation matrix *P*.

For example, we assume that an application with 5 cores and NoC with 5 tiles are given. In order to allocate the cores to the tiles, we should find a 5×5 permutation matrix where only 5 elements will be 1 and 20 elements will be 0 from our A3MAP formulation. We relax the discrete constraint in Eq. (3.13), set all *P(i,j)* to a variable and set an initial threshold to 1/5. Let the relaxed MIQP solved by a QP solver as shown in Figure 3.13(a). Then, our A3MAP-SR algorithm looks for the maximum *P(i,j)* among the variables. In Figure 3.13(a), *P(3,2)* is 0.5 as the maximum. Since it is greater than the initial threshold 1/5, *P(3,2)* is guided to 1 and then *P(3,k)* and *P(k,2)* where $\forall\, k = 1, 2, \ldots, 5$ are set to 0 as shown in Figure 3.13(b). The guided *P(i,j)* to 0 or 1 is set to a non-variable and the threshold is updated to 1/4. Since the next maximum *P(1,3)* and *P(5,5)* are greater than the threshold 1/4 and 1/3 respectively, *P(1,3)* and *P(5,5)* is guided to 1 and *P(1,k), P(k,3),*

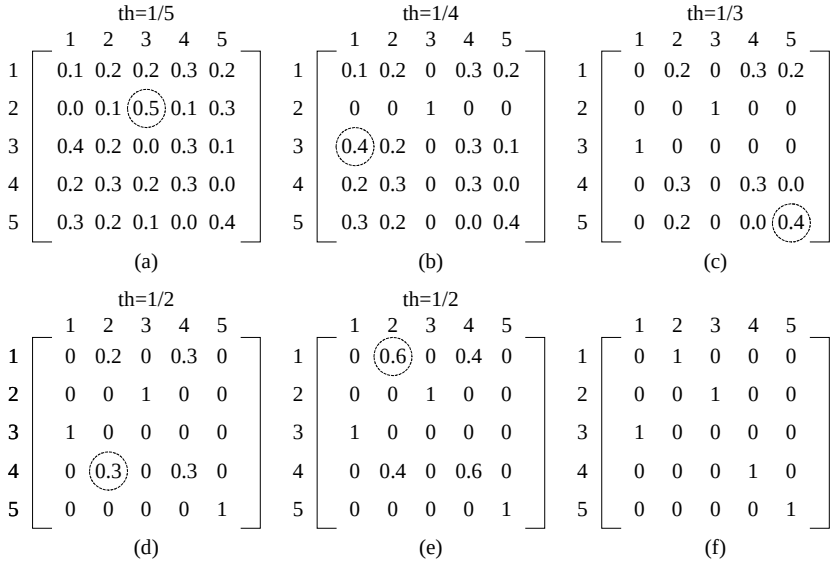


Figure 3.13: Guiding continuous *P(i,j)* to binary *P(i,j)* after solving QP.

P(5,k), and *P(k,5)* where $\forall k = 1, 2, \ldots, 5$ are set to 0 as shown in Figure 3.13(c) and (d). Then, the guided *P(i,j)* to 0 or 1 is set to a non-variable and the threshold is updated to 1/2. Next, since the next maximum *P(2,4)* is less than the threshold 1/2 in Figure 3.13(d), we again solve the relaxed MIQP by a QP solver for the rest of variable *P(i,j)* as shown in Figure 3.13(e). Then, the guiding procedure repeats until all *P(i,j)* are guided to 0 or 1 as shown in Figure 3.13(f).

3.2.2.2 A3MAP-GA

The successive relaxation algorithm solves MIQP with the reasonable mapping quality and runtime. For application mapping with a high mapping quality, runtime may be less important than the reduction of hop count and communication energy consumption. To reflect this demand, we develop another heuristic using a genetic algorithm. A genetic algorithm reproduces the principle of natural evolution to solve search and optimization problems. It is a promising technique for a system-level design and is especially suitable for multiple-objective optimization problems. Starting with an initial population, a genetic algorithm evolves a population using crossover and mutation operations. A genetic algorithm is previously used in [103] to explore the design space efficiently for task assignment, mapping and routing path allocation. However, since the performance of a genetic algorithm depends on encoding, crossover and mutation schemes, we need to select different schemes that fit well in our A3MAP formulation.

Algorithm 8 is the pseudo-code of our genetic algorithm for MIQP. First, we generate two arbitrary permutation matrices as parent individuals. A crossover **s**cheme is widely acknowledged as critical to the success of a genetic algorithm. A crossover scheme should be capable of producing a new feasible solution (i.e., new child

individual) by combining the good characteristics of parent individuals while the child individuals should be considerably different from their parent individuals. We use a cycle crossover [85] which prevents over two cores being allocated into the same tile. Figure 3.14 shows how to generate two child individuals based on the cycle crossover. In the first step, child 1 inherits a column from parent 1 and child 2 inherits a column from parent 2. We start to choose any inherited column in parent 1. In Figure 3.14(a), the first column is arbitrarily chosen in parent 1 and then the same column is chosen in parent 2. Next, we look for a column in parent 1 including the same elements that the chosen column in parent 2 gets. In Figure 3.14(a), the fifth column of parent 1 contains the same elements that the first column of parent 2 gets. Then, the fifth column in parent 2 is selected. Similarly, this procedure repeats until the chosen column is again chosen. In the second step, child 1 inherits a column from parent 2 and child 2 inherits a column from parent 1 inversely. The procedure is similar to the first step except the choice of a column starts from any unselected column of parent 2. If all columns of children are not filled with the column of parents after the second step, we repeat the first step and the second step with the unselected columns of parents by turns. In our example, all columns of children are filled after the second step.

Algorithm 8 A3MAP-GA

Input: ***MIQP formulation***

1: Generate arbitrary parent 1;
2: **repeat**
3: Generate arbitrary parent 2;
4: (child 1, child 2) = cycle crossover (parent 1, parent 2);
5: Mutation of child 1 and 2 by pair-wise swapping;
6: parent 1 = one of two children with minimum f_{obj} computed by Eq. (3.10) for the next evolution;
7: **until** (no improvement during *i*-iterations)

Output: ***Permutation matrix P***

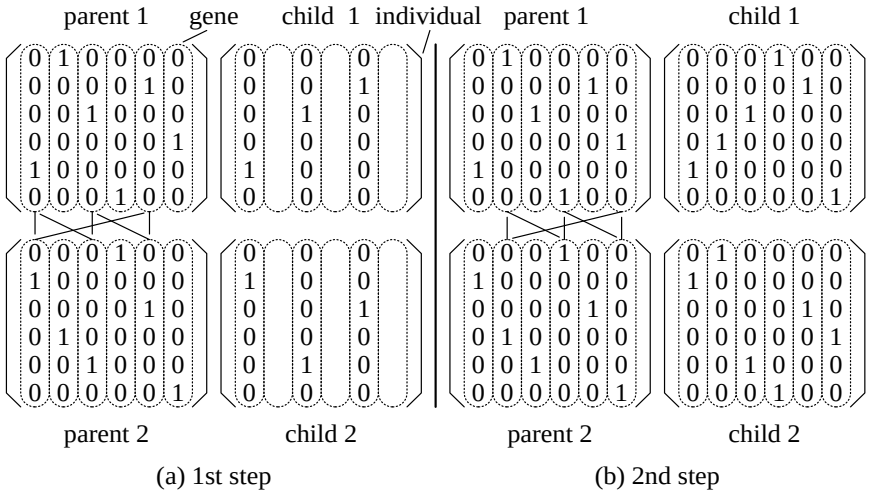


Figure 3.14: Cycle crossover.

Then, a mutation operation is performed for each child. In this operation, two columns randomly selected are swapped to generate a new individual. Then, the swapping is valid only when it reduces the number of traffic. The pair-wise swapping operation for each child continues until the pair of swapped columns cannot minimize our object function, i.e. Eq. (3.10) any more. After the mutation operation, we choose one of two children with the minimum distortion as the parent 1 for the next evolution. Those operations repeat until there is no improvement for several (i) iterations. If there is not any improvement for i-iterations, a permutation matrix providing the near-optimal performance to NoC is obtained.

Our genetic algorithm makes the superior column of parents passed down to their children and the best child again becomes any parent (parent 1) for the next evolution. In addition, since the new elements of columns (parent 2) from the outside are supplied, the possibility of local minima is relatively lower. This approach can efficiently cover wider solution spaces even if runtime is longer than A3MAP-SR.

3.2.3 A3MAP for Large-Scale NoC

Whereas A3MAP enables global optimization, the runtime of A3MAP algorithms is longer and longer as the number of cores or tiles increases. Even NMAP [79] that is one of the fastest mapping algorithms takes a long runtime when the number of cores or tiles is greater than 70. Recently, since NoCs include more tiles for high performance and applications are more complex, an application mapping approach with better tradeoff between runtime and mapping quality is required. Therefore, we propose a partition-based application mapping approach that can be easily extended to any large-scale NoC. In addition, we show that A3MAP algorithms are suitable for the partition-based approach.

Figure 3.15 shows our partition-based application mapping approach in large-scale NoCs, where an application with 9 cores and NoC with 9 tiles are given for a simple explanation. In Figure 3.15, core 1, 5 and 6 have two times higher computation complexity than others and the weight of all edges is just 1. We first perform *k*-way min-cut partitioning for the cores. The number of groups *(k)* is determined by a user. If runtime is more important than mapping quality, a few groups are desirable. Otherwise, few groups are desirable to high mapping quality. The groups are not required to include the same number of cores. Then, the groups are sorted in a decreasing order by the amount of communication inside each group and then mapped in the order. Since the communication volume of group 1, 2, and 3 are 4, 3, and 2, respectively, group 1 are first mapped in Figure 3.15.

Large NoC (R) including N tiles also requires being partitioned to several small networks (R') with a convex region as shown in Figure 3.15. A near convex region selection problem can be formulated as follows [15]:

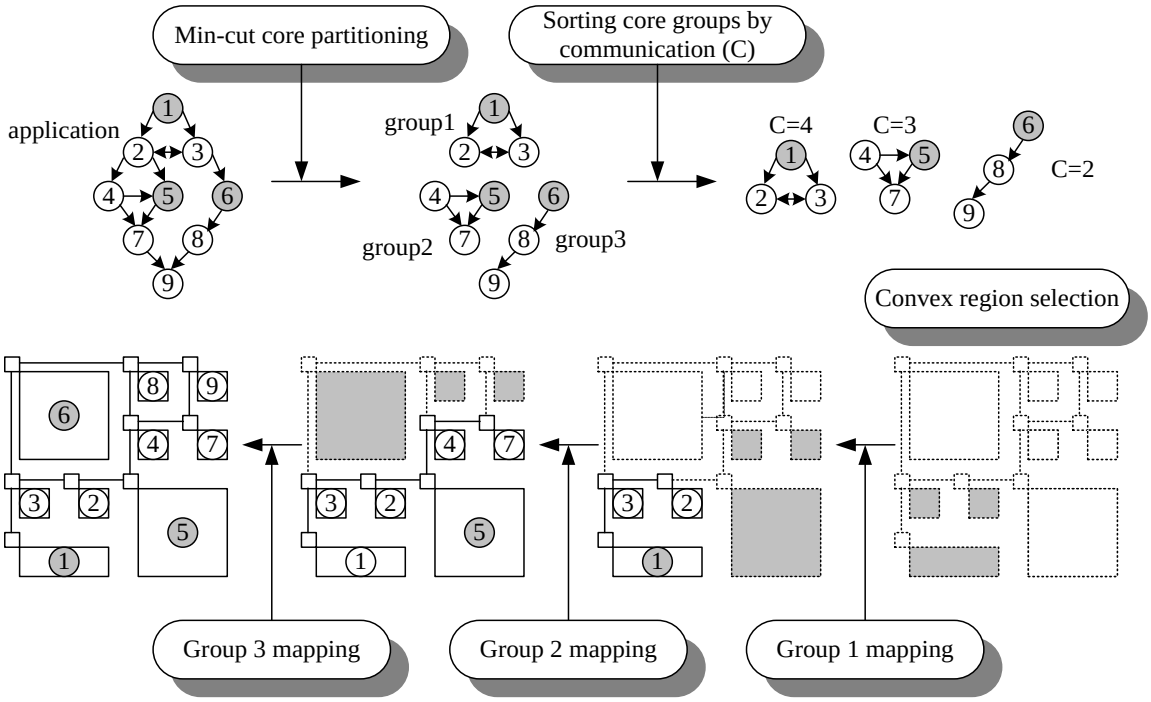


Figure 3.15: Partition-based A3MAP flow for large networks and complex applications.

$$\min\left[L_1(R') + L_1(R - R')\right] \tag{3.15}$$

where $L_1(R)$ is the total Manhattan distance between all tiles inside region R. The objective is to find a subregion R' with N' tiles of which the total computational capacity is greater than or equal to the total computational complexity of cores. The time complexity of the near region selection algorithm is known as $O(NlogN)$ in [15].

Then, cores in each group are mapped to tiles inside the selected convex region. Group 1 which has the maximum communication is first mapped to a possible convex region as shown in Figure 3.15. Since the size of an interconnection matrix of group 1 is 3×3, the runtime of A3MAP algorithms is several hundred times faster than A3MAP algorithms with a 9×9 interconnection matrix. Next, based on the mapping result of group 1, group 2 is mapped to a possible convex region. Even though the size of its

interconnection matrix is 6×6 in the mapping of group 2, its runtime is similar to that of group 1. This is because some variables *P(i,j)* are already determined by the mapping of group 1. Last, group 3 is mapped to the last convex region, based on two previous mapping results. Whereas the size of its interconnection matrix is 9×9 in the mapping of group 3, its runtime is also similar to that of the previous core groups. This is because the number of variables *P(i,j)* that A3MAP algorithms should find in the last mapping are the same as the number of variables *P(i,j)* in the first and second mapping.

Even though most of the application mapping algorithms can be applied to the partition-based approach, our A3MAP algorithms such as A3MAP-SR and A3MAP-GA are more suitable. This is because the runtime of A3MAP solved by a full search algorithm, called A3MAP-FS is still slow and the mapping quality of NMAP is not still satisfactory in the partition-based approach. The partition-based approach gets some inevitable mapping quality loss since cores in each group can be allocated to only tiles inside the selected convex region. In order to minimize the mapping quality degradation in the partition-based approach, each group should include cores or tiles as many as possible, i.e. the number of a group should be the minimum. However, since it takes A3MAP-FS at least 2 seconds to map 10 cores in our experiment, the number of a group should increase in order to reduce the runtime. As a result, the mapping quality of A3MAP-FS deteriorates severely in the partition-based approach. In case of applying a fast application mapping algorithm like NMAP in the partition-based approach, its mapping quality is not satisfactory even though the number of a group is the minimum. For example, NMAP itself shows on average 7%, 8% and 11% lower application mapping quality than A3MAP-FS when 10, 11 and 12 cores are mapped, respectively. As a result, even if NMAP performs on a large convex region with a number of cores, its mapping quality is still low in the partition-based approach. Therefore, A3MAP

algorithms that provide higher mapping quality than NMAP and shorter runtime than A3MAP-FS are suitable for the partition-based approach.

3.2.4 Experimental Results

We implement the A3MAP-SR algorithm by CPLEX11.2 [87] and the A3MAP-GA algorithm by C++. All experiments were performed on a Linux machine with Intel 2.4GHz CoreDuo and 8GB RAM. We repeat each application mapping for ten times and compute their average to obtain reliable statistics.

3.2.4.1 Regular Mesh Network

We carry out experiments by applying A3MAP algorithms on an MPEG-4 video object plane decoder (VOPD) [111], E3S benchmark suites [25] and synthetic benchmarks. The first application including 16 cores is mapped onto a 4×4 regular mesh network. The second benchmark consists of three applications, i.e. consumer, auto-industry (AI) and telecomm containing 12, 24 and 30 tasks respectively, which are scheduled to 9, 16 and 25 by [42] and then mapped to a 3×3, 4×4 and 5×5 regular mesh network, respectively. In addition, we use task graph for free (TGFF) [26] to generate several sets of synthetic applications. The number of tasks and the volume of communication are randomly selected according to specific distributions.

Since the number of cores is generally different from the number of tiles, the pre-processing is required. If the number of cores is less than the number of tiles, additional cores without any communication and computation are added in the core graph. If the number of cores is greater than the number of tiles, we perform $n(V_N)$-way min-cut or balanced core partitioning, where $n(V_N)$ is the number of tiles and the computational

complexity of the grouped cores must be less than the computational capacity of PE. The min-cut partitioning reduces communication energy consumption between tiles that are assigned cores whereas the balanced partitioning for the computational complexity of cores improves the system performance by encouraging parallel computing. Then, we perform the proposed A3MAP-SR and A3MAP-GA algorithms. Finally, we allocate the routing path of packets by a Dijkstra's shortest path algorithm to compute total hop count between routers on a given network.

Table 3.4 shows how exact and fast solution A3MAP algorithms can find in synthetic benchmarks with 9-13 cores, compared to the full searching approach, A3MAP-FS that provides the best solution in terms of mapping quality. A3MAP-SR and A3MAP-GA provide near-best solutions since their mapping qualities are just 3.8% and 2.2% lower on average than A3MAP-FS respectively, when 13 cores are mapped in a regular mesh network. However, their runtimes are about 564 and 153 thousand times shorter than A3MAP-FS respectively. On the contrary, the mapping quality of NMAP [79] which is one of the most famous core mapping algorithms is on average 15.3% lower than A3MAP-FS even if its runtime is the shortest.

# of core or tile	Hop count increase (%) normalized by A3MAP-FS			Runtime improvement (times) normalized by A3MAP-FS		
	A3MAP-SR	A3MAP-GA	NMAP	A3MAP-SR	A3MAP-GA	NMAP
9	1.3	1.1	2.0	155	72	202
10	1.7	1.2	7.1	432	373	470
11	2.0	1.5	8.1	3819	1555	5287
12	2.6	1.8	10.9	47K	14K	67K
13	3.8	2.2	15.3	564K	153K	875K

Table 3.4: The hop count increase and runtime improvement of NMAP, A3MAP-GA, and A3MAP-SR normalized by A3MAP-FS.

Table 3.5 shows the application mapping results performed with industrial benchmarks. A3MAP-SR greatly reduces on average total hop count by 7.4% in a regular mesh network, compared to NMAP. A3MAP-GA achieves on average 3.8% and 11.8% less hop count than A3MAP-SR and NMAP, respectively. On the contrary, the runtimes of A3MAP-SR and A3MAP-GA are longer than NMAP as shown in Figure 3.16.

Application	NMAP	A3MAP-SR	Imp. (%)	A3MAP-GA	Imp. (%)
Consumer	50	50	0	49	2
VOPD	4309	4265	1.0	4141	3.9
AI	187	151	19.3	147	21.4
Telecomm	127	115	9.4	102	19.7
Average	4673	4581	7.425	4439	11.75

Table 3.5: The comparison of hop count for industrial benchmarks in regular mesh networks.

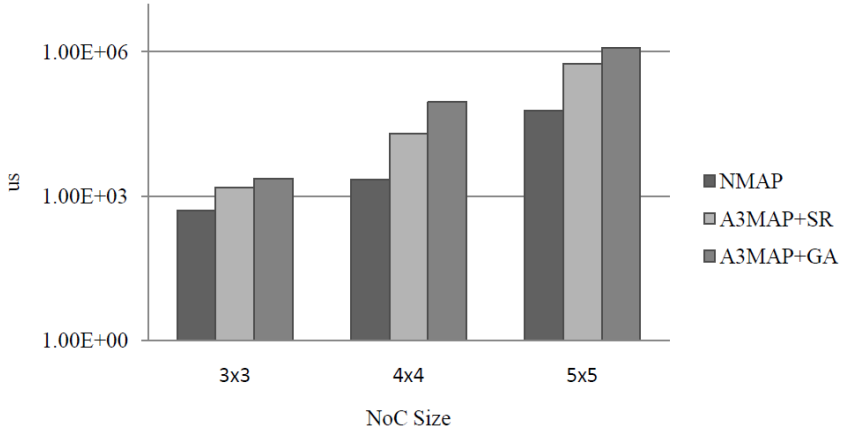


Figure 3.16: The comparison of runtime for industrial benchmarks in 3×3-5×5 regular mesh networks.

Figure 3.17 shows the hop count improvement of A3MAP algorithms compared to NMAP on 3×3-10×10 regular mesh networks. We generate ten synthetic task graphs

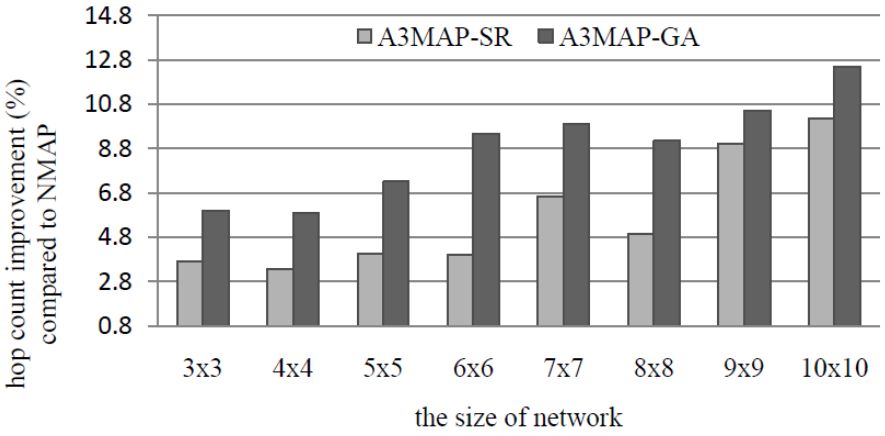


Figure 3.17: The hop count improvement of A3MAP algorithms compared to NMAP for synthetic benchmarks in 3×3-10×10 regular mesh networks.

per network by TGFF and compute their average improvement. As shown in Figure 3.17, A3MAP-SR and A3MAP-GA reduce on average total hop count by 5.7% and 8.8%, respectively, compared to NMAP. In addition, A3MAP algorithms provide much higher mapping quality than NMAP as the size of networks increases.

3.2.4.2 Irregular Mesh Network

In this section, our A3MAP algorithms prove more merits on irregular mesh networks. We perform NMAP on an irregular mesh network even if NMAP is optimized for a regular mesh network. We also implement [109] which considers irregular networks for application mapping, called CMAP. Figure 3.18 shows six irregular mesh networks on which we experiment the application mapping algorithms. Figure 3.18(a) has only bidirectional links, Figure 3.18(b) has both bidirectional and unidirectional links and Figure 3.18(c) has only unidirectional links. Both directions of links have the same bandwidth in Figure 3.18(d) whereas each direction of links has different bandwidth in

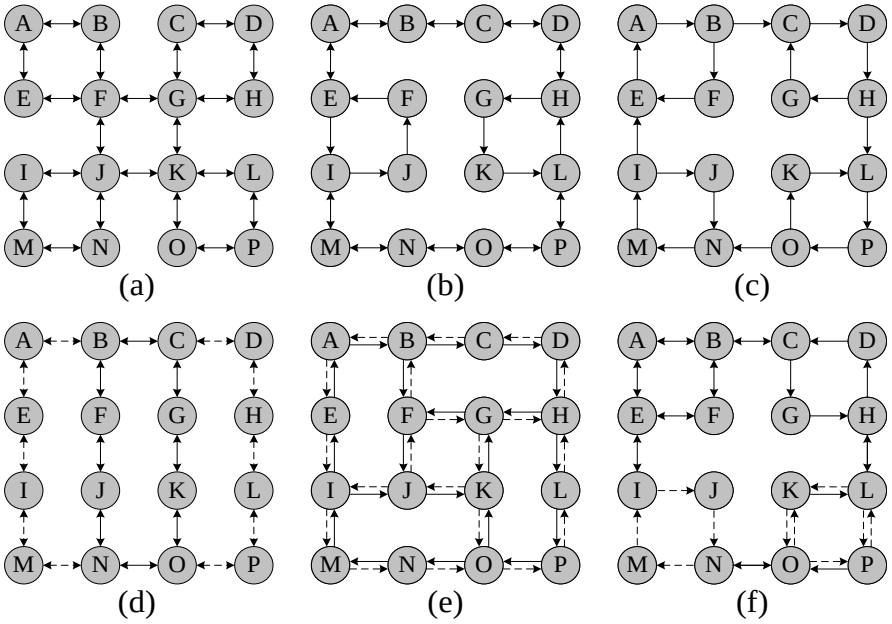


Figure 3.18: Irregular mesh networks used in our experiments.

Figure 3.18(e). In the figure, solid lines have two times higher bandwidth than dotted lines. In Figure 3.18(f), links have all irregularities mentioned in Figure 3.18(a)-(e).

Table 3.6 shows the mapping results of an MPEG-4 VOPD application on the irregular mesh networks. A3MAP algorithms achieve better application mapping improvement in an irregular mesh network than that in a regular mesh network. For example, whereas A3MAP-SR and A3MAP-GA reduce on average total hop count only by 1.0% and 3.9% in a regular mesh network respectively, they reduce on average total hop count by 16.1% and 29.4% in irregular mesh networks respectively, compared to NMAP. In addition, A3MAP algorithms achieve much higher mapping quality than NMAP in Figure 3.18(f) which is the most complex network. That is because A3MAP formulation avoids mapping cores with a lot of communication volume to tiles with little bandwidth and considers the direction of communication and various network topologies adaptively. Even if CMAP is optimized for irregular networks, its mapping quality is

Application	NMAP	CMAP	A3MAP-SR	A3MAP-GA
Figure 3.18(a)	4869	4668	4839	4237
Figure 3.18(b)	5699	6552	4619	4457
Figure 3.18(c)	7810	8992	7317	4619
Figure 3.18(d)	4923	5507	4301	4295
Figure 3.18(e)	5706	4183	4199	4183
Figure 3.18(f)	8103	7920	4844	4410
Average	6185	6304	5187	4367
Ratio	1	1.019	0.839	0.706

Table 3.6: The comparison of hop count for VOPD benchmark in various irregular mesh networks.

lower than that of NMAP. Since CMAP just considers the irregular wirelength of links, it cannot improve mapping quality on irregular mesh networks which have the different direction and irregular bandwidth of links. Furthermore, the runtime of CMAP is slightly slower than NMAP. As a result, A3MAP algorithms provide energy-efficient application mapping to NoC including an irregular mesh network.

3.2.4.3 Custom Network

In this section, we perform A3MAP algorithms on custom networks with an MPEG-4 VOPD benchmark and then it is compared to NMAP and CMAP. Figure 3.19 shows custom networks where A3MAP algorithms are performed. Figure 3.19(a) contains three PEs that have four times larger area than others. Due to the PEs, a custom network including irregular interconnections and different wirelengths is synthesized. Similarly, 16 PEs that have one of three different areas are floorplanned as shown in Figure 3.19(b). Figure 3.19(c) has both unidirectional and bidirectional links and Figure 3.19(d) has links with different bandwidth. In Figure 3.19, links have one of two different wirelengths and assume that a long link consumes two times higher communication

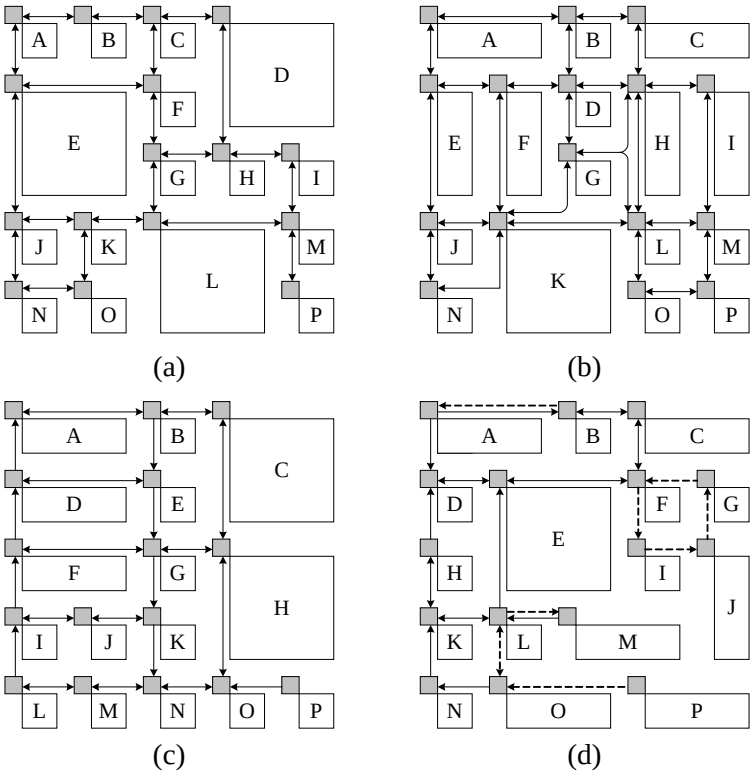


Figure 3.19: Custom NoC networks used in our experiments.

energy than a short link since the long link requires a strong output driver or a number of repeaters. Therefore, α is set to 1/2 when the interconnection matrix of a network is composed.

Table 3.7 shows the application mapping results on the custom networks. A3MAP-SR and A3MAP-GA reduce on average total hop count by 8.1% and 14%, respectively, compared to NMAP. We also measure total wirelength passed by all packets, which is related to communication energy consumption than the total hop count in custom networks. In Table 3.7, total wirelength passed by all packets is reduced on average by 19.1% and 31.2%, respectively, compared to NMAP. CMAP also improves mapping quality on custom networks whereas it has no benefit on irregular mesh networks. However, its hop count is 6.5% and 12.4% greater than those of A3MAP-SR

Total hop count					Total wirelength passed by all packets			
Application	NMAP	CMAP	A3MAP-SR	A3MAP-GA	NMAP	CMAP	A3MAP-SR	A3MAP-GA
Figure 3.19(a)	4488	4752	4531	4087	5879	6300	5332	4543
Figure 3.19(b)	4264	4119	4248	4199	5505	4135	5049	4215
Figure 3.19(c)	6296	5598	5867	5150	7835	6842	7434	5613
Figure 3.19(d)	5524	5735	4263	4263	9196	9627	5170	5170
Average	5143	5051	4727	4425	7104	6726	5746	4885
Ratio	1.000	0.982	0.919	0.860	1.000	0.947	0.809	0.688

Table 3.7: The comparison of hop count and wirelength for VOPD benchmark in custom networks.

and A3MAP-GA, respectively and total wirelength passed by packets is 14.6% and 28.4% longer than those of A3MAP-SR and A3MAP-GA, respectively. These results prove that our weighted interconnection matrix is efficient enough for reducing communication energy consumption since the improvement of wirelength passed by all packets is greater than that of hop count. Therefore, the weighted interconnection matrix is desirable for custom networks. Similarly, the proposed A3MAP algorithms can be easily manageable for more complex NoC by controlling the weighted interconnection matrix.

3.2.4.4 Large-Scale NoC

We prove A3MAP algorithms to be suitable for the partition-based approach which is described in Section 3.2.3. In this experiment, core graphs with one hundred cores are generated by TGFF and mapped to a 10×10 regular network. The cores are partitioned to 9-15 groups with the minimum cuts by hMETIS [37] and the network is also partitioned to 9-15 groups with a convex region. Then, after sorting the groups in a decreasing order by the amount of communication inside each group, A3MAP-FS that provides the best mapping quality, A3MAP-SR, A3MAP-GA and NMAP that provides

one of the fastest solutions, perform application mapping for each ordered core group on a selected convex region, which are called A3MAP-FS-P, A3MAP-SR-P, A3MAP-GA-P and NMAP-P, respectively.

Figure 3.20 shows the hop count comparison of the application mapping algorithms. As the number of groups increases, i.e. the number of cores or tiles included in each group decreases, A3MAP-GA-P and A3MAP-SR-P achieves similar mapping quality to A3MAP-FS. However, total hop count of most partition-based mapping algorithms tends to increase since cores are mapped to a more restricted convex region. On the contrary, if the number of groups decreases, the number of cores included in each group increases. As a result, since cores can be mapped to a larger convex region, most of the application mapping algorithms improves their hop count. The improvement of mapping quality of A3MAP-GA-P and A3MAP-SR-P is less than that of A3MAP-FS whereas it is much greater than that of NMAP-P.

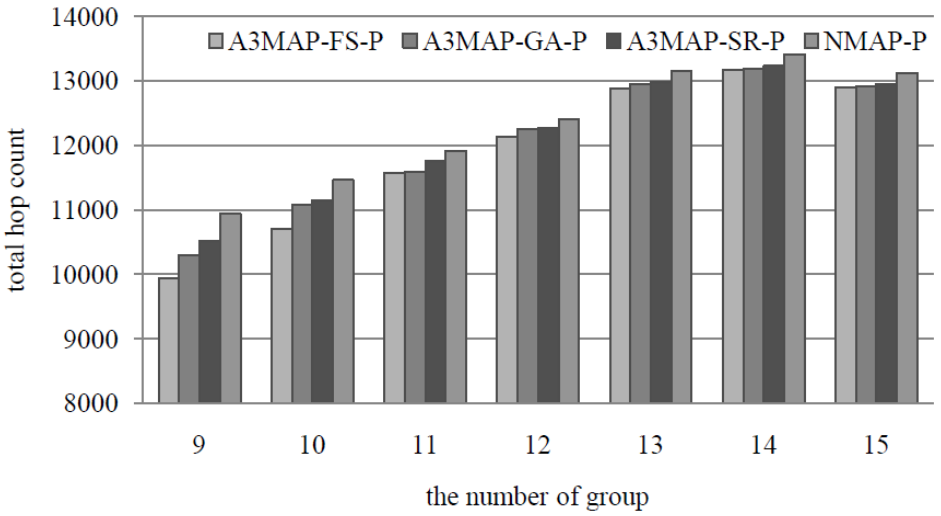


Figure 3.20: The hop count comparison of application mapping algorithms in large networks partitioned to 9-15 subnetworks.

However, since the runtime of A3MAP-FS rapidly gets long in the larger convex region as shown in Figure 3.21, it shows an inefficient trade-off between mapping quality and runtime. The application mapping quality of A3MAP-FS-P can be obtained by A3MAP-GA-P or A3MAP-SR-P if A3MAP-GA-P or A3MAP-SR-P performs in a network that is partitioned to fewer groups. In addition, their runtime is much faster than that of A3MAP-FS-P. For example, the mapping quality of A3MAP-FS-P on a network partitioned to 10 groups is worse than the mapping quality of A3MAP-SR-P and A3MAP-GA-P on a network partitioned to 9 groups in Figure 3.21. In addition, the runtime of A3MAP-SR-P and A3MAP-GA-P on a network partitioned to 9 groups is much faster than that of A3MAP-FS-P on a network partitioned to 10 groups. On the contrary, even though NMAP-P shows slightly faster runtime than A3MAP-SR-P and A3MAP-GA-P in Figure 3.21, its mapping quality is much worse in a network with few groups in Figure 3.20. Therefore, A3MAP algorithms are more suitable for the partition-based approach in large-scale NoC.

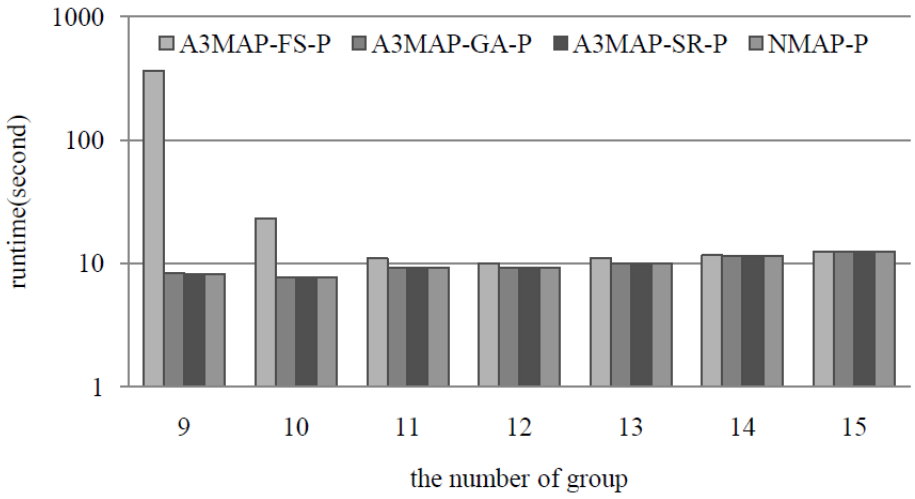


Figure 3.21: The runtime comparison of application mapping algorithms in large networks partitioned to 9-15 subnetworks.

Next, we check how many hop counts the partition-based A3MAP algorithms increase in regular networks, irregular networks and custom networks. We use synthetic benchmarks with 25, 36, 49, 64, 81 and 100 cores. We make each partitioned group not include more than 16 cores such that 25, 36, 49, 64, 81 and 100 cores are partitioned to 2, 3, 4, 4, 6 and 7 core groups, respectively. The groups are not required to include the same number of cores when the cores are partitioned with the minimum cuts. We perform the partition-based A3MAP algorithms ten times with different core graphs and networks.

Figure 3.22 shows the hop count of A3MAP-SR-P normalized by A3MAP-SR in regular networks, irregular networks and custom networks, called A3MAP-SR-P-R, A3MAP-SR-P-I and A3MAP-SR-P-C, respectively. The hop count performed by A3MAP-SR-P slightly increases, compared to A3MAP-SR due to the partitioning process. In addition, the hop count increases in most networks as the number of PEs increases. Consequently, A3MAP-SR-P increases on average total hop count by 1.5%, 2.0% and 2.6% in regular mesh, irregular mesh and custom networks, respectively.

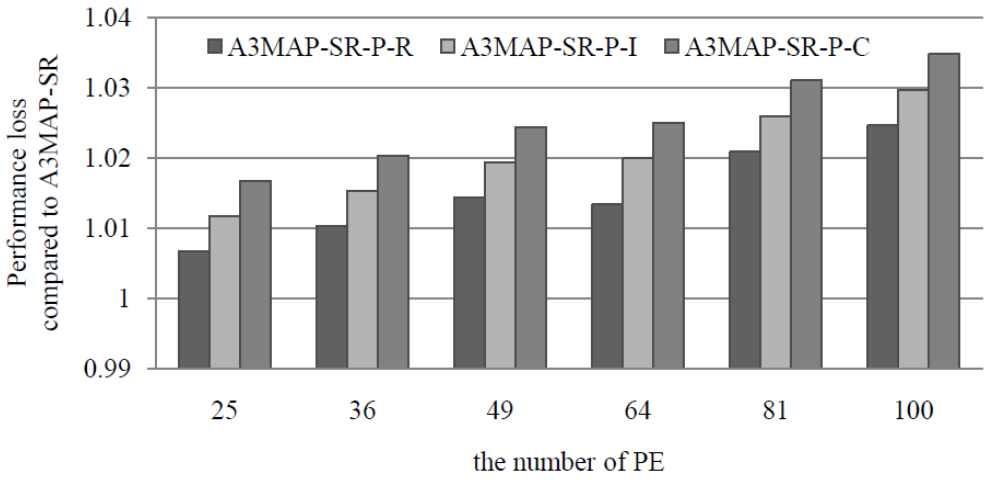


Figure 3.22: The hop count of A3MAP-SR-P normalized by A3MAP-SR on regular mesh, irregular mesh, and custom networks with 25-100 PEs.

Similarly, Figure 3.23 shows the hop count of A3MAP-GA-P normalized by A3MAP-GA in regular mesh, irregular mesh and custom networks, called A3MAP-GA-P-R, A3MAP-GA-P-I and A3MAP-GA-P-C, respectively. A3MAP-GA-P increases on average total hop count by 1.7%, 2.5% and 3.2% in regular mesh, irregular mesh and custom networks, respectively.

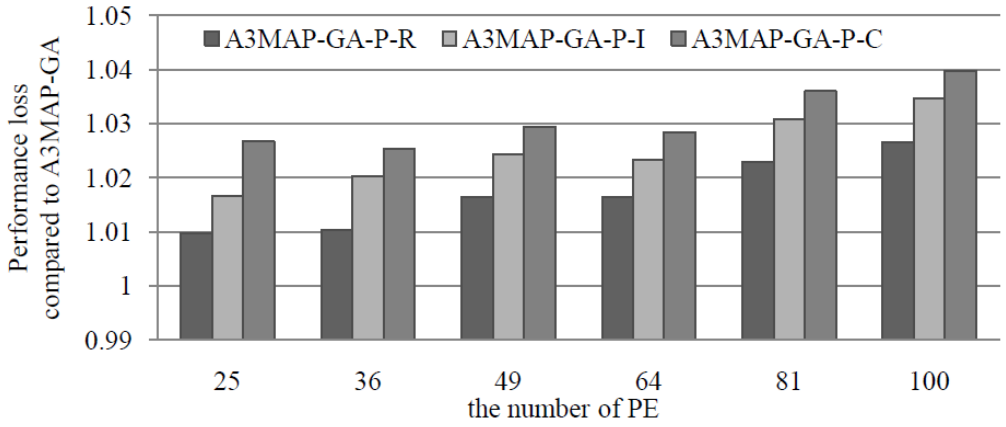


Figure 3.23: The hop count of A3MAP-GA-P normalized by A3MAP-GA on regular mesh, irregular mesh, and custom networks with 25-100 PEs.

Figure 3.24 shows the runtime of A3MAP-GA, A3MAP-SR, A3MAP-GA-P, A3MAP-SR-P and NMAP. A3MAP-SR-P and A3MAP-GA-P show that the increases of their runtime are slower than others as the number of PEs increases. As a result, when the number of PEs is more than 60, they are the fastest even if their runtimes in 25 PEs are similar to A3MAP-SR. Therefore, the A3MAP algorithms are more suitable for the partition-based approach in large-scale NoCs since they provide an efficient trade-off between runtime and mapping quality.

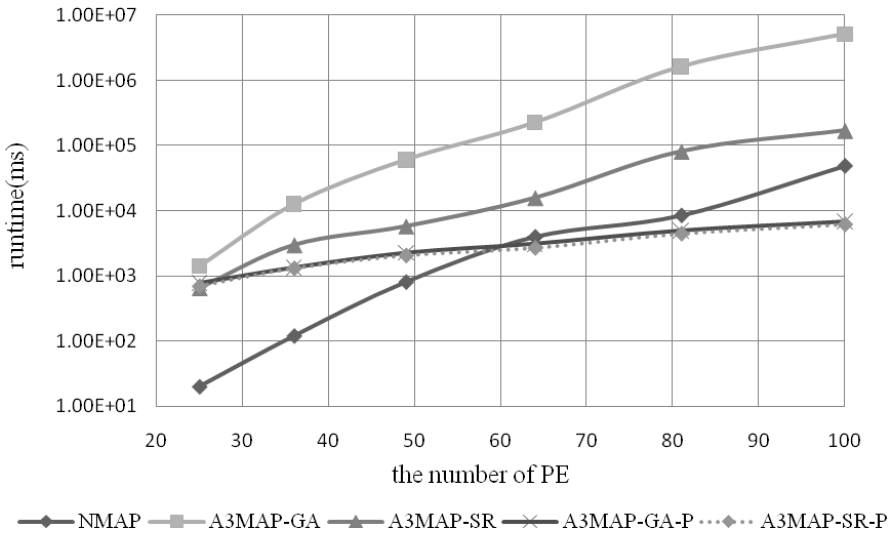


Figure 3.24: The runtime comparison of NMAP, A3MAP-GA, A3MAP-SR, A3MAP-GA-P, and A3MAP-SR-P.

3.2.5 Summary

In this section, we propose novel and global architecture-aware application mapping (A3MAP) algorithms for NoC. Based on a metric embedding technique, we analytically formulate an application mapping problem to MIQP. Then, the MIQP is solved by two effective heuristics, i.e. a successive relaxation algorithm providing short runtime and a genetic algorithm providing high mapping quality. In addition, we propose the partition-based approach for large-scale NoCs, where A3MAP algorithms provide an efficient trade-off between runtime and mapping quality. Experimental results show that our A3MAP algorithms greatly reduce hop count on various networks, compared to the previous state-of-the-art works. Especially, A3MAP algorithms show more merits on irregular mesh and custom networks. All networks can be easily converted to the simple but efficient interconnection matrix such that our A3MAP algorithms have no limitation

to map cores to tiles on any arbitrary, faulty and degraded network. Furthermore, A3MAP algorithms are easily manageable for low communication energy consumption and high performance by an architecture-aware analytical manner.

Chapter 4

NoC Architecture and Physical Design for Emerging Technologies

The architecture and circuit techniques for NoC should be compatible with physical design and design for manufacturability (DFM) constraints since network designs are subject to technology constraints. With aggressive scaling transistor and wire dimensions, variability and reliability have become important for NoC designs. Fast computation relative to communication time motivates more intelligent routing algorithms designed to minimize hop count and network congestion. This trend indicates a need for research in technology-driven and scalable router, switch, and link designs. Moreover, as emerging technologies, such as 3D die integration and on-chip optical/wireless communication become viable, new opportunities and constraints will further drive the need for innovation in interconnection networks. In particular, as 3D die integration based on through-silicon vias (TSVs) becomes feasible, a 3D NoC design brings in new challenges. Since 3D NoC must satisfy not only application constraints such as latency, throughput, and power, but also manufacturing/design constraints imposed by 3D technologies such as the number of TSV, chemical-mechanical polishing (CMP), TSV stress, and thermal effect, a 3D NoC design compatible with the constraints is required for innovation in interconnection networks.

4.1 CMP-Aware Application-Specific 3D NoC Design

As shrinking the horizontal feature size has critical limitations, vertically stacking silicon based on TSVs has gained tremendous interests from both academia and industry for the future integrated circuits (ICs). NoC is an effective solution for scalable on-chip communication in the complex three-dimensional (3D) interconnections since it can

control the number of TSVs necessary for various applications. However, 3D NoC must meet not only application performance/power constraints, but also manufacturing constraints imposed by the 3D technologies. Therefore, the combination of the 3D technologies and the NoC offers new challenges and opportunities.

So far, many researchers have addressed the issues of 3D floorplanning and NoC topology generation with consideration of thermal hot spots. For example, in 3D floorplanning, cores with high power density can be assigned to the silicon layer attached to heat sinks and spread out at each silicon layer to reduce peak temperature and help mitigate the thermal and reliability problems such as electromigration, stress, dielectric breakdown, leakage-thermal run-away, and speed of devices [17][43]. Based on 3D thermal-aware floorplanning, 3D NoC topology is then synthesized [80][102][120].

Besides thermal and related thermal-mechanic stress effects [3][121], 3D-IC integration has other manufacturability and layout related challenges related with TSVs and landing pads [68][91]. One particular challenge is that the wide range of the metal area by TSVs and landing pads increases non-uniform metal density distribution, and thus results in the critical variation of wire thickness and TSV height during the CMP process [30][70][104]. The CMP processes in 3D-IC are used for both Cu-CMP (for the removal of extra Cu on silicon after filling TSVs with Cu or depositing Cu on TSV landing pads) and silicon-CMP (for silicon backside thinning). The uneven Cu-wire thickness changes wire resistance and coupling capacitance between wires, and thus results in critical timing variation. Moreover, the uneven TSV height leads to bonding failure between TSVs and landing pads. To mitigate the non-uniform metal density, dummy metal fill insertion can be used in empty spaces, but that may affect RC parasitics [56][57]. Dummy TSV insertion can be also inserted for reducing silicon-CMP variation, but it may significantly reduce usable silicon area of the entire chip. Since TSV height

variation after silicon-CMP strongly depends on the regularity and density of TSV distributions, 3D NoC designs with different vertical links composed of tens to hundreds of TSVs should consider the TSV height variation.

In this section, we propose the first CMP-aware application-specific 3D NoC design that minimizes TSV height variation, thus reduces bonding failure, and meanwhile optimizes conventional NoC design objectives such as hop count, wirelength, power consumption, and area. For NoC vertical links composed of tens to hundreds of TSVs, the layout of each individual TSV is not efficient since it results in complex global routing and TSV manufacturing stresses affect more transistors [3][121]. Therefore, TSVs should be placed as an array type in 3D NoC. However, the array with dense TSVs is sensitive to CMP process which results in high TSV height variation, and thus leads to severe bonding failure. Moreover, if the arrays with different TSV density are used in the same layer, bonding TSVs on landing pads is more difficult. NoC includes one-way and two-way links of which the metal densities may be different. Therefore, TSVs in an array should be placed with a pitch resulting in low TSV height variation endured by a bonding technique and TSV arrays with the same density should be inserted in each layer. In addition, previous 3D NoC designs cannot handle TSV arrays during placement and routing stage since the size of the TSV arrays is too large [80][102][121]. Therefore, TSV arrays should be handled during the floorplanning stage in physical design. Based on these motivations, the major contributions of this work include:

- We show that TSV height variation during silicon-CMP process is more severe in 3D NoC where dense TSV arrays are used as vertical links.
- We propose a CMP-aware application-specific 3D NoC design that minimizes TSV height variation and optimizes conventional NoC design objectives.

- We present CMP-aware 3D NoC techniques for core-to-layer assignment, topology synthesis, and floorplanning.
- We show that the proposed 3D NoC design reduces TSV height variation with lower design cost, and meanwhile achieves less hop count, wirelength, and power consumption.

To the best of our knowledge, this is the first work that addresses CMP variation in 3D NoC. The rest of this section is organized as follows: Section 4.1.1 introduces CMP and Cu-Cu thermo-compression direct bonding, and then addresses various TSV layouts and their CMP variation. Section 4.1.2 shows the problem formulation and proposed CMP-aware application-specific 3D NoC design flow. Section 4.1.3 presents detailed techniques of our proposed algorithms. Section 4.1.4 shows experiment results and Section 4.1.5 concludes the section.

4.1.1 Preliminaries

4.1.1.1 Chemical-Mechanical Polishing and Cu-Cu Thermo-Compression Direct Bonding

One of the most potential sources of yield loss and timing variation in 3D technologies is TSV bonding on land pads. In a typical industrial bonding procedure [108][110], a TSV-wafer is ground down to a target thickness slightly above the TSV depth (keeping TSVs unexposed) and further thinned using CMP process. CMP uses both chemical and mechanical means to polish the surface of the wafer. In a typical rotary

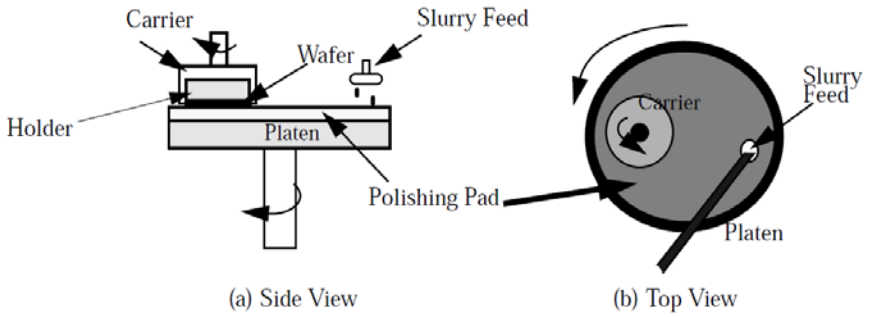
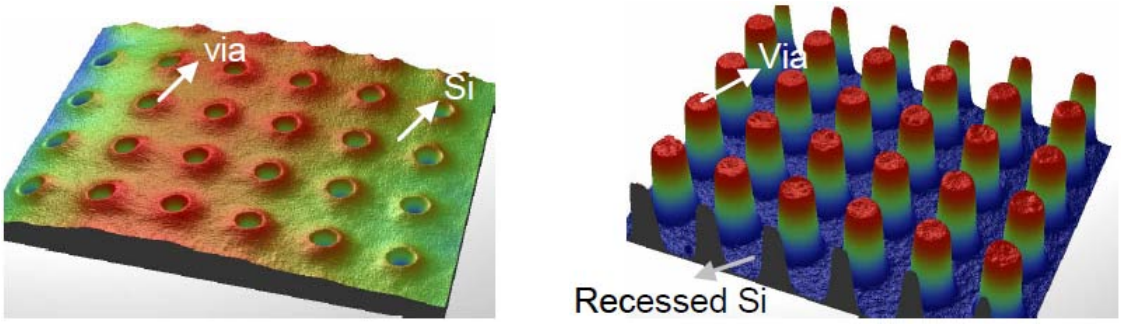


Figure 4.1: Typical rotary CMP tool [30].

CMP tool, the wafer is held on a rotating holder, as shown in Figure 4.1. The surface of the wafer being polished is pressed against the polishing pad which is mounted on a rotating disk. A slurry composed of particles suspended in a chemical solution is also deposited on the pad as the chemical abrasive. The material-removal mechanism of CMP is similar to the removal found in glass polishing. A chemical reaction softens the surface of a material to be removed later. The chemical reaction creates a hydroxilated-form material which has weaker atomic bonds. It is, therefore, more easily removed during the polishing process. Then, a mechanical surface abrasion aided by slurry particles removes the material. Figure 4.2(a) shows the uneven surface of wafer backside after grinding and CMP. Subsequently, the polished silicon surface is plasma-etched, such that the TSVs protrude from the wafer as shown in Figure 4.2(b). On the contrary, TSV Landing pads are commonly fabricated on the top metal layer in a damascene process and designed to be larger than TSVs to prevent overlay error. The top metal layer with TSV land pads is also polished by CMP to remove overburden Cu. Finally, a wafer or die with TSV landing pads is bonded with a different wafer or die with TSVs.



(a) After grinding and CMP (b) Si-recess etch following CMP

Figure 4.2: Local topography on backside of wafer [108].

Recently, micro-bump-less Cu-Cu direct bonding techniques attract great attentions in 3D die integration since the same bonding medium can prevent the formation of an intermetallic at the interface between TSVs and landing pads [108][110]. In addition, Cu-Cu direct bonding is desired compared to solder-based connections since (1) Cu-Cu bond is more scalable and ultra-fine pitch can be achieved; (2) Cu has better electrical and thermal conductivities; and (3) Cu has much better electromigration resistance for higher current density. The direct Cu-Cu bonding has been demonstrated using thermo-compression bonding via parallel application of heat and pressure (typically ~300-400^{o}C and ~200 kPa). The bonding mechanism is based on interdiffusion of Cu atoms and grain growth, and hence it is also widely known as diffusion bonding.

4.1.1.2 TSV Layouts and CMP Variation

The goal in Cu-CMP is to polish the barrier and remove overburden Cu on silicon after filling TSVs with Cu and depositing Cu for landing pads. Cu-CMP involves simultaneous polishing of three materials: Cu, dielectric (oxide), and barrier (Tan, Ti, etc.). However, due to different chemical effects on the materials and pattern differences

in terms of pattern density, Cu line width, and oxide line space, the removal rates of the these materials are different. Their difference in removal rates results in different polish times across the wafer. For example, in Figure 4.3(a), by the time the excess Cu and barrier on TSVs used for a 64-bit link are cleared at a point on the die, those on TSVs used for 128-bit links might at another point have already been cleared, where chemicals used for Cu-CMP react well on Cu rather than silicon. Hence, either the 128-bit TSVs are overpolished at the time the excess Cu and barrier on the 64-bit TSVs are cleared or the excess Cu and barrier on the 64-bit TSVs are not cleared at the time the excess Cu and barrier on TSVs used for a 128-bit link are cleared. Figure 4.3(a) shows that the 128-bit TSVs are overpolished after Cu-CMP. The uneven polishing problem in Cu-CMP can be solved by CMP fill synthesis where dummy metals grounded or floating are inserted in the empty spaces of a metal layer [56][57].

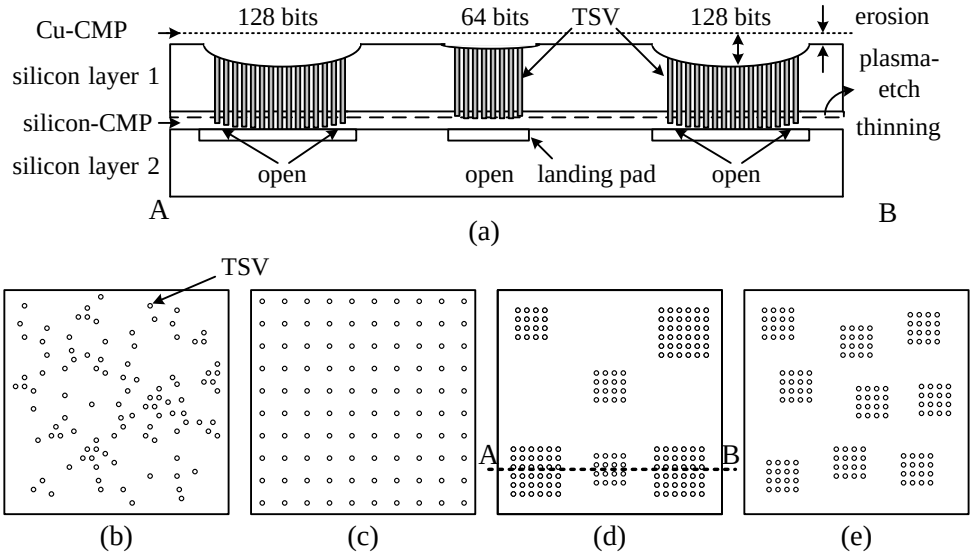


Figure 4.3: TSV layouts and their TSV height variation induced by a CMP process.

Silicon-CMP is used for finely thinning silicon after grinding silicon backside since the processing time of CMP is too long. As silicon-CMP also involves simultaneous polishing of silicon, Cu, and barrier, their removal rates are different according to different chemical effects on the materials and different TSV densities in terms of TSV diameter, TSV pitch, and TSV array size. The different removal rates of these materials results in different polish times across the wafer backside. For example, in Figure 4.3(a), by the time the silicon and barrier under TSVs used for a 64-bit link are cleared at a point, the silicon and barrier under TSVs used for 128-bit links might have been not cleared yet, where chemicals used for silicon-CMP react well on silicon rather than Cu. Hence, either the silicon and barrier under the 128-bit TSVs are underpolished at the time the silicon and barrier under the 64-bit TSVs are cleared or the 64-bit TSVs are overpolished at the time the silicon and barrier under the 128-bit TSVs are cleared. Figure 4.3(a) shows the 128-bit TSVs are underpolished after silicon-CMP. In [108], IMEC TSV technology showed that within-die thickness variation after silicon-CMP was 1.5μm for a die size of $10.6\times10.6mm^2$ when TSVs of which the diameter, pitch, and density are 5μm, 10μm, and $10k/mm^2$, respectively, were evenly distributed over the whole chip as shown in Figure 4.3(a). The within-die thickness variation is more sensitive to irregular and high TSV density and directly related to TSV height variation. Consequently, the uneven TSV height variation can induce severe TSV bonding failure as shown in Figure 4.3(a). In particular, the bonding failure will be more severe in Cu-Cu direct thermo-compression bonding widely used in 3D-IC since TSVs must be directly contacted to landing pads without any micro-bump. Unlike Cu-CMP, CMP fill synthesis is not an efficient solution for silicon-CMP since dummy TSV insertion would significantly increases the overall chip area.

TSVs can be placed with different schemes during placement and routing [3][68][91]. If TSVs are laid out without any constraints imposed by 3D technology, they can be distributed as shown in Figure 4.3(b). Whereas such layout achieves much shorter wirelength, TSV height variation induced by silicon-CMP greatly increases due to uneven TSV density. In Figure 4.3(c), TSVs are placed with globally uniform density distributions. The TSV distribution provides the least TSV height variation to 3D ICs. However, such TSV layout is not suitable for NoC vertical links composed of tens to hundreds of TSVs since it results in so complex global routing that any wire in the same vertical link may detour with a long path. The long wires detoured makes system performance degraded or timing closure difficult. In addition, the layout of each individual TSV causes manufacturing stresses to more devices [3][121]. Therefore, grouping TSVs to an array and then laying out the array is more desirable for 3D NoC.

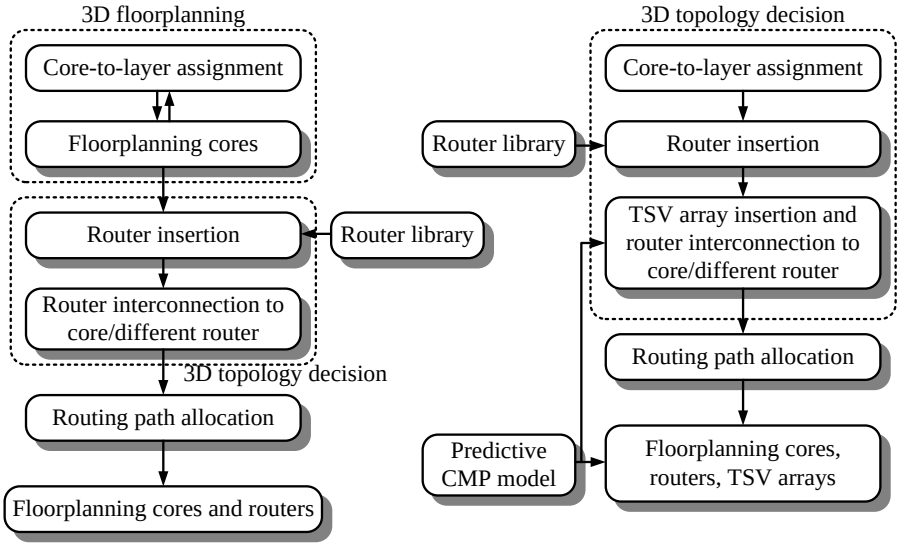
In Figure 4.3(d), there exist two kinds of TSV arrays. The small array includes one one-way NoC link and the large array includes one two-way NoC link which has two times more TSVs than the one-way NoC link. TSVs in the small array fail to contact landing pads since TSVs in the large array are less cleared than those in the small array during silicon-CMP such that the surface in a die is uneven. In Figure 4.3(a) that is the cross section of AB in Figure 4.3(d), the 64-bit TSV array has the strong possibility of failing to contact landing pads on silicon layer 2 since the 128-bit TSV array is underpolished. In addition, since the metal density of the 128-bit TSV is high, its own silicon-CMP variation can be so high that TSVs in the array have the possibility of failing to contact landing pads. We can control the local TSV density defined as the size of a TSV array divided by a TSV pitch. If the 128-bit TSV array has a wider TSV pitch, its density can be as low as that of the 64-bit TSV array. However, since it has the penalty of area, we focus on reducing the size of a TSV array as shown in Figure 4.3(e).

4.1.2 CMP-Aware NoC Design Flow and Problem Formulation

In most previous application-specific 3D NoC designs [80][102][120], 3D floorplanning is first performed and then a 3D network topology is determined, based on the 3D floorplanning as shown in Figure 4.4(a), where their 3D technology constraint is just the number of allowable TSVs. The constraint is not sufficient for robust and reliable 3D ICs and the CMP variation resulting in severe bonding failure is not considered. In addition, since the 3D floorplanning composed of assigning cores to layers and floorplanning the cores in each layer is first performed without any routers and TSV arrays, there may be no enough dead space where the routers can be physically placed after deciding a 3D network topology [80][102]. The area of the latest routers is no longer small since its complexity rapidly increases due to a virtual channel, a complex flow controller, and an adaptive routing path allocator. In order to prevent overlapping routers inserted and cores already floorplanned, additional floorplanning is performed in each layer after deciding a network topology [120]. However, such 3D NoC design flow is not efficient for reducing wirelength, hop count, and thus energy consumption as the routers gets more and more complex. In addition, the previous 3D NoC designs have not considered the layout of TSVs since they assume that TSVs are laid out during placement and routing. Furthermore, the layout of each individual TSV without considering NoC architecture in the placement and routing stage worsens CMP variation, complex global routing, and manufacturing stress to more devices. Finally, since TSV arrays are much larger than other placement objects, it is not efficient that they are considered in the placement and routing stage.

Figure 4.4(b) shows the proposed CMP-aware NoC design flow covering such issues. We first assign *n* cores to *k* layers with the purpose of reducing communication between layers under a given area constraint and thus using few TSVs. Based on the

cores assigned to each layer, the number of allowable routers is inserted in each layer and then routers are interconnected to cores and different routers in the same layer, where the number of interconnecting the single router to cores and other routers is limited. The goal of our network topology decision in each layer is to minimize hop count with limited network resources. Then, routers are interconnected to different routers in adjacent layers by only one-way vertical links. That is, any routers in different layers are not interconnected by two-way vertical links. Using only one-way links over the whole chip makes different layers interconnected with uniform and low local TSV density. Since the number of allowable TSVs between layers is also limited by a given area constraint, vertical interconnections minimizing the total hop count are selected. Then, routing paths without deadlock and livelock are allocated on the existing interconnections. We compute a TSV pitch applied in the one-way link and then a TSV array is composed. Finally, all cores, routers, and TSV arrays are simultaneously floorplanned in each layer.



(a) Conventional 3D NoC design flow (b) CMP-aware 3D NoC design flow

Figure 4.4: The conventional and proposed 3D NoC design flows.

We start to solve the CMP-aware application-specific 3D NoC issues from a core graph. A graph *G(V,E)* with *n* vertices is a directed graph, where each vertex $v_i \in V$ represents a core, a router, a TSV array and each directed edge $e_{i,j} \in E$ represents communication relation between v_i and v_j. $vol(e_{i,j})$ represents communication volume between v_i and v_j and $wl(e_{i,j})$ represents wirelengh between v_i and v_j.

4.1.2.1 Core-to-Layer Assignment

Core-to-layer assignment allows cores to move from continuous space to discrete space, forcing each core to exactly occupy one layer. That is, a set of cores $V=\{v_1, v_2, \ldots, v_n\}$ is assigned to *k* layers $L=\{l_1, l_2, \ldots, l_k\}$, and thus $V=\{V^{l_1}, V^{l_2}, \ldots, V^{l_k}\}$ is obtained, where $V^{li}=\{v_1^{li}, v_2^{li}, \ldots, v_j^{li}\}$, where $j<n$. The area of cores is represented as $\{A_1, A_2, \ldots, A_n\}$. To equally assign the area of cores to layers, an area constraint is defined as:

$$\alpha_{\min} \sum_{i=1}^{n} \frac{A_i}{k} < A_l < \alpha_{\max} \sum_{i=1}^{n} \frac{A_i}{k} \tag{4.1}$$

where α_{min} and α_{max} are acceptable minimum and maximum area coefficients (α_{min} <1< α_{max}). We consider thermal hot spots in this step, using the thermal model proposed in [17]. The thermal model makes a high power density core assigned to a lower silicon layer that is attached to a heat sink. With the area constraint, the objective of our layer assignment is to minimize communication between different layers and temperature as follows:

$$\min\left[\beta_1 \sum vol\left(e_{i,j}\right)\cdot|u-v| + \beta_2\left\{\sum_{p=1}^{k}\left(P_p\sum_{q=1}^{p}R_q\right)+R_b\sum_{p=1}^{k}P_p\right\}\right] \tag{4.2}$$
$$\text{s.t. } \forall v_i \in V^{l_u}, \forall v_j \in V^{l_v}\,(u \neq v)$$

where β_1 and β_2 are weighting coefficients, R_q is a thermal resistor in layer q, P_p is the sum of current source in layer p, and R_b is the thermal resistor of the bottom layer material.

4.1.2.2 3D NoC Topology Decision and Routing Path Allocation

Given the number of allowable routers and TSV arrays in each layer, the number of allowable cores, and the number of different routers interconnected to one router, we interconnect routers to cores and different routers in the same layer. A router communication graph *RCG(R,C)* with *m* vertices is a directed graph, where each vertex $r_i \in R$ represents a router, and each directed edge $c_{i,j} \in C$ represents communication between r_i and r_j. The objective of our topology decision in each layer is as follows:

$$\begin{aligned} &\min\left[\sum vol\left(e_{i,j}\right)\cdot dist\left(M\left(v_i\right),M\left(v_j\right)\right)\right] \\ &\text{s.t. } bw\left(link\left(M\left(v_i\right),M\left(v_j\right)\right)\right)\geq vol\left(e_{i,j}\right),\left(\forall v_i,\forall v_j\right)\in V^{l_u} \end{aligned} \tag{4.3}$$

where $dist(r_p,r_q)$ is distance (or hop count) between r_p and r_q and $M()$ is a core-to-router mapping function, e.g. $r_p=M(v_i)$ and $r_q=M(v_j)$. $link(r_p,r_q)$ is all links which any packet in r_p passes for reaching r_q. Then, we interconnect routers in adjacent layers, based on the RCG graphs. The objective of our topology decision among layers is as follows:

$$\begin{aligned} &\min\left[\sum vol\left(e_{i,j}\right)\cdot dist\left(r_p,r_q\right)\right] \\ &\text{s.t. } link_{TSV}\left(r_p,r_q\right)\neq link_{TSV}\left(r_q,r_p\right), \\ &bw\left(link\left(r_p,r_q\right)\right)\geq vol\left(e_{i,j}\right),\forall v_i\in V^{l_u},\forall v_j\in V^{l_v}\left(u\neq v\right) \end{aligned} \tag{4.4}$$

where $link_{TSV}(r_p,r_q) \in link(r_p,r_q)$ is a vertical link which any packet in r_p passes for reaching r_q. This equation indicates that routers in different layers are interconnected by only one-

way links. Thus, CMP variation resulting in uneven TSV heights can be greatly reduced and the yield of TSV bonding can be greatly improved.

4.1.2.3 Floorplanning

We compute a TSV pitch where a boding technique used can endure TSV height variation in the number of TSVs covering a one-way vertical link, based on our predictive CMP model. Then a TSV array is composed and inserted between routers in adjacent layers. As the inputs of our floorplanner, we take a set of cores, routers, and TSV arrays, $\{v_1, v_2, \ldots, v_n\}$. v_i is a $W_i \times H_i$ rectangle and aspect ratio H_i/W_i. Each block can be free to rotate and change the aspect ratio continuously in a given range $[AR_{min,i}, AR_{max,i}]$. A floorplan F is the assignment of (x_i, y_i) for each block v_i without any overlap of all cores, routers and TSV arrays, where half-perimeter wire length (HPWL) estimation is used. We consider thermal hot spots, using the thermal model proposed in [17]. The thermal model minimizes the maximum temperature difference in the same layer. Therefore, the objective of our floorplan F is as follows:

$$\min \left[\begin{array}{l} \alpha_2 \sum A_i + \beta_2 \sum \left(wl\left(e_{p,q}\right) \times vol\left(e_{p,q}\right)\right) \\ +\gamma_2 \left(\max\left(T\left(m,n,l_j\right)\right) - \min\left(T\left(m,n,l_j\right)\right)\right) \end{array} \right] \quad s.t.\ \forall v_i \in l_j, \forall v_p \in l_j, \forall v_q \in l_j \tag{4.5}$$

where γ_1, γ_2, and γ_3 are weighting factors. $T(x,y,l_u)$ is the temperature of a tile in x, y, and l_u at *x-axis*, *y-axis*, and *layer*, respectively and th_w is the maximum allowable wirelength.

4.1.3 CMP-Aware 3D NoC Design

4.1.3.1 CMP-Aware Core-to-Layer Assignment

Since the number of TSVs required depends on communication volume between different layers, the communication volume should be minimized with thermal consideration. In addition, the area of each layer should meet the area constraint, Eq. (4.1).

Figure 4.5 shows two core-to-layer approaches where eight cores are assigned to four layers. Let a core graph given as shown in Figure 4.5(a) where all edges have the same weight, all cores have the same power density, and the number is the area of a core for simple explanation.

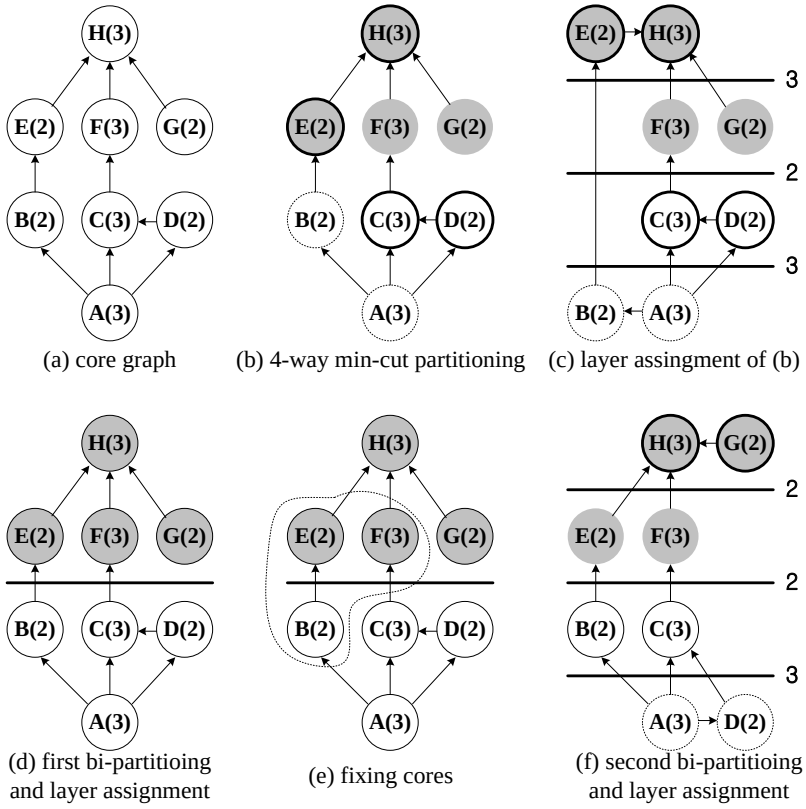


Figure 4.5: Examples of assigning eight cores to four layers.

The first approach is that 4-way minimum-cut area-balanced partitioning is performed and then the partitioned subgroups are one-to-one assigned to different layers. For example, in Figure 4.5(b), the cores are partitioned to {A, B}, {C, D}, {F, G}, and {E, H} that have the same area and the minimum cuts. Then, the partitioned subgroups are one-to-one assigned to any layers, achieving the minimum hops as shown in Figure 4.5(c).

The second approach we propose in Algorithm 9 recursively performs area-balanced bi-partitioning with the minimum cost computed from Eq. (4.2). Figure 4.5(d) shows the result of the first bi-partitioning where the same area and the minimum cut are obtained (line 2). Then, any core which communicates other cores in a different layer is assigned in advance, depending on their communication gain as shown in Figure 4.5(e) (line 5). The communication gain is computed as the subtraction of the amount of intra-layer communication from that of inter-layer communication. If the communication gain of any core is greater or equal to 0, the core is assigned to a current layer. In Figure 4.5(e), core B, C, E, and F communicate cores in a different layer and their communication gains are 0, -1, 0, and 0, respectively. Thus, core B, E, and F are assigned to a current layer. Then, the second bi-partitioning in each sub-group is again performed for the minimum cut under the area constraint. Figure 4.5(f) shows the final result where hop count between four layers is 7 whereas the hop count of the first approach between four layers is 8 in Figure 4.5(c). Therefore, the second one is useful to reduce hop count between layers, thus requires less TSVs.

Even if the number of a given layer is not a power of two, the basic idea of Algorithm 9 can be easily extended. For example, let eight cores assigned to five layers in Figure 4.5(a), where total area is 20. When the first bi-partitioning is performed, both partitions do not get the same area, but one gets 8 and the other gets 12. Then, the first

group with area 8 is again bi-partitioned for the minimum cut and same area. On the contrary, the second group with area 12 is also bi-partitioned for the minimum cut but different areas (where one is 4 and the other is 8). Finally, the last sub-group with area 8 is again bi-partitioned for the minimum cut and same area. Finally, all five sub-groups get area 4.

Algorithm 9 Core-to-Layer Assignment by Recursive Bi-Partitioning

```
1: while the number of partitioned layers is not equal to the target number of layers
   do
2:     Find bi-partitions of cores with min. cost computed by Eq. (4.2);
3:     Compute communication gain (CG_i) of core i in layer k;
4:     if CG_i ≥0 then
5:         Core i is assigned to layer k;
6:     end if
7: end while
```

4.1.3.2 CMP-Aware 3D NoC Topology Decision

Since a 3D network topology decision problem is NP-Hard [92], we present efficient heuristics in this section. Furthermore, since the integrated problem makes it difficult to reach guaranteed quality bounds on the solution, we divide the 3D network topology decision problem into two distinct subproblems, called router-to-core/router interconnection in the same layer and router-to-router interconnection between different layers, and then we solve the respective subproblems. Whereas a bandwidth requirement can be easily satisfied by finding alternative routing paths or adding more interconnection resources, satisfying latency constraints is difficult if cores communicating each other are too wide apart. Therefore, any master core sensitive to latency should be interconnected to the same router as its slave core. A TSV array covering a one-way vertical link is used for interconnection between different layers and any router is not interconnected to

routers in a different layer if it is already interconnected to the router with one direction as shown in Eq. (4.4), which minimizes TSV density variation, thus reduces TSV height variation resulting in TSV bonding failure.

1. 2D Router-to-router/core interconnection

Given a core graph, the number of allowable routers *(max_router),* and the number of allowable interconnection to a router *(max_int),* our 2D network topology synthesis approach interconnects possible cores to routers. The objective of our 2D network topology decision is to minimize power consumption in each layer. Varying the number of routers in NoC designs has a great impact on power consumption and communication latency. NoC using few routers leads to longer core-to-router interconnections and hence, higher interconnection power consumption. On the contrary, when a number of routers are used, data flows have to traverse more routers, leading to high router power consumption and increasing area. Thus, we need to explore NoC designs with the different number of routers to obtain the best solution, starting from a design point where each core is interconnected to the minimum routers to one where cores are connected to the maximum allowable routers (*max_router*) in each layer. For example, we assume that there are 20 cores within any layer, the maximum number of allowable routers *(max_router)* is 6 and the maximum allowable interconnection to one router *(max_int)* is 5. We explore a 2D network topology with 4 (equal to *the number of core/max_int*) to 6 routers.

The objective of Algorithm 10 is to establish efficient physical links between a router and a router/core in each layer. First, *i*-way minimum-cut partitioning is performed for cores in the same layer under the *max_int* constraint (line 2) and then each group is assigned to one router (line 3). Next, network links between the routers are inserted

according to user's design objective (line 4). In our implementation, we use the minimum spanning tree (MST) or point-to-point (P2P) interconnection. MST requires distance information between routers. However, since floorplanning is not performed yet, we use different metrics instead of the distance information. MST first interconnects two vertices close to each other. Similarly, since two routers, r_p and r_q which heavily communicates each other should be interconnected with high priority, we use $1/vol(c_{p,q})$ as the distance information. Then, the breadth-first-search or depth-first algorithms are used for seraching MST. MST requiring only *i-1* links decreases total wirelength but increases hop count, where *i* is the number of routers. On the contrary, P2P decreases hop count but increases total wirelength. Next, a new router communication graph (RCG) is generated and then prohibited turn set for RCG is build to avoid deadlocks (line 5-6). Based on the inserted links, paths for flows across different routers in the same layer are allocated, using Dijkstra's shortest path algorithm (line 7). Application constraints such as hop count, communication latency, and bandwidth are evaluated (line 8). If they are not satisfied, a different network topology in each layer is again synthesized (line 9). Finally, the best network topology and design point are selected (line 11).

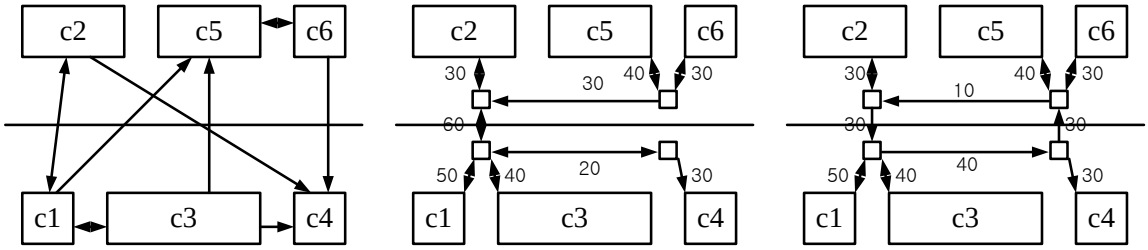
Algorithm 10 2D NoC Topology Decision

```
1:  for i= max_router to (the number of core/max_int) do
2:      Find i-way min-cut partitions under max_int constraints;
3:      Assign each group to one router;
4:      Interconnect router to router by user's design objective;
5:      Build router communication graph (RCG);
6:      Build prohibited turn set for RCG to avoid deadlocks;
7:      Find paths for flows across routers in the same layer;
8:      Evaluate the average and peak hop count;
9:      Repeat step 6 and 7 until application constraints are satisfied;
10: end for
11: Choose the best topology and design point;
```

2. Layer-to-layer interconnection

After deciding a 2D network topology in all layers, any layer must be interconnected to adjacent layers, using TSV arrays. In section 4.1.3.1, we already minimized hop count between different layers, which made few TSVs used. However, due to the few TSVs, total hop count may increase according to the location of the TSVs. In addition, inserting either both one-way and two-way links in the same layer or a TSV array with high metal density results in severe TSV height variation during CMP. Thus, the objective of our layer-to-layer interconnection is to insert one-way links between layers for uniform TSV distribution and the minimum hop count under performance constraints.

Figure 4.6(a) is a core graph assigned to two layers, where the weight of all edges is 1. After deciding the network topology in each layer, let TSV arrays inserted for the minimum hop count as shown in Figure 4.6(b) and (c), where one two-way link and two one-way links are used, respectively. If the industrial open core protocol (OCP) [86] and AMBA advanced extensible interface (AXI) protocol [2] which have been widely used for a network interface have a 32-bit data bus and a 32-bit address, the number of TSV required for a one-way vertical link is 113 and 204, respectively. Thus, the number of TSV required for a two-way vertical link is 226 and 408 in the OCP and AMBA AXI protocol, respectively. As a result, TSV height variation during CMP is more critical in the two-way link with higher metal density. In addition, if another one-way link is inserted in Figure 4.6(b), TSVs array for the link may be open as shown in Figure 4.3(a). Therefore, Figure 4.6(c) is more desirable for low and uniform local TSV density if total hop count of case 2 are similar to that of case 1. In our technique, if a one-way vertical link for $c_{p,q}$ is established, the opposite one-way link for $c_{q,p}$ is removed in the list of TSV array insertion candidates, where $r_p \in V^m$ and $r_q \in V^n$, $(m \neq n)$.



(a) layer-assigned core graph (b) case 1: routing path (b) case 2: routing path

Figure 4.6: CMP-aware router-to-router interconnections in adjacent layers.

4.1.3.3 CMP-Aware Floorplanning

We first compute a TSV pitch for one-way links, based on our predictive CMP model. The pitch must result in low TSV height variation endured by a bonding technique. Then, the TSV array is build and then simultaneously floorplanned with routers and cores in each layer. The goal of our floorplanning is to generate the layout that minimizes area, power consumption, and peak temperature. We modify an existing floorplanning technique [35] and invoke it with our unique cost function.

The power consumption on the given network architecture can be presented as the power required by point to point physical links (core-to-router or router-to-router). It is desirable to place cores, routers, and TSV arrays close to each other if they heavily communicate one another. This is because the power consumption of NoC is directly proportional to the number of hop and the length of wire. Hence, we define the cost function as the product of communication volume $vol(e_{i,j})$ and wirelength $w_{i,j}$ in Eq. (4.5). In addition, it is necessary to place cores, routers, and TSV arrays communicating within the maximum allowed wirelength to operate cores at a given clock speed. We start floorplanning from the top layer with TSV. After floorplanning each layer, terminals with zero area are inserted at the same XY location of the next layer floorplanned as that of

TSV. Since TSV arrays in each layer have no relation one another, they are floorplanned wide apart.

4.1.4 Experimental Results

4.1.4.1 TSV Density and Predictive CMP Model

CMP is a complex process with a large number of input variables including slurry flow rate, pressure, velocity, friction force, lubrication, pad, and wafer geometry and output variables including polish rate, planarization rate, polish rate uniformity, and surface quality. While there are some researches on modeling the CMP variation [30][70], there is very little study on the 3D TSV CMP modeling. Figure 4.7 shows TSV heights measured from the latest 3D ICs of IMEC after silicon-CMP, where the TSV diameter is 5μm [46]. With these industry measurement data, we model TSV height variation as follows:

$$hv = 0.8017 \ln\left(\frac{s}{p}\right) + 1.226 \tag{4.6}$$

where *hv* is TSV height variation, *s* is the size of TSV array, and *p* is a TSV pitch in the array. This equation shows that a small TSV array and a wide TSV pitch are desirable for low TSV height variation. Based on this model, we can compute a TSV pitch for the size of a given TSV array, which guarantees low TSV height variation endured by a bonding technique. Then, TSV arrays are built and then simultaneously floorplanned with cores and routers. For example, if the size of a TSV array including a one-way link (113 wires) in OCP is 11×11, its TSV pitch must be at least 14.58μm for TSV height variation less than 1μm. On the contrary, if the size of a TSV array including a two-way link (226 wires) in OCP is 16×16, its TSV pitch must be at least 21.21μm for TSV height variation less than 1μm. Thus, the widths of 11×11 and 16×16 TSV arrays are 160μm and 339μm

and their areas are 0.0256mm^2 and 0.1151mm^2, respectively. Consequently, two one-way vertical links shows lower CMP variation or smaller design area than a single two-way vertical link if the performance and energy constraints of a synthesized network are satisfied.

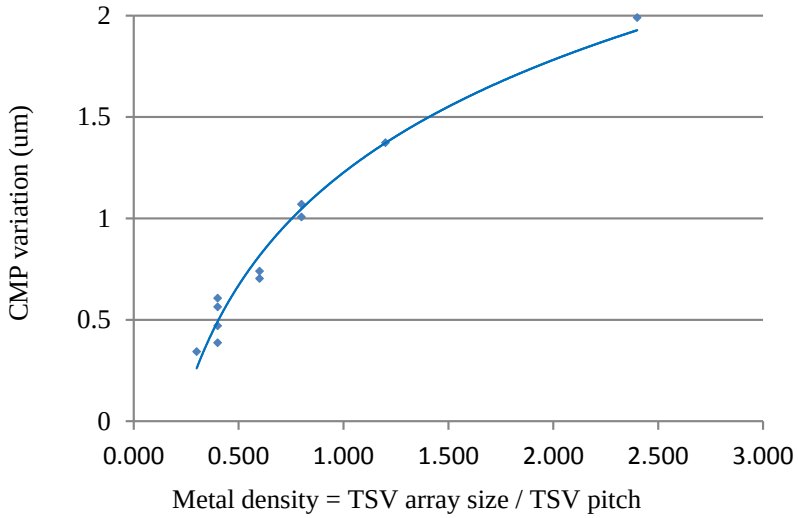


Figure 4.7: TSV height variation by TSV density.

4.1.4.2 CMP-Aware Application-Specific 3D NoC

We implement the proposed CMP-aware application-specific 3D NoC, called CAS with 4-8 layers in C++. We repeat CAS for ten times on GSRC Benchmarks with 100, 200 and 300 modules [34] and compute average to obtain reliable statistics. Wafers are stacked in a face-to-back fashion and we set the diameter and pitch of TSV to 5μm and 10μm, respectively. Both [120] and CAS employ MST and P2P as a 2D network topology. Since the goal of MST is extremely opposite to that of P2P, the performance and design cost improvement of CAS with other 2D network topologies will be within

the gap of improvement of CAS with MST and P2P. Note that [80] and [102] are not suitable for comparison for 3D NoC with large routers and TSV arrays.

Table 4.1 shows TSV height variation when various network interfaces are used. When a 3D network topology is decided, CAS inserts only one-way links between layers whereas [120] inserts both one-way and two-way links. Thus, the local TSV density of CAS is more uniform and lower than that of [120] and after silicon-CMP, CAS has 17.9% lower TSV height variation than [120]. Using only one-way links results in increasing hop count since it may not provide the shortest path. However, our 3D NoC design flow recovers the penalty of the hop count and even improves total hop count since a topology decision is first performed.

Table 4.2 shows total hop count. CAS achieves, on average, 15% lower hop count than [120]. CAS tends to further improve hop count in complex NoC with a number of modules and layers. In addition, when a network is synthesized with limited resources like MST, CAS further improves hop count.

In Table 4.3, we compare the total wirelength of CAS with that of [120]. Even if CAS performs floorplanning with the maximum allowable wirelength constraint after synthesizing a network topology, it achieves just 0.3% longer total wirelength than [120] in MST and even 4.6% shorter total wirelength than [120] in P2P.

Network protocol	# of wire of one(two)-way link	[120]	CAS	Imp. (%)
AHB [2]	137 (274)	1.651	1.372	16.9
AXI [2]	204 (408)	1.821	1.551	14.8
APB [2]	99 (198)	1.551	1.226	21.0
OCP [86]	113 (226)	1.603	1.302	18.7
Average		1.657	1.363	17.9

Table 4.1: TSV height variation comparison (μm).

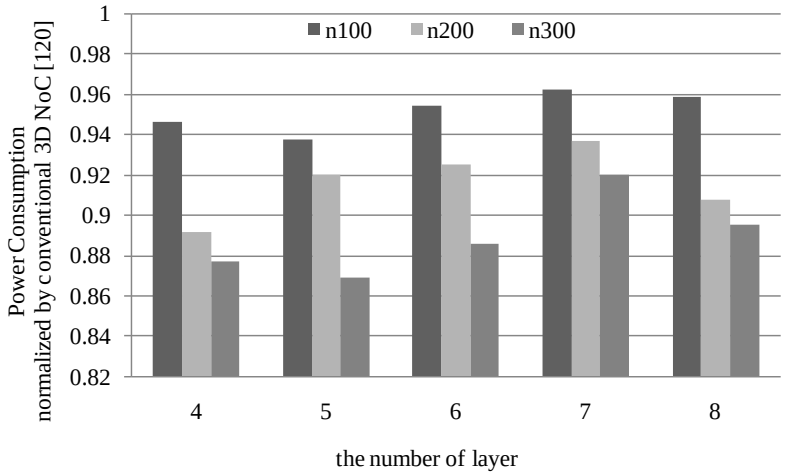
the number of layer			4	5	6	7	8	Imp. (%)
MST	n100	[120]	1410	1470	1625	1671	1927	16.5
		CAS	1201	1254	1299	1361	1650	
	n200	[120]	3341	3366	3459	3737	3927	20.0
		CAS	2654	2931	2698	2934	3043	
	n300	[120]	5211	5158	5178	5257	5230	22.6
		CAS	4065	3912	4077	3984	4118	
	Imp. (%)		20.5	19.0	21.3	22.4	20.5	19.7
P2P	n100	[120]	1193	1336	1488	1629	1799	13.1
		CAS	1077	1194	1207	1416	1575	
	n200	[120]	2051	2487	2744	3136	3285	8.7
		CAS	2041	2278	2441	2788	2968	
	n300	[120]	2638	3279	3433	4163	4371	11.1
		CAS	2626	2943	3405	3234	3695	
	Imp. (%)		2.3	9.7	8.0	16.7	12.9	11.0

Table 4.2: Hop count comparison.

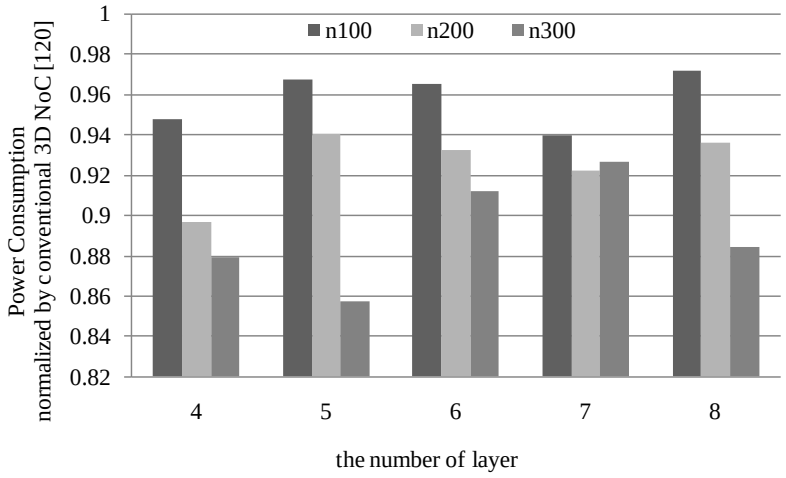
the number of layer			4	5	6	7	8	Imp. (%)
MST	n100	[120]	9.6	8.4	7.5	7.1	6.5	-0.3
		CAS	9.6	8.5	7.4	7.1	6.6	
	n200	[120]	22.6	19.1	16.6	15.1	13.8	0.0
		CAS	23.2	19.1	16.6	15.0	13.3	
	n300	[120]	46.5	39.5	34.4	30.5	27.2	-0.6
		CAS	47.0	40.0	34.1	30.7	27.3	
	Imp. (%)		-1.4	-0.9	0.7	-0.2	0.6	-0.3
P2P	n100	[120]	47.2	38.6	32.9	29.6	26.4	2.1
		CAS	46.5	38.3	32.3	28.0	26.2	
	n200	[120]	95.4	77.7	67.8	61.3	55.2	4.7
		CAS	89.6	75.1	65.0	58.5	52.3	
	n300	[120]	144.6	129.9	115.5	102.6	94.5	7.0
		CAS	132.1	119.6	108.6	99.4	86.1	
	Imp. (%)		7.5	5.4	4.8	3.9	6.5	4.6

Table 4.3: Total wirelength comparison (mm).

Figure 4.8 shows power consumption normalized by [120]. The power consumption of CAS is 8.1% and 7.8% lower than that of [120] in MST and P2P, respectively. CAS tends to further improve power consumption in NoC with a lot of modules.



(a) MST



(b) P2P

Figure 4.8: Network topologies and layouts performed by CMP-aware 3D NoC.

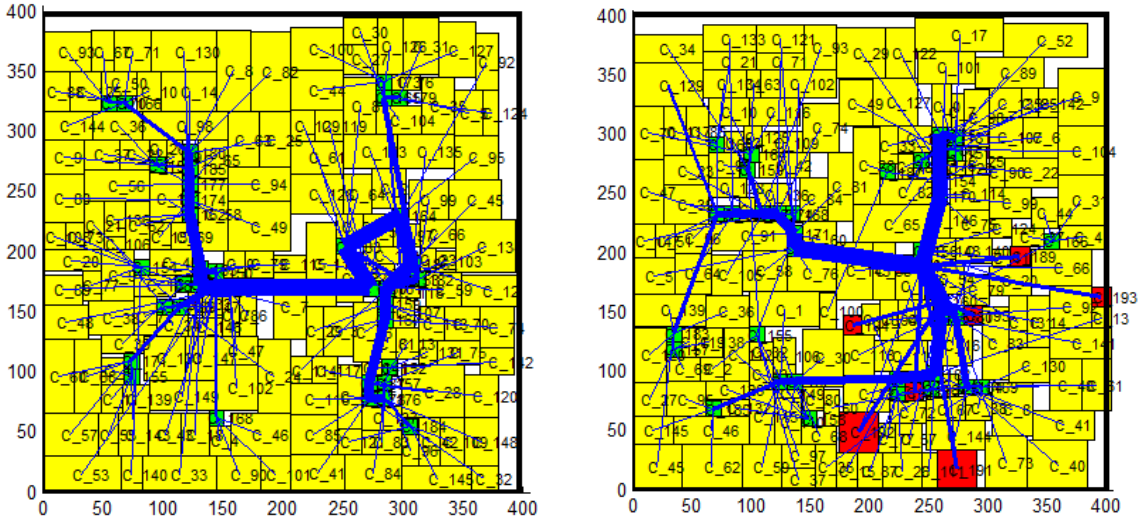
The total area of CAS is slightly smaller than [120] since CAS has smaller total TSV array area than [120]. The runtime of CAS ranges from 48-99 seconds in n300, which is about three times faster than [120].

Figure 4.9 and Figure 4.10 show the layouts of [120]+MST and CAS+MST with 2 layers in n300, respectively, where blue lines show communication relations and their thickness indicates communication volume. Yellow rectangles, red rectangles, and green rectangles are cores, TSV arrays, and routers, respectively. While layer 2 in Figure 4.9 includes both one-way and two-way links, layer 2 in Figure 4.10 includes just one-way links. Therefore, TSV heights are less variable, and thus can contact landing pads easily.

In Figure 4.11, CAS proves more merits on NoC with complex routers, where CAS further improves power consumption and total wirelength as the area of routers increases. Since previous NoC designs first floorplan only cores before synthesizing network topology, neither is the dead space sufficient for complex routers inserted nor wirelength and power consumption are well optimized even if floorplanning is again performed after synthesizing a network.

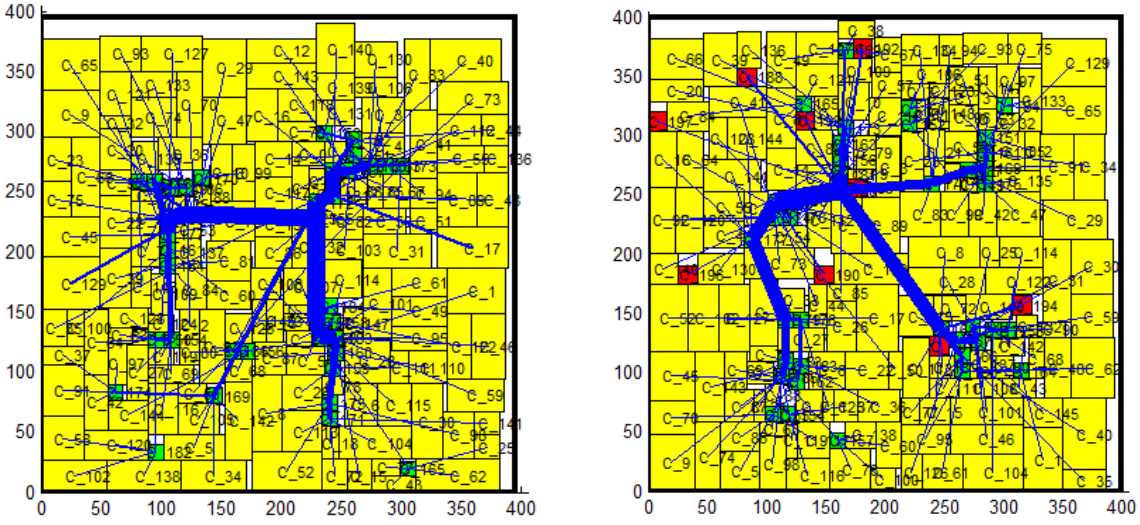
4.1.5 Summary

In this section, we propose the first CMP-aware application-specific 3D NoC design. Our vertical integration managing architecture, physical design, and manufacturing issues together enables a reliable and robust 3D NoC. In particular, our CMP-aware 3D NoC approach reduces TSV height variation after the CMP process, and thus prevents severe bonding failures and timing variation. Meanwhile, it also improves hop count, wirelength, power consumption, and area, compared to the previous state-of-the-art 3D NoC [120].



(a) Layer 1 (b) Layer 2

Figure 4.9: Typical application-specific 3D NoC with 2 layers [120].



(a) Layer 1 (b) Layer 2

Figure 4.10: CMP-aware application-specific 3D NoC with 2 layers.

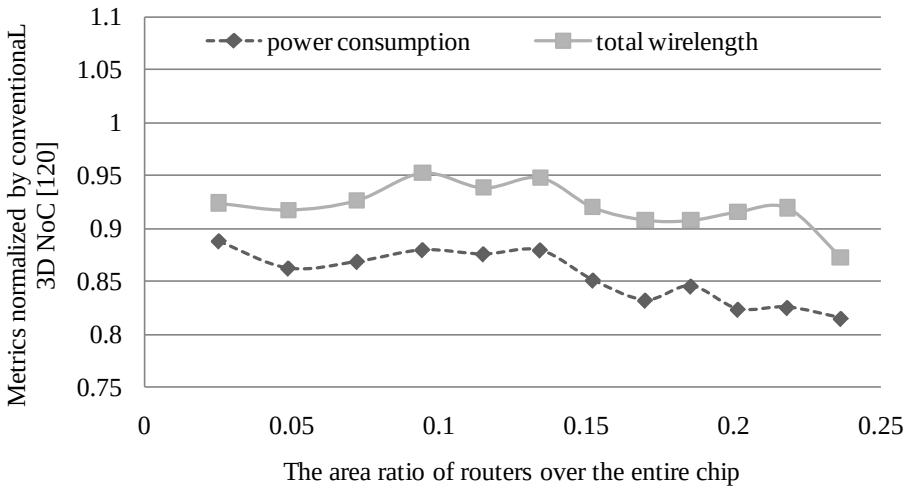


Figure 4.11: Improvement according to the area of routers.

Chapter 5

Conclusions

This dissertation presents systematic architecture and physical design for mitigating the challenges of advanced NoCs in terms of latency, power, and emerging technologies. Our major contributions include:

- In Chapter 2, we developed SDRAM- and application-aware routers and memory subsystems to improve application-level or system-level latency. The multiple SDRAM-aware routers instead of a single memory subsystem scheduled memory requests to prevent bank conflict, data contention, and short turn-around bank interleaving. Moreover, the SDRAM-aware router was advanced to an application-aware router with the consideration of the demands of various applications, such as different memory latency requirements and memory access granularities. Our results showed that the cost-effective SDRAM- and application-aware NoC design significantly provided not only high memory utilization and short average latency but also high QoS.

- In Chapter 3, we proposed a VFI-aware NoC optimization framework in order to reduce both computation and communication energy consumption. It consisted of three key VFI-aware components, i.e. VFI-aware core partitioning with voltage and frequency assignment, VFI-aware mapping, and VFI-aware routing path allocation. Moreover, we developed VFI interfaces and their insertion algorithm to easily satisfy performance constraints. The proposed methodology made cores using the same voltage and clock speed unified to single VFI and thus considerably reduced VFI overheads. In addition, we presented architecture-aware analytic application mapping algorithms applied to various

networks in order to reduce communication energy consumption and average latency. The application mapping problem was formulated to MIQP and then solved by successive relation and genetic algorithms. Our results showed that the proposed application mapping algorithms greatly reduced power consumption on various networks. Especially, they showed more merits on irregular mesh and custom networks.

- In Chapter 4, we presented a CMP-aware application-specific 3D NoC design that minimized TSV height variation, thus reduces a bonding failure, and meanwhile optimized conventional NoC design objectives such as hop count, wirelength, power consumption, and area. Since synthesizing an on-chip network has been always subject to technology constraints, NoC architecture and physical design techniques should be compatible with 3D technologies. The key idea behind the proposed 3D NoC design flow was to determine the CMP-aware 3D NoC topology where different layers were interconnected by one-way links with the minimum hops. Our results showed that our CMP-aware 3D NoC design achieved smaller chip area, lower hop count, shorter wirelength, and lower power consumption than the previous state-of-the-art 3D NoC designs.

In this dissertation, we emphasize the importance of synergistic architecture and physical design techniques for emerging technologies. We hope that this work motivates future research follow-up in this domain. Some of the future directions may include:

- Phase change memory (PCM) is an emerging memory technology for future computing systems. Compared to other non-volatile memory alternatives, PCM is relatively more matured to production and has a fast read latency and potentially high

storage density. The main bottleneck precluding PCM from being used, in particular, in the main memory hierarchy, is its limited write endurance. To mitigate this problem, recent studies proposed to either reduce the write frequency of PCM or use wear-leveling to evenly distribute writes. Although these techniques can extend the lifetime of PCM, they will not prevent deliberately designed malicious codes from wearing it out quickly. Furthermore, most previous techniques did not consider the dynamic access pattern of various applications, in particular, interleaved access pattern in MPEG 1/2/4 and H.264. Therefore, we need to improve PCM write endurance at an application or system level.

- Based on recent opto-electro material/device level break-throughs, on-chip nanophotonics offers compelling high throughput/bandwidth communication and promising low power integration opportunities compared with traditional Cu/low-K interconnect, therefore is considered as a potential quantum leap towards the next generation on-chip interconnect. Despite of its superior signaling speed and low power potentials, the on-chip nanophotonics faces major roadblocks for interconnecting on-chip computation resources. Major challenges in this field include but are not limited to: photonic network architecture design, low power high performance integration, device characterization, and thermal reliability modeling/optimization. With the promising low power on-chip optical links, NoC optimization flows can be extended for further improving power efficiency. Other important future works along this direction may include thermal reliability optimizations for on-chip nanophotonic links and potential applications.

Bibliography

[1] B. Akesson, K. Goossens and M. Ringhofer. Predator: a predictable SDRAM memory controller. In *Proc. International Conference on Hardware/Software Codesign and System Synthesis*, 2007.

[2] *AMBA open specifications*. ARM [Online]. Available: http://www.arm.com.

[3] K. Athikulwongse, A. Chakraborty, J.-S. Yang, D. Z. Pan, and S. K. Lim. Stress-driven 3D-IC placement with TSV keep-out zone and regularity study. In *Proc. International Conference on Computer-Aided Design*, 2010.

[4] L. Benini and G. D. Micheli. Network on chips: a new SoC paradigm. *Computer*, vol. 35, no. 1, pp. 70-78, 2002.

[5] T. Bjerregaard and J. Sparsø. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proc. Design, Automation and Test in Europe*, 2005.

[6] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny. Routing table minimization for irregular mesh NoCs. In *Proc. Design, Automation and Test in Europe*, 2007.

[7] J. A. Butts and G. S. Sohi. A static power model for architects. In *Proc. International Symposium of Microarchitecture*, 2000.

[8] G. Campobello, M. Castano, C. Ciofi, and D. Mangano. GALS networks on chip: a new solution for asynchronous delay-insensitive links. In *Proc. Design, Automation & Test in Europe*, 2006.

[9] E. Carvalho, N. Calazans, and F. Moraes. Heuristics for dynamic task mapping in NoC-based heterogeneous MPSOCs. In *Proc. International Workshop on Rapid System Prototyping*, 2007.

[10] P.-C. Chang, I-W. Wu, J.-J. Shann, and C.-P. Chung. ETAHM: an energy-aware task allocation algorithm for heterogeneous multiprocessor. In *Proc. Design Automation Conference*, 2008.

[11] K S. Chatha, K. Srinivasan, and G. Konjevod. Automated techniques for synthesis of application-specific network-on-chip architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27. no. 8, Aug. 2008.

[12] T. Chelcea and S. M. Nowick. A low latency FIFO for mixed-clock system. In *Proc. IEEE Computer Society Annual Workshop on VLSI*, 2000.

[13] G. Chen, F. Li, S. W. Son, and M. Kandemir. Application mapping for chip multiprocessor. In *Proc. Design Automation Conference*, 2008.

[14] R. L. S. Ching, E. F. Y. Young, K. C. K. Leung, and C. Chu. Post-placement voltage island generation. In *Proc. International Conference on Computer-Aided Design*, 2006.

[15] C.-L. Chou, U. Y. Ogras, and R. Marculescu. Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1866-1879, Oct. 2008.

[16] P. Choudhary and D. Marculescu. Hardware based frequency/voltage control of voltage frequency island systems. In *Proc. International Conference on Hardware/Software Codesign and System Synthesis*, 2006.

[17] J. Cong, J. Wei, and Y. Zhang. A thermal-driven floorplanning algorithm. In *Proc. International Conference on Computer-Aided Design*, 2004.

[18] W. J. Dally and B. Towles. Route Packets, Not wires: On-chip interconnection networks. In *Proc. Design Aumation Conference*, 2001.

[19] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.

[20] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen. A low-latency and memory-efficient on-chip network. In *Proc. International Symposium on Networks-on-Chip*, 2010.

[21] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das. Application-aware prioritization mechanisms for on-chip networks. In *Proc. International Symposium on Microarchitecture*, 2009.

[22] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das. Aergia: exploiting packet latency slack in on-chip networks. In *Proc. International Symposium on Computer Architecture*, 2010.

[23] *Databahn DRAM Memory Controller IP*. *Denali Software Inc*. [Online]. Available: http://www.denali.com.

[24] *DDR I, II and III device operations & timing diagram*. Samsung Electronics [Online]. Available: http://www.samsung.com/global/business/semiconductor.

[25] R. P. Dick. Embedded system synthesis benchmarks suites (E3S) [Online]. Available: http://www.ece.northwestern.edu/~dickrp/e3s/.

[26] R. P. Dick, D. L. Rhodes, and W. Wolf. TGFF: task graphs for free. In *Proc. International Workshop on Hardware/Software Codesign*, 1998.

[27] N. Dutt. Memory-aware NoC exploration and design. In *Proc. Design Automation and Test in Europe*, 2008.

[28] S. Dutta, R. Jensen, and A. Rieckkmann. Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems. *IEEE Design and Test of Computers*, vol. 18, no. 5, pp. 21-31, Sep./Oct. 2001.

[29] M. A. A. Faruque, R. Krist, and J. Henkel. ADAM: Run-time agent-based distributed application mapping for on-chip communication. In *Proc. Design Automation Conference*, 2008.

[30] T. Gan. Modeling of chemical mechanical polishing for shallow trench isolation. In *Ph.D. Thesis, Massachusetts Institute of Technology, 2000.*

[31] K. Goossens, J. Dielissen, and A. Rădulescu. Æthereal network on chip: concepts, architectures and implementations. *IEEE Design and Test of Computers*, 22(5): 414–421, 2005.

[32] P. Gopalakrishnan, X. Li, and L. Pileggi. Architecture-aware FPGA placement using metric embedding. In *Proc. Design Automation Conference*, 2006.

[33] I.E. Grossmann and Z. Kravanja. *Mixed-integer nonlinear programming: A survey of algorithms and applications. Large-Scale Optimization with Applications, Part II: Optimal Design and Control, A.R. Conn, L.T. Biegler, T.F. Coleman, and F.N. Santosa (Eds.),* Springer: New York, Berlin, 1997.

[34] *GSRC Floorplan Benchmarks* [Online]. Available: http://vlsicad.eecs.umich.edu/BK/GSRCbench.

[35] O. He, S. Dong, J. Bian, S. Goto, and C.-K. Cheng. A novel fixed-outline floorplanner with zero deadspace for hierarchical design. In *Proc. International Conference on Computer-Aided Design*, 2008.

[36] S. Heithecker and R. Ernst. Traffic shaping for an FPGA based SDRAM controller with complex QoS requirement. In *Proc. Design Automation Conference*, 2005.

[37] *hMETIS - Hypergraph & Circuit Partitioning* [Online]. Avaiable: http://glaros.dtc.umn.edu/gkhome/views/metis

[38] R. Holsmark, M. Palesi, and S. Kumar. Deadlock free routing algorithms for irregular mesh toplogy NoC systems with rectangular regions. *Journal of System Architecture*, vol. 54, no. 3-4, pp. 384-396, Mar. 2008

[39] J. Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Proc. Asian and South Pacific Design Automation Conference*, 2003.

[40] J. Hu and R. Marculescu. Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures. In *Proc. Design, Automation & Test in Europe*, 2003.

[41] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu. Architecting voltage islands in core-based system-on-a-chip designs. In *Proc. International Symposium on Low Power Electronics and Design*, 2004.

[42] J. Hu and R. Marculescu. Communication and task scheduling of application-specific networks-on-chips. In *Proc. IEE Comuters & Digital Techniques*, 2006.

[43] W.-L. Hung, G.M. Link, Y. Xie, N. Vijaykrishnan, and M. J. Irwin. Interconnect and thermal-aware floorplanning for 3D microprocessors. In *Proc. International Symposium on Quality Electronic Design*, 2006.

[44] I. Hur and C. Lin. Memory scheduling for modern microprocessors. *ACM Trans. on Computer Systems*, vol. 25, no. 10, pp. 10.1-10.36, Dec. 2007.

[45] *IBM ASIC Solutions*. IBM [Online]. Available: http://www.ibm.com.

[46] IMEC [Online]. Available: http://www.imec.be.

[47] *Intel XScale core*. Intel [Online]. Available: http://www.intel.com.

[48] International Technology Roadmap for Semiconductors, 2009.

[49] W. Jang, D. Ding, and D. Z. Pan. Voltage-frequency island aware energy optimization framework for networks-on-chip. In *Proc. International Conference on Computer-Aided Design*, 2008.

[50] W. Jang and D. Z. Pan. An SDRAM-aware router for networks-on-chip. In *Proc. Design Automation Conference*, 2009.

[51] W. Jang and D. Z. Pan. A3MAP: Architecture-aware analytic mapping for networks-on-chip. In *Proc. Asian and South Pacific Design Automation Conference*, 2010.

[52] W. Jang and D. Z. Pan. Application-aware NoC design for efficient SDRAM access. In *Proc. Design Automation Conference*, 2010.

[53] W. Jang, D. Ding, and D. Z. Pan. Voltage frequency island optimization for many-core/NoC designs. In *Proc. International Conference on Green Circuits and Systems*, 2010.

[54] W. Jang and D. Z. Pan. An SDRAM-aware router for networks-on-chip. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1572-1585, Oct. 2010.

[55] W. Jang and D. Z. Pan. Application-aware NoC design for efficient SDRAM access. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2011. (to appear).

[56] A. B. Kahng and K. Samadi. CMP fill synthesis: a survey of recent studies. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 3-19, 2008.

[57] D. H. Kim, Y.-K. Wu, R. O. Topaloglu, and S. K. Lim. Enabling 3D integration through optimal topograph. In *Proc. International Workshop on Design for Manufacturability and Yield Workshop*, 2010.

[58] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *Proc. Design Automation Conference*, 2005.

[59] J. Kim, B.-C. Lai, M.-C. F. Chang, and I. Verbauwhede. A cost-effective latency-aware memory bus for symmetric multiprocessor systems. *IEEE Transactions on Computers*, vol. 57, no. 12, pp. 1714-1719, 2008.

[60] W.-C. Kwon, S.-M. Hong, S. Yoo, B. Min, K.-M. Choi and S.-K. Eo. An open-loop flow control scheme based on the accurate global information of on-chip communication. In *Proc. Design, Automation and Test in Europe*, 2008.

[61] W.-C. Kwon, S. Yoo, S.-M. Hong, B. Min, K.-M. Choi and S.-K. Eo. A practical approach of memory access parallelization to exploit multiple off-chip DDR memories. In *Proc. Design Automation Conference*, 2008.

[62] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn. Managing power and performance for system-on- chip designs using voltage islands. In *Proc. International Conference on Computer-Aided Design*, 2002.

[63] Y.-C. Lan, S.-H. Lo, Y.-C. Lin, Y.-H. Hu and S.-J. Chen. BiNoC: a bidirectional NoC architecture with dynamic self-reconfigurable channel. In *Proc. International Symposium on Networks on chip*, 2009.

[64] C. J. Lee, O. Mutlu, V. Narasiman, and Y. N. Patt. Prefetch-aware DRAM controller. In *Proc. International Symposium on Microarchitecture*, 2008.

[65] C. J. Lee, V. Narasiman, O. Mutlu, and Y. N. Patt. Improving memory bank-level parallelism in the presence of prefetching. In *Proc. International Symposium on Microarchitecture*, 2009.

[66] L.-F. Leung and C.-Y. Tsui. Energy-aware synthesis of networks-on-chip implemented with voltage island. In *Proc. Design Automation Conference*, 2007.

[67] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan and M. Kandemir. Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. In *Proc. International Symposium on Computer Architecture*, 2006.

[68] S. K. Lim. TSV-aware 3D Physical design toll needs for faster mainstream acceptance of 3D ICs [Online]. Available: http://www.dac.com

[69] Y. Liu, Y. Yang, and J. Hu. Clustering-based simultaneous task and voltage scheduling for NoC systems. In *Proc. International Conference on Computer-Aided Design*, 2010.

[70] J. Luo and D. A. Dornfeld. Material removal mechanism in chemical mechanical polishing: theory and modeling. *IEEE Trans. on Semiconductor Manufacturing*, vol. 14, no. 2, pp. 112-133, May. 2001.

[71] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3-21, Jan. 2009.

[72] Y. Markovsky, Y. Patel, and J. Wawrzynek. Using adaptive routing to compensate for performance heterogeneity. In *Proc. International Symposium on Networks-on-Chip*, 2009.

[73] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proc. International Conference on Computer-Aided Design*, 2002.

[74] J. Matousek, *Lectures in Discrete Geometry*, Springer, 2002.

[75] *MemMax Scheduler*. Sonics Inc. [Online]. Available*:* http://www.sonicsinc.com.

[76] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *Proc. Design Automation and Test in Europe*, 2004.

[77] *Mobile AMD Athon 4 processor model 6*. AMD [Online]. Available: http://www.amd.com.

[78] P. Morrow, M. J. Kobrinsky, S. Ramanathan, C.-M. Park, M. Harmes, V. Ramachandrarao, H. mog Park, Grant Kloster, Scott List, and Sarah Kim. Wafer-level 3D interconnects via Cu bonding. In *Proc. Advanced Metalization Conference*, 2004.

[79] S. Murali and G. De Micheli. Bandwidth-constrained mapping of cores onto NoC architecture. In *Proc. Design, Automation & Test in Europe*, 2004.

[80] S. Murali, C. Seiculescu, L. Benini, and G. De Micheli. Synthesis of networks on chips for 3D systems on chips. In *Proc. Asia South Pacific Design Automation Conference*, 2009.

[81] K. Niyogi and D. Marculescu. Speed and voltage selection for GALS systems based on voltage/frequency islands. In *Proc. Asian and South Pacific Design Automation Conference*, 2005.

[82] *Nomadik Multimedia Processors*. STMicroelectronics [Online]. Available: http://www.st.com.

[83] U. Y. Ogras and R. Marculescu. Prediction-based flow control for network-on-chip traffic. In *Proc. Design Automation Conference*, 2006.

[84] U. Y. Ogras, R. Marcuescu, P. Choudhary, and D. Marculescu. Voltage-frequency island partitioning for GALS-based networks-on-chip. In *Proc. Design Automation Conference*, 2007.

[85] I. Oliver, D. Smith, and J. Holland. A Study of permutation crossover operators on the traveling salesman problem. In *Proc. Conference on Genetic Algorithm*, 1987.

[86] *Open core protocol specification*. OCP [Online]. Available: http://www.ocpip.org.

[87] *Optimization software for operations research applications*. AIMMS [Online]. Available: http://www.aimms.com.

[88] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L.-S. Peh. Research challenges for on-chip interconnection networks. IEEE Micro, vol. 27, no. 5, pp. 96-108, 2007.

[89] F. Paganini, J. Doyle and S. Low. Scalable laws for stable network congestion control. In *Proc. International Conference on Decision and Control*, 2001.

[90] P. R. Panda, N. D. Dutt, and A. Nicolau. Exploiting off-chip memory access modes in high-level synthesis. In *Proc. International Conference on Computer-Aided Design*, 1997.

[91] M. Pathak, Y.-J. Lee, T. Moon, and S. K. Lim. Through-silicon-via management during 3D physical design: when to add and how many? In *Proc. International Conference on Computer-Aided Design*, 2010.

[92] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Efficient synthesis of networks on chip. In *Proc. International Conference on Computer Design*, 2003.

[93] R. Puri, L. Stok, J. Cohn, D. Dung, D. Pan, D. Sylvester, and A. Srivastava. Pushing ASIC performance in a power envelope. In *Proc. Design Automation Conference*, 2003

[94] D. Qiu and N. B. Shroff. A Predictive flow control scheme for efficient network utilization and QoS. *IEEE Trans. on Networking*, vol. 12, no. 1, pp. 161-172, Feb. 2004.

[95] J. Quartana, S. Renane, A. Baixas, L. Fesquet, and M. Renaudin. GALS systems prototyping using multiclock FPGAs and asynchronous network-on-chips. In *Proc. International Conference on Field Programmable Logic and Applications*, 2005.

[96] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson and J. D. Owens. Memory access scheduling. In *Proc. International Symposium on Compter Architecture*, 2000.

[97] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, vol. 23, no. 3, pp. 555-565, Jul. 1976.

[98] T. Sakurai and A. R. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2): 584-594, 1990.

[99] Samsung unveils HDTV system-on-chip. *EE Times* (2004, Jul) [Online]. Available: http://www.eetimes.com/electronics-news/4049612/Samsung-unveils-HDTV-system-on-chip.

[100] M. F.-F. Schafer, T. Hollstein, H. Zimmer, and M. Glesner. Deadlock-free routing and component placement for irregular mesh-based network-on-chip. In *Proc. International Conference on Cumputer Aided Design*, 2005.

[101] C. Seiculescu, S. Murali, L. Benini, G. D. Micheli. SunFloor 3D: A tool for networks on chip topology synthesis for 3D systems on chips. In *Proc. Design Automation and Test in Europe*, 2009

[102] C. Seiculescu, S. Murali, L. Benini, and G. D. Micheli. NoC topology synthesis for supporting shutdown of voltage island in SoCs. In *Proc. Design Automation Conference*, 2009.

[103] D. Shin and J. Kim. Power-aware communication optimization for networks-on-chip with voltage scalable links. In *Proc. International Conference on Hardware/Software Codesign and System Synthesis*, 2004.

[104] S. Sivaram, H. Bath, R. Legegett, A. Maury, K. Monning, and R. Tolles. Planarizing interlevel dielectrics by chemical mechanical polishing. In *Proc. International Electron Devices Meeting*, 2006.

[105] L. T. Smit, G. J.M. Smit, J. L. Hurink, H. Broersma, D. Paulusma, and P. T. Wolkotte. Run-time assignment of tasks to multiple heterogeneous processors. In *Proc. 4th PROGRESS Workshop on Embedded System*, 2004.

[106] *STBus communication system concepts and definitions*. STMicroelectronics [Online]. Available: http://www.st.com

[107] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal. Free PDK: An open-source variation-aware design kit. In *Proc. IEEE International Conference on Microelectronic Systems Education*, 2007.

[108] B. Swinnen, W. Ruythooren, P. De Moor, L. Bogaerts, L. Carbonell, K. De Munck, B. Eyckens, S. Stoukatch, D. Sabuncuoglu Tezcan, Z. Tokei, J. Vaes, J. Van Aelst, and E. Beyne. 3D integration by Cu-Cu thermo-compression bonding of extremely thinned bulk-Si die containing 10um pitch through-SI vias. In Proc international Electron Devices Meeting, 2006.

[109] R. Tornero, M. J. Orduna, M. Palesi, and J. Duato. A communication-aware topological mapping technique for NoCs. In *Proceedings of 14th International conference on Parallel and Distributed Computing*, 2008.

[110] J. Van Olmen, A. Mercha, G. Katti, C. Huyghebaert, J. Van Aelst, E. Seppala, Zhao Chao, S. Armini, J. Vaes, R. Cotrin Teixeira, M. Van Cauwenberghe, P. Verdonck. K. Verhemeldonck, A. Jourdain, W. Ruythooren, M. de Potter de ten Broeck, A. Opdebeeck, T. Chiarella, B. Parvais, I. Debusschere, T.Y. Hoffmann, B. De Wachter, W. Dehaene, M. Stucchi, M. Rakowski, Ph. Soussan, R. Cartuyvels, E. Beyne, S. Biesemans, and B. Swinnen. 3D stacked IC demonstration using a through silicon via first approach. In *Proc. International Electron Devices Meeting*, 2008.

[111] E. B. Van der Tol and E. G. T. Jaspers. Mapping of MPEG-4 decoding on flexible architecture platform. In *Proc. SPIE*, 2002.

[112] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar and S. Borkar An 80-tile sub-100-w teraflops processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[113] L. A. Vervoort, P. Yeung and A. Reddy. *Addressing memory bandwidth needs for next-generation digital TVs with cost-effective, high- performance Consumer DRAM Solutions*. Rambus inc., Jul. 2007.

[114] W. D. Weber. *Efficient shared DRAM subsystem for SoCs*. Sonics Inc., 2001.

[115] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and System for Video Technology*, vol. 13, no. 7, pp. 560-576, Jul. 2003.

[116] D. Wingard. Micronetwork-based integration for SoCs. In *Proc. Design Automation Conference*, 2001.

[117] *Wireless Handset Solutions: OMAP Platform*. Texas Instruments [Online]. Available: http://www.ti.com.

[118] H. Wu, I-M. Liu. M. D. F. Wong, and Y. Wang. Post-placement voltage island generation under performance requirement. In *Proc. International Conference on Computer-Aided Design*, 2005.

[119] H. Wu, M. D. F. Wong, and I-M. Liu. Timing-constrained and voltage-island-aware voltage assignment. In *Proc. Design Automation Conference*, 2006.

[120] S. Yan and B. Lin. Design of application-specific 3D networks-on-chip architectures. In *Proc. International Conference on Computer Design*, 2008.

[121] J.-S. Yang, K. Athikulwongse, Y.-J. Lee, S. K. Lim, and D. Z. Pan. TSV Stress aware timing analysis with applications to 3D-IC layout optimization. In *Proc. Design Automation Conference*, 2010.

Vita

Woo Young Jang received the B.E. degree in Radio Science and Technology from Kyunghee University, South Korea, in 1998 and the M.S. degree in Electrical and Computer Engineering from Yonsei University, South Korea in 2000. He has been with System LSI Division in Samsung Electronics CO., LTD., South Korea since 2000, where he has developed memory subsystems, on-chip communication architectures, and MPEG-1/2 and H.264 decoders for DTV SoCs. He joined in the Department of Electrical and Computer Engineering at the University of Texas at Austin in Fall 2006, where he started graduate studies for Ph.D. degree with Prof. David Z. Pan. His main research includes computer architecture and nanometer physical design for on-chip communication and image/video processing. Mr. Jang was the recipient of SK telecom Scholarship for 1994-1997, Samsung Outstanding Achievement Award in 2005 and Samsung Scholarship for 2006-2011.

Permanent address (or email): wooyoung.jjang@gmail.com

This dissertation was typed by Woo Young Jang.