國立交通大學

電子工程學系 電子研究所

碩士論文

應用於無線影像娛樂系統之以記憶體為重心的晶內互聯網路

Memory-Centric On-Chip Interconnection Network for Wireless Video Entertainment Systems

研究生：王湘斐

指導教授：黃 威 教授

中華民國九十九年七月

應用於無線影像娛樂系統之以記憶體為重心的晶內互聯網路

Memory-Centric On-Chip Interconnection Network for Wireless Video Entertainment Systems

研 究 生：王湘斐　　Student：Shiang-Fei Wang

指導教授：黃　威 教授　　Advisor：Prof. Wei Hwang

國 立 交 通 大 學

電 子 工 程 學 系 電 子 研 究 所

碩 士 論 文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical Engineering and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electronics Engineering

July 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年七月

應用於無線影像娛樂系統之以記憶體為重心的晶內互聯網路

學生：王湘斐　　指導教授：黃　威　教授

國立交通大學電子工程學系電子研究所

摘　　要

本論文實現一個可應用於無線影像娛樂系統之以記憶體為重心的晶內互聯網路。在這個晶內互聯網路之中，我們利用記憶體借用的機制讓網路介面可以有效的降低資料阻塞的情況。除此之外，對於無線影像娛樂系統，晶內互聯網路提供微架構和基本建構元素給晶內網路包含網路介面，路由器，連接線路。藉由考量可借出的記憶體區塊大小和分散式的記憶體管理單元，可以動態調整輸出隊列的大小。基於在 SystemC 週期推動的模擬結果，本論文所提出的網路介面相較於一般的網路介面可以提升 1.15 倍的效能。同時可以減少 2%~4%的資料阻塞情況。提供一個良好的資料傳輸環境給無線影像娛樂系統。

Memory-Centric On-Chip Interconnection Network for Wireless Video Entertainment Systems

Student : Shiang-Fei Wang Advisors : Prof. Wei Hwang

Department of Electronics Engineering & Institute of Electronics

National Chiao-Tung University

ABSTRACT

A memory-centric on-chip interconnection network (OCIN) with an efficient network interface is realized in this thesis. Additionally, a borrowing mechanism is proposed to reduce head-of-line data blocking. Furthermore, for wireless video entertainment system, on-chip interconnection network (OCIN) provides the micro architecture and the building blocks, including network interfaces (NIs), routers and link wires.

By considering the borrowed memory blocks and distributed memory management unit (d-MMU), the size of output queue in the NI can be dynamically scheduled. Based on the cycle-driven simulation results in SystemC, the proposed efficient NI can achieve performance improvement by 1.15x compared to the conventional NI. The blocking condition can be reduced 2%~4%. For on-demand memory system, we can efficiently reduce data blocking by adjusting buffer size and borrowed memory blocks. Under the condition of 70% blocking rate of receiver and 16 words of borrowed memory blocks, the data blocking reduction rate can reach 25%. With the proposed memory-centric OCIN, we can improve the data communication environment for wireless video entertainment systems.

致謝

感謝許多人的幫助，讓我完成了這一篇論文。

首先，我要感謝我的指導教授黃威，在他的指導下讓我對自己研究的領域有更深入的瞭解，建立了研究的興趣與自信心。黃教授提供了一個常優良的研究環境與充足的研究資源，讓我能夠充分發揮自己的能力完成這一篇論文。

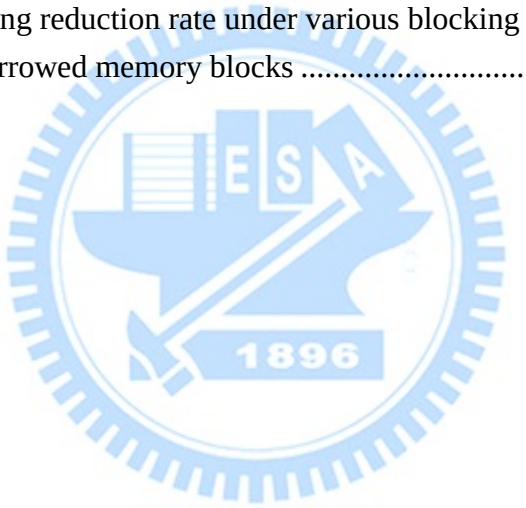
感謝實驗室的成員，在過去的兩年當中在生活上與研究上對我的幫助。感謝黃柏蒼、張銘宏和張雍對於我在研究上的幫忙，讓我能夠有更加優良的研究成果。

最後我要感謝我的家人對我在生活上的關心與幫助，讓我能夠順利的完成碩士的論文研究。

Contents

List of Figures

List of Tables



Chapter 1

Introduction

1.1 Background

With development of System-on-Chip (SoC) and multimedia communication technologies, a great amount of data computing requirement increases rapidly. The bandwidth requirement between the processing cores in SoCs is increasing. The aggregate communication bandwidth between the processing cores is in the GBytes/s range for many video applications. In the future, with the integration of many applications onto a single device and with increased processing speed of cores, the bandwidth demands will scale up to much larger values. Multiprocessor system-on-chip (MP-SoC) [1.1][1.2][1.3][1.4][1.5] architectures are emerging as appealing solutions for embedded multimedia applications. In general, MP-SoCs are composed of core processors, memories and some application-specific coprocessors. Communication is provided by advanced interconnect fabrics, such as high performance and efficient networks-on-chip (NoCs)[1.6].

System-on-Chip (SoC) design is an integrated solution for merging processor elements (PEs) in communications, multimedia and consumer electronics. However, as design complexity of SoC continues to increase, the requirements for on-chip communication bandwidth among PEs are growing continually. Therefore, a global approach is needed to effectively transport and manage on-chip communication traffic, and optimize wire efficiency. Process-independent network-on-chip (NoC) has been considered an effective solution to integrate a multi-core system and a packet switched approach [1.7][1.8][1.9]. NoC was investigated for dealing with the challenges of on-chip data communication caused by the increasing scale of next

generation SoC designs. Furthermore, on-chip interconnection networks (OCINs) provide the micro-architecture and the building blocks for NoCs, including network interfaces (NIs), routers and link wires [1.10].

1.2 Motivation

NoCs design is essentially important for the future System-on-Chip (SoC) design. However, as the increasing multimedia data computing grow, data blocking condition in the output of processer elements will be more and more common. A fine-design network interface and interconnection network can provide a bridge between various processor elements which can reduce occurrence of data blocking efficiently. Moreover, it provides a well communication environment for wireless video entertainment system.

1.3 Contribution

In this thesis, we propose the memory-centric on-chip data communication platform for e-Home II project which includes two parts: on-demand memory system and on-chip interconnection network (OCIN). My thesis focuses the on on-chip interconnection network. We propose an efficient network interface and a crossbar (interconnection network) for data communication among various processor elements. Finally, we integrate a heterogeneous platform for wireless video entertainment system.

1.4 Organization

The organization of this thesis is as follows. An overview of on-chip interconnection network is introduced in the chapter 2. In this chapter, the design concept of NoCs will be described. Then, we would introduce flow control mechanism and interconnection network (crossbar) including arbitration mechanism in the chapter 3. Chapter 4 presents an efficient network interface for memory-centric on-chip interconnection network which can reduce the data blocking by a borrowing mechanism. Finally, a memory-centric on-chip data communication platform developed for a wireless video entertainment system will be introduced in the chapter 5.

Chapter 2
Previous Work of On-Chip Interconnection Network

In this chapter, I describe how Network on Chip (NoC) will be the next major challenge to implement complex and function-rich application in advanced process technologies in section 2.1. The general design concept is discussed in section 2.2. The interconnect architecture, topologies, of NoC should be efficient for a huge amount of processor elements. A number of different interconnect architectures will been present in section 2.3. Moreover, some advance topologies are present to adopt with on-chip platform. Switching fabrics (or called router) is a key component in network-on-chip to command the data communication. I will describe the components in switch fabrics and how they influence the NoC systems in section 2.4, which includes four parts: routing units, buffers, switching circuits, and arbitration unit. In addition, the implementation of each unit will be described also.

2.2 Why NoC?

System-on-chip (SOC) designs provide the integrated solution to the challenging design problems in the communications, multimedia and consumer electronics. Moreover, every year System-on-Chip designs become increasingly complex, while the associated numbers of transistors grows exponentially. The successful design of SoC depends on the availability of the methodologies that allow designers to copy with two major challenges: the extreme miniaturization of device and wire features, and the extremely large scale of integration. Most SoC will find their application within embedded systems, traditional figures of merit, such as performance, energy consumption and cost. It will be as important as the first-design correct and reliable operation and robustness. Modern SoC design is faced with a number of problems

caused by the scale and complexity of the designs. For ideal IP-based SoC, on-chip bus interfaces between each IP and a good verification environment [2.1][2.2].

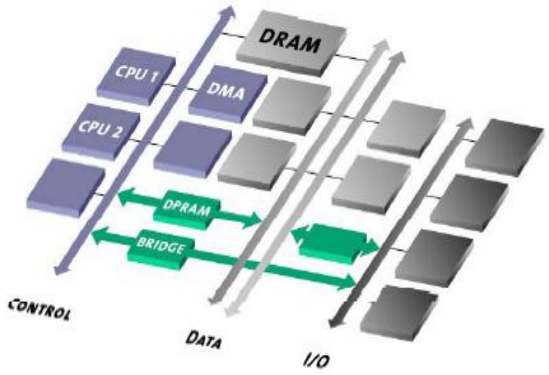


Figure 2.1 Traditional Synchronous Bus

In the next SoC era, however, there are some challenges for traditional on-chip bus platform which is shown in Fig. 2.1. First, the required on-chip communication bandwidth is growing beyond that provided by standard on-chip buses [2.3]. The shared bus architecture will limit the development factor for integration with increasing IP blocks. Existing bus architectures and techniques are proving to be non-scalable, unable to meet leading edge complexity and performance requirements.

Second, the interconnect delay across the chip exceeds the average clock period of the IP blocks, especially in nano-scale technologies [2.4]. The ratio of global interconnect delay to average clock period will continue to grow. In a 60nm process, a signal can reach only 5% of the die's length in a clock cycle. However, an interconnect channel design methodology for high performance ICs has proposed in [2.5], it devised a methodology to size the FIFOs in an interconnect channel containing one or more FIFOs connected in series and shows that the sizing of the FIFOs in the channel is a function of system parameters such as data production rate and communication rate, number of channel stages etc.

Third, in nano-scale technologies, increased coupling effect for interconnects not only aggravates the power-delay metrics but also deteriorates the signal integrity due to capacitive and inductive crosstalk noises. Several options were proposed to reduce the inter-wire capacitances. The first option is to widen the pitch between bus lines. The second option is using P&R (place & route) tools to avoid routing of the bus lines side by side. In System-on-Chip, however, the interconnect complexity and the

routing time do not allow us trying it to minimize the coupling capacitances. The third option is to change the geometrical shape of bus lines. But the disadvantage of this method is that the frank area will increase since the cross-sectional area of a bus line is fixed. The fourth technique is to add a shielding line (VDD/Ground) between two adjacent signal lines. The fifth option is to reduce power is through the use of bus encoding schemes [2.6][2.7][2.8][2.9][2.10][2.11]. By the end of the decade, using 60 nm transistors operating below one volt, with grow to 4 billion transistors running at 10GHz, according to the International Technology Roadmap for Semiconductors. On-chip physical interconnections will present a limited factor for performance and energy consumption. The encoding schemes for low power and reliability issues are proposed in [2.12]. The designers must overcome the challenge of noises to provide the function correct, reliable operation of the interacting components. A robust self-calibrating transmission scheme for interconnections is proposed in [2.13] and it examines some physical properties of on-chip interconnects, with the goal of achieving fast, reliable and low-energy communication.

Forth, both the system design and performance are limited by the complexity of the interconnection between the different modules and blocks into single clocked design. Different data transfer speeds are required, as well as parallel transmission. The traditional system buses may not be suitable for such a system since only one module can transmit at a time. Additionally, the modern SOC designer assembles the system using ready virtual components which might not be easily adaptable to different clocking situations. The solution to above problems is a segmented bus design combined with the concept of the globally asynchronous local synchronous (GALS) system architecture [2.14][2.15][2.16][2.17][2.18][2.19]. Asynchronous design can make the circuits resilient to delay variation.

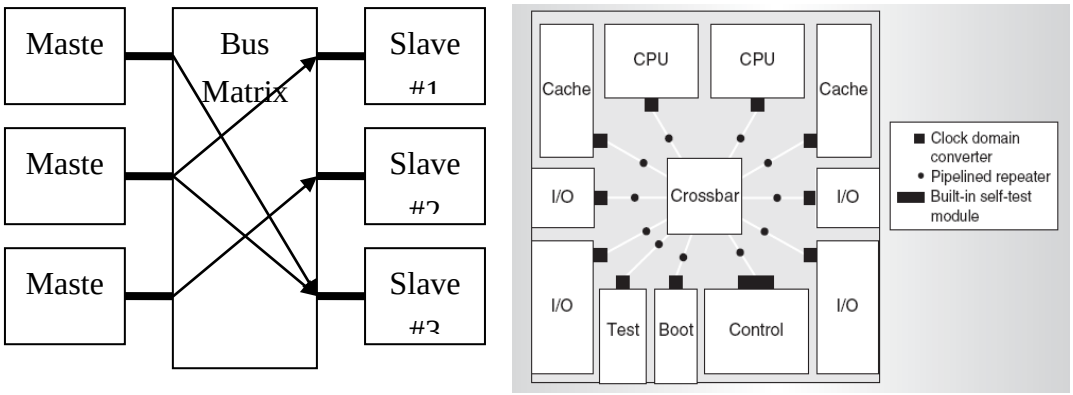


Figure 2.2 (a) Multi-Layer Bus Architecture (b) Centralized Crossbar Switch

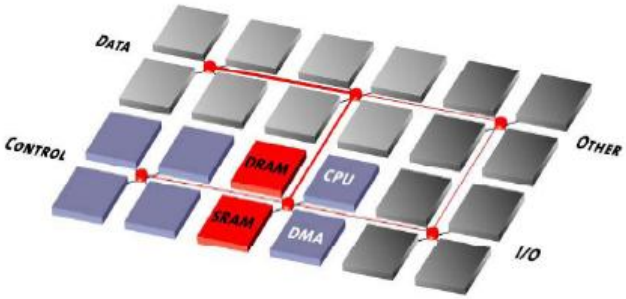


Figure 2.3 Network-on-Chip Architecture

For the above mentioned problems, new architectures for the on-chip communications are proposed to adapt the next SoC era. The traditional synchronous on-chip bus architectures as Fig. 1 are faced a serious of acid tests which are mentioned in the last paragraphs. Multi-layer on-chip shared bus as Fig. 2.2(a) is the advised version of the traditional on-chip bus to reduce the shared-medium channels [2.20][2.21][2.22]. It's the specification of an interconnect scheme that overcome the limitations of shared bus. Therefore, it enables parallel access paths between multiple masters and slaves by a bus matrix. When each master has its corresponding bus, the structure is equivalent to a full crossbar as Fig. 2.2(b). However, not only centralized crossbar switching systems but also multi-layer bus architectures will be confused with complex wire routings which will introduce larger power consumption and interconnect delay with increasing processor elements.

The network-on-chip architecture as Figure 2.3 is based on a homogeneous and scalable switch fabric network, which considers all the requirements of on-chip communications and traffic. NoCs have a few distinctive characteristics, namely low communication latency [2.23][2.24][2.25], energy consumption constraints and

design-time specialization. The motivation of establishing NoC platform is to achieve performance using a system perspective of communication. The core of NoC technology is the active switching fabric that manages multi-purpose data packets within complex, IP laden designs. The most important characteristics of NoC architecture can be summarized as packet switched approach [2.26], flexible and user-defined topology and global asynchronous locally synchronous (GALS) implementation.

2.2 The Design Concept of Network-on-Chip

The topic of Network-on-Chip(NoC) designs is vast and complex. There is a large literature on architectures for NoCs. Consider on-chip communication and its abstraction of network-on-chip as a micro-network and analyze the various levels of the micro-network stack bottom to up as right part in Fig. 2.4, starting from physical layer to software layer. NoC protocols are typical organized in layers, in a fashion that resembles the OSI protocol stacks as the left part in Fig. 2.4 [2.27]. However, the OSI protocol stacks is resembled for a marco-network. For a micro-network, the protocol stack will be reduced to physical layer, data-link layer, network and transport layer and software later [2.28]. The characteristics of each layer will be described in this section.

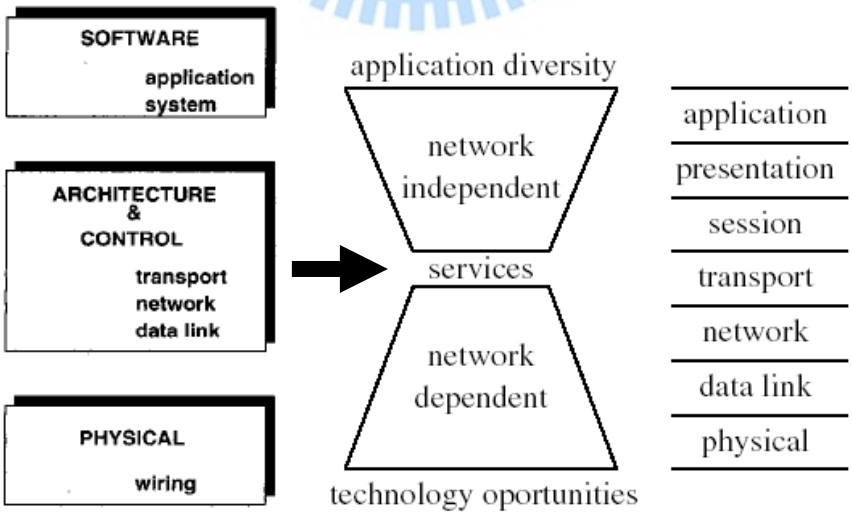


Figure 2.4 The design abstraction levels of NoC

2.2.1 The Design Abstraction Levels of Network-on-Chip

NoC protocols are described bottom-up, starting from the physical up to the software layer. In the physical layer, global wires are the physical implementation of the communication channels. Traditional rail-to-rail voltage signaling with capacitive termination, as used today for on-chip communication, is definitely not well-suited for high-speed, low-energy communications for future global interconnect. Reduced swing can significantly reduce communication power dissipation which preserves the speed of data communication. Nevertheless, as the technology trends lead us to use smaller voltage swings and capacitances, the upset probabilities will rise. It is important to realize that a well-balanced design should not over design wires so that their behavior approaches an ideal one, because that the corresponding cost in performance, energy-efficiency and modularity may be too high. Physical layer design should find a compromise between competing quality metrics and provide a clean and complete abstraction of channel characteristics to micro-network layers above.

Due to the limitations in the physical level and the high bandwidth requirement, the SoC design will use network architectures similar to those used for multi-processors. Network-on-chip design entails the specification of network architectures and control protocols. The data-link layer abstracts the physical layer as an unreliable digital link, where the probability of bit upsets is non null. Furthermore, reliability can be traded off for energy**.** The main purpose of data-link protocols is to increase the reliability of the link up to a minimum required level, under the assumption that the physical layer by itself is not sufficiently reliable. At the data link layer, error correction can be complemented by several packet-based error detection and recovery protocols. Several parameters in the protocols can be adjusted depending on the goal to achieve maximum performance at a specified residual error probability within given energy consumption bounds.

At the network layer, packet data transmission can be customized by the choice of switching and routing algorithms. The NoC designers establish the type of connection to its final destination. Switching and routing affect heavily performance

and energy consumption. Robustness and fault tolerance will also be highly desirable. At the transport layer, algorithms deal with the decomposition of messages into packets at the source and their assembly at destination. Packetization granularity is a critical design decision because the behavior of most network control algorithm is very sensitive to packet size. Packet size can be application specific in SoCs, as opposed to general network.

Software layers comprise system and application software which includes processing element and network operating systems. The system software provides us with an abstraction of the underlying hardware platform. Moreover, policies implemented at the system software layer request either specific protocols or parameters at the lower layers to achieve the appropriate information flow. The hardware abstraction is coupled to the design of wrappers for processor cores which perform as network interfaces between cores and NoC architecture.

2.3 Topologies for Network-on-Chip Architecture

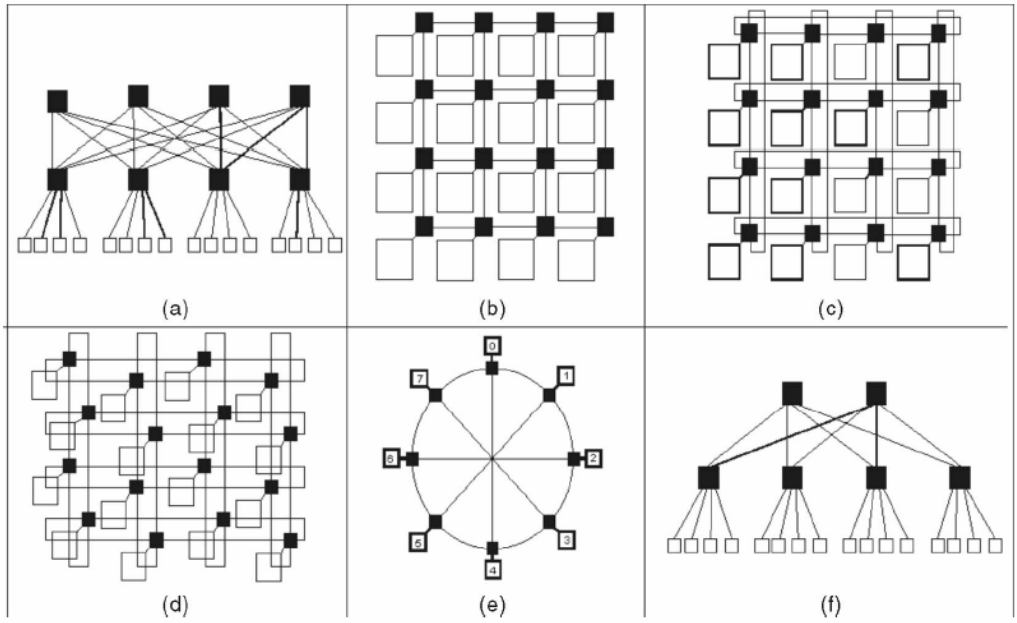


Figure 2.5 NoC architecture (a) SPIN (b) Mesh (c) Torus (d) Folded tours (e) Octagon (f) Butterfly Fat Tree

Network on Chip (NoC) technologies will enable designing parallel systems resembling cellular structures including thousands of processors. Such systems combined with multi-threaded computing can increase system efficiency for fine-grain parallel programs [2.29][2.30]. Therefore, the interconnect architecture of

NoC should be efficient for a huge amount of processor elements. A number of different interconnect architectures have been proposed as Fig. 2.8. Their origins can be traced back to the field of parallel computing. However, a different set of constraints exists when adapting these architectures to the SoC design paradigm.

2.3.1 Conventional Topologies of Network-on-Chip

A generic interconnect template has proposed which is called SPIN (Scalable, Programmable, Integrated Network) for on-chip packet switched interconnections as Fig. 2.8(a), where a fat-tree architecture is used to interconnect IP blocks. In this fat tree, every node has four children and the parent is replicated four times at any level of the tree. The functional IP blocks reside at the leaves and the switches reside at the vertices. A mesh-based [2.31][2.32] interconnect architecture consists of an mxn mesh of switches interconnecting computational resources (IPs) placed along with the switches, as shown in Fig. 2.8(b). Every switch, except those at the edges, is connected to four neighboring switches and one IP block.

2D torus has proposed as NoC architecture, shown in Fig. 2.8(c). The Torus architecture is basically the same as a regular mesh. The only difference is that the switches at the edges are connected to the switches at the opposite edge through wrap-around channels. Every switch has five ports, one connected to the local resource and the others connected to the closest neighboring switches. The long end-around connections can yield excessive delays. However, this can be avoided by folding the torus as Fig. 2.8(d). This renders to a more suitable VLSI implementation.

Karim et al. [2.33] have proposed the OCTAGON MP-SoC architecture. Fig. 2.8(e) shows a basic octagon unit consisting of eight nodes and 12 bidirectional links. Each node is associated with a processing element and a switch. Communication between any pair of nodes takes at most two hops within the basic octagonal unit. For a system consisting of more than eight nodes, the octagon is extended to multidimensional space. Of course, this type of interconnection mechanism may significantly increase the wiring complexity. In a Butterfly Fat-Tree (BFT) architecture which is shown as Fig. 2.8(f), the IPs are placed at the leaves and switches placed at the vertices. A pair of coordinates is used to label each node. The

number of switches in the butterfly fat tree architecture converges to a constant independent of the number of levels.

2.3.2 Advanced Network-on-Chip Architectures

A popular network topology of NoC implementations is the two-dimensional mesh architecture, and it provides a regular topology and communications. Therefore, many advanced NoC architectures are proposed which are based on mesh topologies. An advanced NoC architecture, called Xpipes as Fig. 2.9, targeting high performance and reliable communication for on-chip multi-processors is introduced [2.34]. Data links can be pipelined with a flexible number of stages to decouple link throughput from its length and to get arbitrary topologies. The I/O ports of each switch can be parameterized, and Xpipes is optimized from tile-based network on chip architecture. Although it has dealt with the floorplan and different bandwidth between neighboring IP blocks, it belongs to the 2-D links architecture.

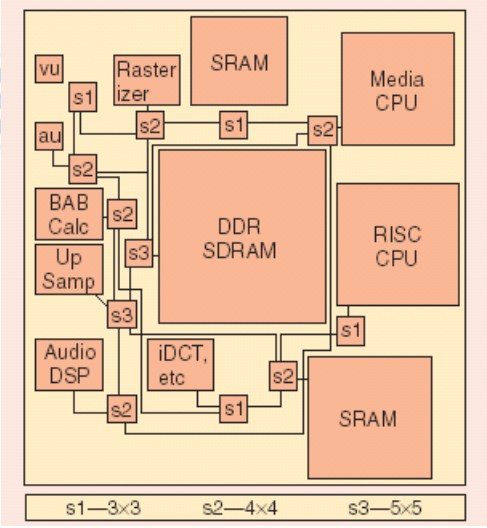


Figure 2.6 Xipies Architecture

An idea is presented to connect to the hierarchy network-on-chip as shown in Figure 2.10. The network on chip can be divided into two kinds of architecture, local network and global network. The local network preserves the features of 2-D links network on chip, and the global network is designed as centralized crossbar [2.35]. With the increasing of the processor elements and numbers of the local network, however, the global network might be designed as the distributed crossbars. In Figure2.13, block M is mentioned as memory block and block P is about the

processor element. Other hierarchy Network-on-chip or hybrid network-on-chip are also proposed to adopt multiple processor elements and heterogeneous systems [2.36][2.37] [2.38] [2.39].

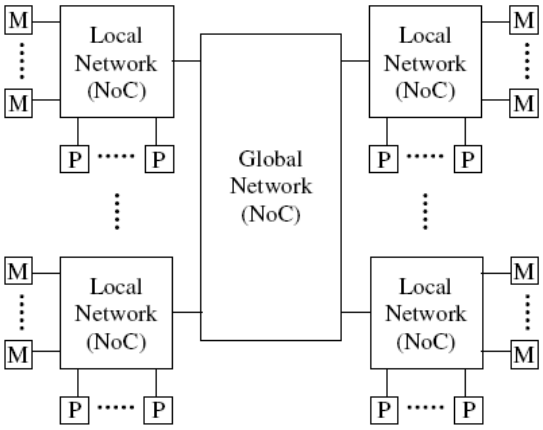


Figure 2.7 Hierarchy Network-on-Chip Architecture

In order to achieve better performance, functionality and packaging density, three dimensional ICs are proposed with multiple layers of active devices. Besides, three-dimensional (3D) ICs allow for performance enhancements in the absence of scaling. This is the result of each transistor being able to reduce interconnect length and access more nearest neighbors. The performance improvement arising from the architectural advantages of NoCs will be significantly enhanced if 3D ICs are adopted as the basic fabrication methodologies. Therefore, new topologies of 3-D network are also proposed for the future ICs [2.40].

2.4 Switching Fabrics in Network-on-Chip

Switching fabrics (or called router) is a key component in network-on-chip to command the data communication. Every processor element is called resource and connects to a switch fabric. The resources consist of process elements, IP blocks, embedded memory, DMA controllers etc. The implementation of routers depends on the topology and protocol of network-on-chip. In addition, the topology and control flows are the design issue for the interfaces of processor elements. No whether which network-on-chip architecture is, the router could be divided into five parts as follow:

- I/O Ports
- Link Control Unit (Routing Unit)
- Buffers (queues)

- Switching Circuit
- Arbitration Unit

The link control units (routing units) control the communication in the network-on-chip backbone, and the arbitration unit arbitrates contention data which are routed to the same channel. The NoC former should avoid deadlock [2.41] of the on-chip communications and traffic which are intruded by the bad policy routing algorithms. Besides, it will influence on the sizes of buffers, number of MUXs for switching and the complexity of interconnects. For example, each switch connects to the side of switches with four directions in a mesh network-on-chip. The links of the switches are shown in Figure 2.8(b) and the architecture of the switches in a mesh (tile-based) NoC is shown in Figure 2.10.

We would not introduce link control unit (routing unit) here because in this thesis we focus on buffers (queues), switching circuit (network interface) and arbitration unit. The detail of these units will be described in following sections.

2.4.1 Buffers in Switch Fabrics

In network-on-chip platform, buffers will significantly affect the overall performance and the arbitration algorithm. Buffer allow for local storage of data that cannot be immediately routed. Unfortunately, queuing buffers have a high cost in terms of area and power consumption, and thus many NoC implementations strive with limited buffer sizes. If the design lacks sufficient buffer space, on the contrary, the buffers may fill up too fast while over-provisioning of buffers clearly is a waste of scarce area resources [2.42].

Queuing buffer is used in switch fabrics or network interfaces to store un-routed data, and buffer architectures can be classified by the location and circuit implementation of buffers. The queuing buffers consume the most area and power consumption among composing blocks in NoCs. However, insufficient buffer size is a factor to induce head-of-line blocking problems as Fig. 2.8. When the head data of a virtual channel could not be routed and data behind the head data are occupied queuing buffers, it will influence the performance of the network. That’s the so-called “head-of-line blocking problem.” Nevertheless, head-of-line blocking problems not

only reduce the performance but also increase power consumption of on-chip communication. Therefore, head-of-line blocking is a key factor to evaluate different buffer architecture.

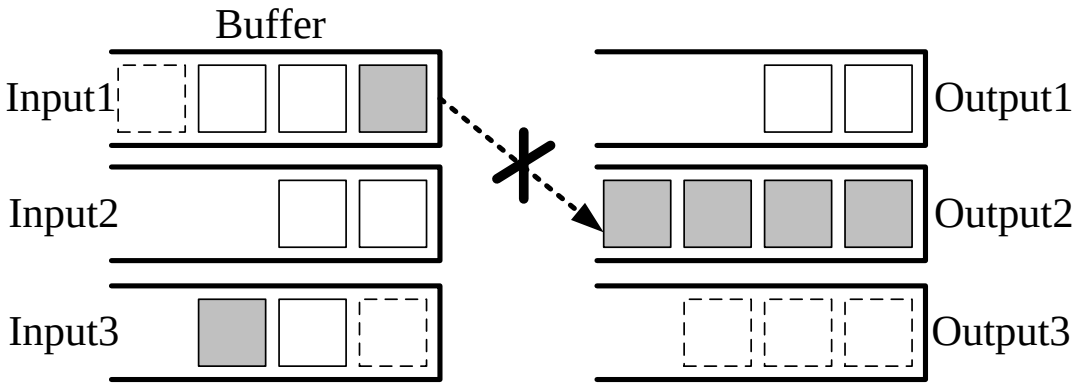


Figure 2.8 Head-of-line blocking problem

Depending on the location of queuing buffers, the buffers can be placed either before or after the interconnection matrix in a switch fabric, which are input buffer and output buffer, respectively. To be sure, there is a distinction between input buffers and output buffers. If a data word is delayed in a switch fabric with input buffers, it will stall all data words arriving on the same input. None of them can be processed until the first one has been forwarded successfully. With the output buffers, the situation is different because that the switching is performed before the buffering. If a switch fabric cannot send the data over one of its outputs, the buffers at that output will fill up. There is, however, no immediate influence on the inputs. The successive data words can still be received. An architectural disadvantage of output buffering is that in one cycle, data from multiple input ports may have to be written to the same output port. Nevertheless, the multiple buffers can be implemented in parallel at the output to deal with the disadvantage. No whether output buffers or input buffers, they will introduce head-of-line blocking problem to stall the input data.

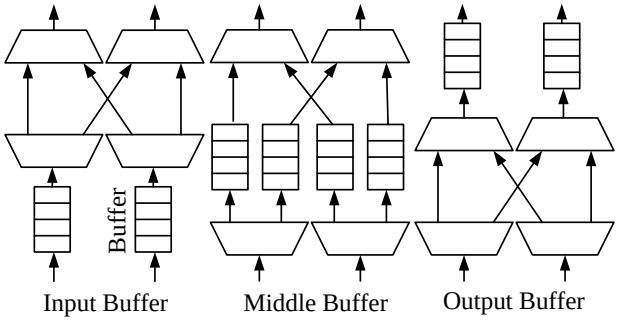


Figure 2.9 Buffer architecture

Fig. 2.9 shows the input buffers, middle buffers and output buffers in the switch fabrics. The concept of middle buffering describes that the cutest of the buffers placement moves to the middle of the switch. The middle buffer architectures have $O(N^2)$ buffer blocks for a N-port switch fabric while input and output buffering have only O(N) buffer blocks. The middle buffer architecture, however, can reduce the effects of the head-of-line blocking with multiple virtual channels in the switching. It will be a trade-off between traffic problems and buffer sizes. Consequently, both output buffers and middle buffers are looking for the best FIFO utility.

The traditional buffer circuits can be implemented by two different memory units, either registers (flip-flops) or SRAM cells. Register-based implementations have a definite limitation in their capacity as to the increasing power consumption and area. Therefore, the queuing buffer should be implemented as SRAM cells with separated read/write ports for large capacity queuing. This is because that area of SRAM cells is tenth of registers.

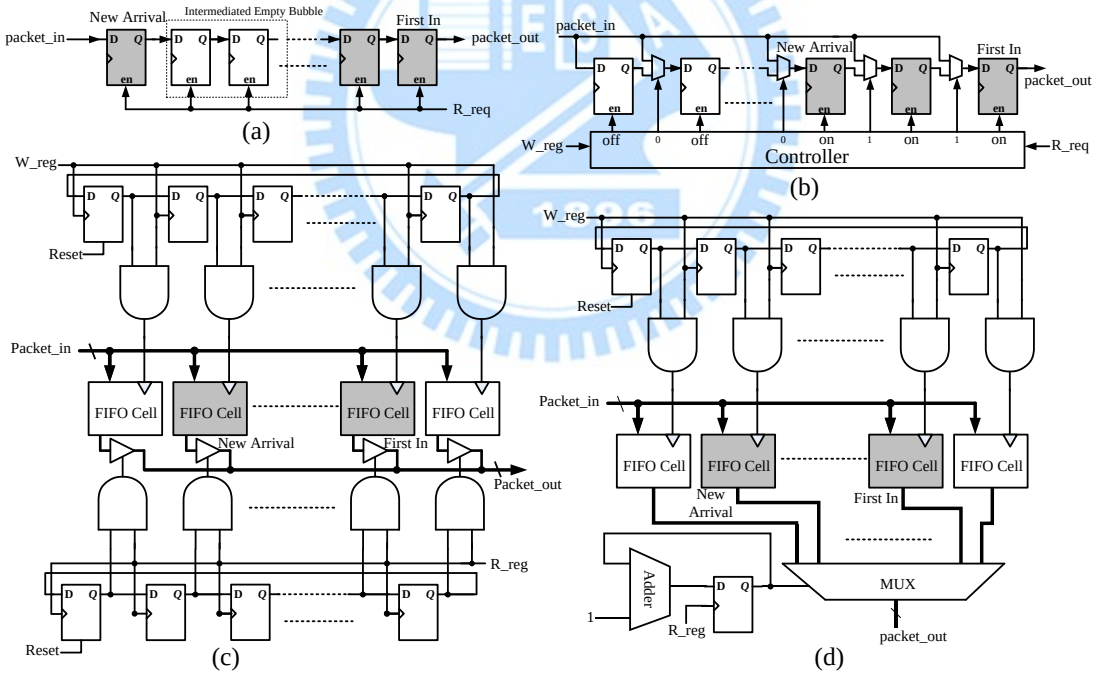


Figure 2.10 Buffer Implementation

The register-based buffers could be classified into four different implementations, which are shown as Fig. 2.10. Fig. 2.10(a) shows a conventional shift-register. When the consumer sends the request to the buffer, it will enable all registers and shift the data to the output port. Indeed, the implementation of a shift-register is uncomplicated

than others. However, intermediate empty cells as Fig. 2.10(a) induced by different packet in/out rates temporally will influence the performance as to unnecessary latency. Nevertheless, shifting all the registers in a buffer consumes huge amount of power. The shift-register is not desirable to implement on a chip as to unnecessary latency and huge power consumption. In order to remove the intermediate empty bubble, the arrival packet can be stored at the empty cell behind the full cells as Fig. 2.10(b). This style is called as "Bus-In Shift-Out Register", and it only shifts the full cells. Therefore, it can remove unnecessary latency and power consumption caused by the empty bubbles. However, as the queuing capacity increases, the driver of the sender should be larger for the increasing fan-outs. Besides, it still consumes huge amount power by shifting all the occupied cells. To reduce the huge power consumption of shifting operation, the outputs of all registers are connected to a shared output bus via tri-state buffers as Fig. 2.10(c) which is called "Bus-In Bus-Out Register". The writing and reading tokens which are constructed in rings indicate the head and tail of full cells, respectively. The tri-stare buffers are controlled by the reading token to read the first-in packet, and the writing token enables the register which is behind the full cells to store the input packet. As the queuing capacity increases, the capacitance of the shared input/output buses increase as well, output bus especially. The parasitic capacitance of tri-state buffers will enlarge not only the delay but also power consumption. Therefore, "Bus-In Mux-Out Register" with output multiplexers as Fig. 2.10(d) can be used to eliminate the parasitic capacitance of tri-state buffers. It needs an extra adder to be a pointer and to calculate the address of output packet.

Some approaches are proposed to optimize the location and size of buffers. Application-specific buffer space allocation is a novel system-level buffer planning algorithm to customize the router design [2.43][2.44][2.45]. Centralized buffer and dynamic virtual channel regulator are proposed to decrease the buffer size without performance overhead. FC-CB is designed for virtual channels in wormhole routing [2.46]. Other approaches are proposed to define the buffer model and buffer constraint in NoC systems.

2.4.2 Switching Circuit in Switch Fabrics

There are two kinds of switch design: a cross-point switch and a MUX-based switch. The cross-point switch has pass transistors at each crossing junction of input and output wires. The capacitive loading of input driver is the junction capacitance of pass transistors on input and output wires and the wire capacitance itself. The voltage swing on the output wire is reduced to Vdd-Vth as to the threshold voltage drop. The fabric area is determined by the wiring area so that its area cost can be the minimum. However, this design is hard to be synthesized and sensitive to the noise. The MUX-based switches use multiplexer for each output port. The power consumption, delay and area are all worse than cross-point switching, especially for large input/output. Nevertheless, the power consumption and delay will exponentially increase with the number of I/O ports no whether Mux-based switch or cross-point switch. Crossbar partial activation technique as Fig. 2.11 is proposed to reduce the power consumption and delay with large input and output [2.47].

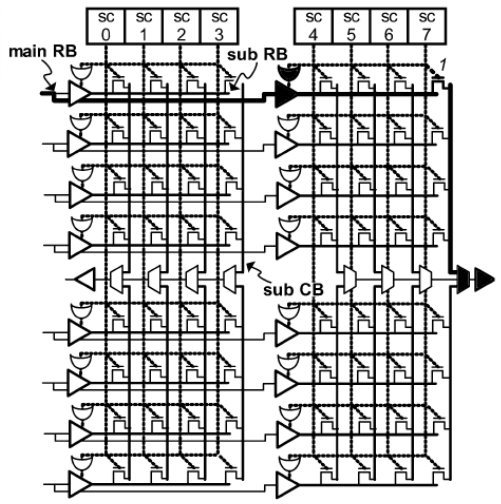


Figure 2.11 Crossbar partial activation technique

2.4.3 Arbitration Unit Circuit in Switch Fabrics

To arbitrate the output conflicts, arbitration unit is used on each output. The latency of the arbitration unit becomes larger as the switch size gets bigger. Besides, the contention data will increase also. TDMA [2.48][2.49][2.50] and round-robin

scheduling algorithm [2.51][2.52] are most widely used as to its simple implementation and its fairness, respectively. A pseudo-LRU algorithm is also proposed for lower are and lower latency than those of the round-robin algorithm. Besides, input contention-aware arbitration algorithm is proposed to achieve higher performance by considering the traffic of neighbor nodes. For a distributed parallel characteristic of NoC platform, the arbitration unit is less important than shared medium platform, such as on-chip bus architecture.

Chapter 3
Flow Control for On-Chip Interconnection Network

3.1 Overview of Flow Control

Inter-processor communication can be view as a hierarchy of services starting from the physical layer that synchronizes the transfer of bit streams to higher-level protocol layers that perform functions such as packetization, data encryption, data compression, and so on. We find it useful to distinguish between three layers in the operation of the interconnection network: the routing layer, the switching layer, and the physical layer. The switching layer utilizes these physical layer protocols to implement mechanisms for forwarding messages through the network. The switching techniques determine when and how internal switches are set to connect router inputs to outputs and the time at which message components may be transferred along these paths. These techniques are coupled with flow control mechanisms for the synchronized transfer of units of information between routers and through routers in forwarding message through network.

Flow control determines how a network's resource such as channel bandwidth, buffer capacity, and control state, are allocated to packets traversing the network. A good flow-control method allocates these resources in an efficient manner so the network achieves a high fraction of its ideal bandwidth and delivers packets with low predictable latency. On the other hand, a poor flow-control method wastes channel bandwidth by leaving resources idle and doing unproductive work with other resources. This results in a network which only a tiny fraction of the ideal bandwidth is realized and has high and variable latency.

Flow control is tightly coupled with buffer management algorithms that determine how messages are handled when blocked in the network. One can view flow control as either a problem of resource allocation or one of contention resolution. From the resource allocation perspective, resources in the form of channels, buffers, and state must be allocated to each packet as it advances from the source to the destination.

Flow control is a synchronization protocol for transmitting and receiving a unit of information. The unit of flow control refers to that portion of the message whose transfer must be synchronized. This unit is defined as the smallest unit of information whose transfer is requested by the sender and acknowledged by the receiver. The request/acknowledgement signaling is used to ensure successful transfer and the availability of buffer space at the receiver.

3.2 Bufferless Flow Control

The simplest flow-control mechanisms are bufferless, and rather than temporarily storing blocked packets, they either drop or misrouter these packets. These forms of flow control use no buffering and simply act to allocated channel state and bandwidth to competing packet. In these cases, the flow-control methods must perform an arbitration to decide which packet gets the channel it has requested. The arbitration method must also decide how to dispose of any packets that did not get their requested destination. Since there are no buffers, we cannot hold the losing packets until their channels become free. Instead we must either drop them or misroute them.

3.2.1 Circuit Switching Flow Control

The next step up in complexity and efficiency is circuit switching, where only packet headers are buffered. In circuit switching, the header of a packet traverses the network ahead of any packet payload, reserving the appropriate resources along the path. If the header cannot immediately allocate a resource at a particular node, it simply waits at that node until the resource becomes free. Once the entire path, or circuit, has been reserved, data may be sent over the circuit until it is torn down by

deallocating the channels.

Circuit switching is a form of bufferless flow control that operates by first allocating channels to form a circuit from source to destination and then sending one or more packets along this circuit. When no further packets need to be sent, the circuit is deallocated. Circuit switching differs from dropping flow control in that if the request flit is blocked, it is held in place rather than dropped. Compared to dropping flow control, circuit switching has the advantage that it never wastes resource by dropping a packet. Because it buffers the header at each hop, it always makes forward progress. However, circuit switching does have two weaknesses that make it less attractive than buffered flow control methods: high latency and low throughput. Circuit switching has advantage of being very simple to implement. All of these flow-control mechanisms are rather inefficient because they waste costly channel bandwidth to avoid using relatively inexpensive storage space.

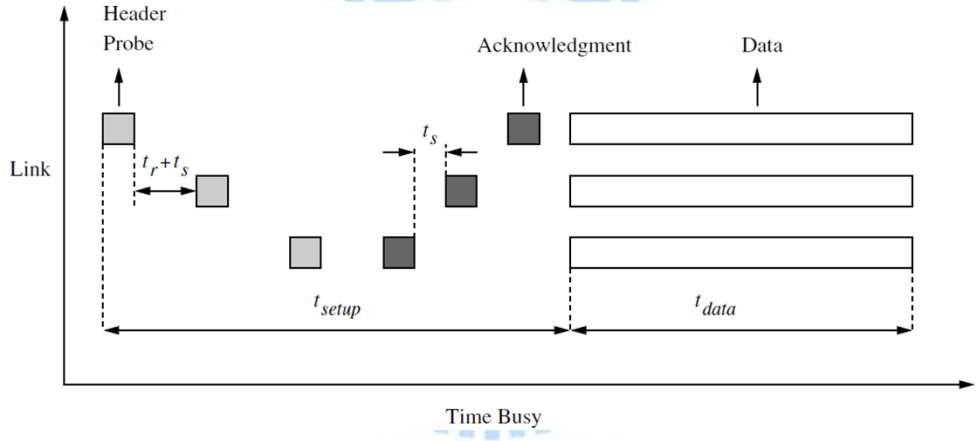


Figure 3.1 Time-space diagram of a circuit-switched message [3.1]

For each switching technique we will consider the computation of the base latency of an L-bit message in the absence of any traffic. The flit size is assumed to be equivalent and equal to the physical data channel width of W bits. The routing header is assumed to be 1 flit; thus the message size is L + W bits. A router can make two routers operates at B Hz; that is the physical channel bandwidth is BW bits per second. We assume that channel wires are short enough to complete a transmission in one clock cycle. Therefore, the propagation delay across this channel is denoted by $t_w = \frac{1}{B}$. Once a path has been set up through the router, the intrarouter delay or switching delay is denoted by t_s . The source and destination processors are assumed to be D links apart. We would continue using these representations later.

The base latency of a circuit-switched message is determined by the time to set up a path and the subsequent time path is busy transmitting data. From Figure 3.1 we can write an expression for the base latency of a message as follows:

$$t_{circuit} = t_{setup} + t_{data}$$

$$t_{setup} = D[t_r + 2(t_s + t_w)]$$

$$t_{data} = \frac{1}{B}\left[\frac{L}{W}\right]$$

3.3 Buffered Flow Control

Adding buffers to our networks results in significantly more efficient flow control. This is because a buffer decouples the allocation of adjacent channels. Without a buffer, the two channels must be allocated to packet (or flit) during consecutive cycles, or the packet must be dropped or misrouted. There is nowhere else for the packet to go. Adding a buffer gives us a place to store the packet (or flit) while waiting for the second channel, allowing the allocation of the second channel to be delayed without complications.

Once we add buffers to an interconnection network, our flow control mechanism must allocate buffers as well as channel bandwidth. Moreover, we have a choice as to the granularity at which we allocate each of these resources. As depicted in Figure 3.2, we can allocate either buffers or channel bandwidth in units of flits or packets. Allocating storage in units of flits rather than packets has three major advantages. It reduces the storage required for correct operation of a router, it provides stiffer backpressure from a point of congestion to the source, and it enables more efficient use of storage.

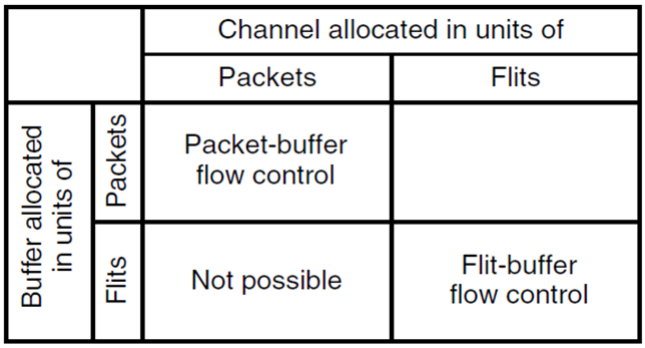


Figure 3.2 Buffered flow control methods can be classified based on their granularity of channel bandwidth allocation and buffer allocation [3.1]

3.3.1 Packet-Buffer Flow Control

If we allocate both channel bandwidth and buffers in units of packets, we have packet-buffer flow control. Storing a flit (or a packet) in a buffer allows us to decouple allocation of the input channel to a flit from the allocation of the output channel to a flit. Adding a buffer prevents the waste of the channel bandwidth caused by dropping or misrouting packets or the idle time inherent in circuit switching. As a result, we can approach full channel utilization with buffered flow control.

3.3.1.1 Store-and-Forward Flow Control

A packet is completely buffered at each intermediate node before it is forwarded to the next node. This is the reason why this switching technique is also referred to as store-and forward (SAF) switching. The packet must be allocated two resources before it can be forwarded: a packet-sized buffer on the far side of the channel and exclusive use of the channel. Once the entire packet has arrived at a node and these two resources are acquired, the packet is forwarded to the next node. While waiting to acquired resources, if they are not immediately available, no channels are being held idle and only a single packet buffer on the current node is occupied.

Packet switching [3.2] is advantageous when messages are short and frequent. Unlike circuit switching, where a segment of a reserved path may be idle for a significant period of time, a communication link is fully utilized when there are data to be transmitted. The major drawback of store-and-forward flow control is its very

high latency. Since the packet is completely received at one node before it can begin moving to the next node, serialization latency is experienced at each hop.

The base latency of a packet-switched message can be computed as follow:

$$t_{packet} = D\left\{t_r + (t_s + t_w)\left[\frac{L + W}{W}\right]\right\}$$

The important point to note is that the latency is directly proportional to the distance between source and destination nodes.

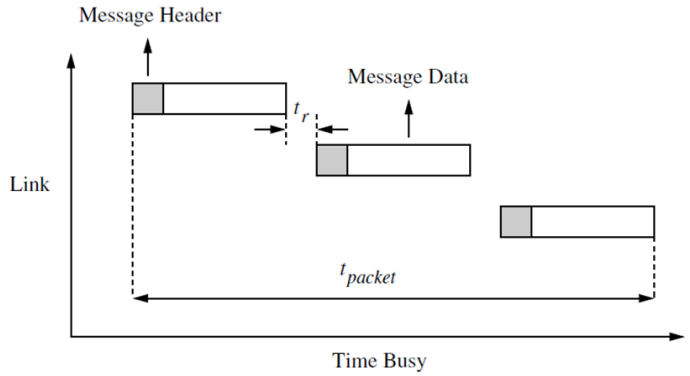


Figure 3.3 Time-space diagram of a packet-switched message [3.1]

3.3.1.2 Virtual Cut Through (VCT) Flow Control

Packet switching is based on the assumption that a packet must be received in its entirety before any routing decision can be made and the packet forwarded to the destination. Rather than waiting for the entire packet to be received, the packet header can be examined as soon as it is received. The router can started forwarding the header and following data bytes as soon as routing decisions have been made and the output buffer is free. In fact the message does not even have to be buffered at the output and can cut through to the input of the next router before the complete packet has been received at the current router. This switching technique is referred to as virtual cut-through switching (VCT). In the absence of blocking, the latency experienced by the header at each node is the routing latency and propagation delay through the router and along the physical channels. If the header is blocked on a busy output channel, the complete message is buffered at the node. Thus, at high network loads, VCT switching behaves like packet switching.

Cut-through flow control overcomes the latency penalty of store-and-forward flow control by forwarding a packet as soon as the header is received and resources (buffer and channel) are acquired, without waiting for the entire packet to be received. As with store-and-forward flow control, cut-through flow control allocates both buffers and channel bandwidth in units of packets. It differs only in that transmission over each hop is started as soon as possible without waiting for the entire packet to be received.

Virtual cut through flow control reduced the latency from the product of the hop count and the serialization latency. At this point, cut-through flow control may seem like an ideal method. It gives very high channel utilization by using buffers to decouple channel allocation. It also achieves very low latency by forwarding packets as soon as possible. However, the cut-through method, or any other packet-based method, has two serious shortcomings. First, by allocating buffers in units of packets, it makes very inefficient use of buffer storage. As we shall see, we can make much more effective use of storage by allocating buffers in units of flits. This is particularly important when we need multiple, independent buffer sets to reduce blocking or provide deadlock avoidance. Second, by allocating channels in units of packets, contention latency is increased. For example, a high-priority packet colliding with a low-priority packet must wait for the entire low-priority packet to be transmitted before it can acquire the channel. In the next section, we will see how allocating resources in units of flits rather than packets results in more efficient buffer use (and hence higher throughput) and reduced contention latency.

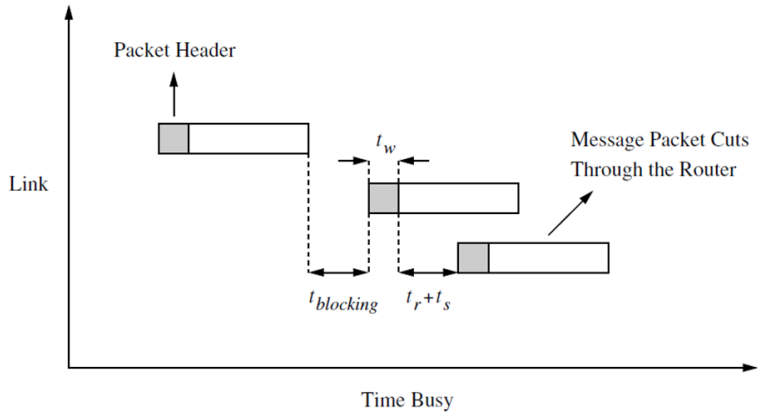


Figure 3.4 Time-space diagram of a virtual cut-through switched message [3.1]

The base latency of a message that successfully cuts through each intermediate

router can be computed as follows:

$$t_{vct} = D(t_r + t_s + t_w) + \max(t_s, t_w)\left\lceil \frac{L}{W} \right\rceil$$

Cut-through routing is assumed to occur at the flit level with the routing information contained in 1 flit. This model assumed that there is no time penalty for cutting through a router if the output buffer and output channel are free. Depending on the speed of operation of the routers, this may not be realistic. Note that the header experiences routing delay, as well as the switching delay and wire delay at each router. This is because the transmission is pipelined and the switched is buffered at the input and output. Once the header flit reaches the destination, the cycle time of this message pipeline is determined by the maximum of the switching delay and wire delay between routers. If the switch had been buffered only at the input, then in one cycle of operation, a flit traverses the switch and channel between routers. In this case, the coefficient of the second term and the pipeline cycle time would be $(t_s + t_w)$. Note that the unit of message flow control is a packet. Therefore, even though the message may cut through the router, the sufficient buffer space must be allocated for a complete packet in case the header is blocked.

3.3.2 Flit-Buffer Flow Control

3.3.2.1 Wormhole Flow Control

The need to buffer complete packets within a router can make it difficult to construct small, compact, and fast router. Wormhole flow control operates like cut-through, but with channel and buffers allocated to flits rather than packets. In wormhole switching, the buffer requirements within the routers are substantially reduced over the requirement for VCT switching. The primary difference between wormhole switching and VCT switching is that, in the former, the unit of message flow control is a single flit and, as a consequence, small buffers can be used.

Compared to cut-through flow control, wormhole flow control makes far more efficient use of buffer space, as only a small number of flit buffers are required per

virtual channel. In contrast, cut-through flow control requires several packets of buffer space, which is typically at least an order of magnitude more storage than wormhole flow control. This savings in buffer space, however, comes at the expense of some throughput, since wormhole flow control may block a channel mid-packet. Blocking may occur with wormhole flow control because the channel is owned by a packet, but buffers are allocated on a flit-by-flit basis.

The base latency of a wormhole-switched message can be computed as follows:

$$t_{wormhole} = D(t_r + t_s + t_w) + \max(t_s, t_w)\left\lceil \frac{L}{W} \right\rceil$$

This expression assumes flit buffers at the router inputs and outputs. Note in the absence of contention, VCT and wormhole switching have the same latency. Once the header flit arrives at the destination, the message pipeline cycle time is determined by the maximum of the switch delay and wire delay. For an input-only and output-only buffered switch, this cycle time would be given by the sum of the switch and wire delays.

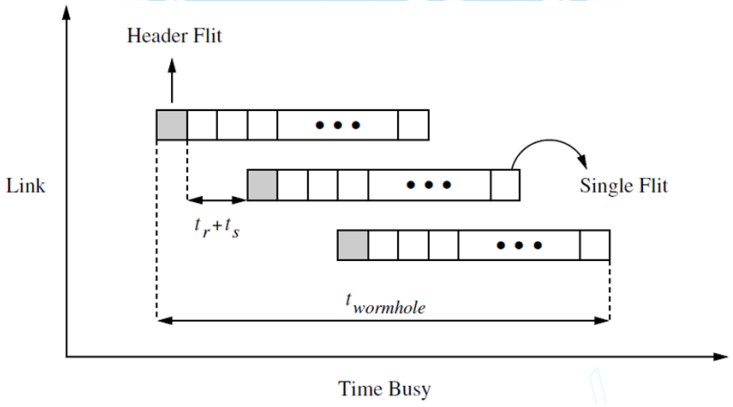


Figure 3.5 Time-space diagram of a wormhole-switched message [3.1]

3.3.2.2 Virtual Channel Flow Control

The preceding switching techniques were described assuming that messages or parts of messages were buffered at the input and output of each physical channel. Buffers are commonly operated as FIFO queues. Therefore, once a message occupies a buffer for a channel, no other message can access the physical channel, even if the message is blocked. Alternatively, a physical channel may support several logical or

virtual channels multiplexed across physical channel. Each unidirectional virtual channel is realized by an independently managed pair of message buffers as illustrated in Figure 3.6. Consider wormhole switching with a message in each virtual channel [3.3]. Each message can share the physical channel on a flit-by-flit basis. The physical channel protocol must be able to distinguish between the virtual channels using the physical channel. Logically, each virtual channel operates as if each were using a distinct physical channel operating half the speed. Virtual channel were originally introduced to solve the problem of deadlock in wormhole-switched networks. Deadlock is a network state where no messages can advance because each message requires a channel occupied by another message. By allowing messages to share a physical channel, messages can make progress rather than remain blocked. Virtual channels can also be used to improve message latency and network throughout. Virtual-channel flow control decouples the allocation of channel state from channel bandwidth. This decoupling prevents a packet that acquires channel state and then blocks from holding channel bandwidth idle. This permits virtual-channel flow control to achieve substantially higher throughput than wormhole flow control.

As in wormhole flow control, an arriving head flit must allocate a virtual channel, a downstream flit buffer, and channel bandwidth to advance. Subsequent body flits from the packet use the virtual channel allocated by the header and still must allocate a flit buffer and channel bandwidth. However, unlike wormhole flow control, these flits are not guaranteed access to channel bandwidth because other virtual channels may be competing to transmit flits of their packets across the same link.

In fact, given the same total amount of buffer space, virtual-channel flow control also outperforms cut-through flow control because it is more efficient to allocate buffer space as multiple short virtual-channel flit buffers than as a single large cut-through packet buffer.

We can envision continuing to add virtual channels to further reduce the blocking experienced by each message. The result is increased network throughput measured in flits/s, due to increased physical channel utilization. However, each additional virtual channel improves performance by a smaller amount, and the increase channel multiplexing reduces the data rate of individual messages, increasing the message latency. This increase in latency due to data rate multiplexing will eventually

overshadow the reduction in latency due to blocking, leading to overall increasing average message latency.

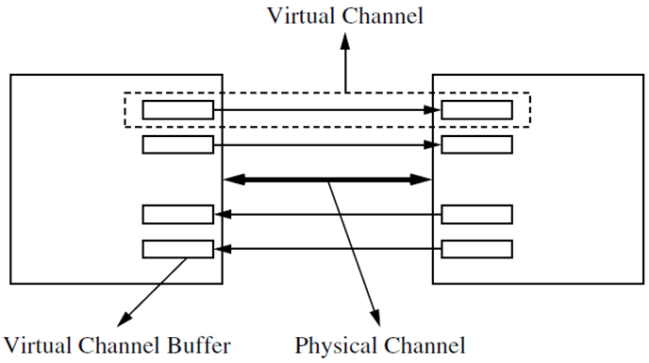


Figure 3.6 Virtual channels [3.1]

3.3.3 Buffer Management and Backpressure

All of the flow control methods that use buffering need a means to communicate the availability of buffers at the downstream nodes. Then the upstream nodes can determine when a buffer is available to hold the next flit (or packet for store-and-forward or cut-through) to be transmitted. This type of buffer management provides backpressure by informing the upstream nodes when they must stop transmitting flits because all of the downstream flit buffers are full. Three types of low-level flow control mechanisms are in common use today to provide such backpressure [3.4]: credit-based [3.5], on/off, and ack/nack [3.6]. We examine each of these in turn.

3.3.3.1 Credit-Based Flow Control

With credit-based flow control [3.7][3.8][3.9], the upstream router keeps a count of the number of free flit buffers in each virtual channel downstream. Then, each time the upstream router forwards a flit, thus consuming a downstream buffer, it decrements the appropriate count. If the count reaches zero, all of the downstream buffers are full and no further flits can be forwarded until a buffer becomes available. Once the downstream router forwards a flit and frees the associated buffer, it sends a credit to the upstream router, causing a buffer count to be incremented.

From Figure 3.7 we can see that the minimum time between the credit being sent at time t_1 and a credit being sent for the same buffer at time t_5 is the credit round-trip delay t_{crt}. This delay, which includes a round-trip wire delay and additional processing time at both ends, is a critical parameter of any router because it determines the maximum throughput that can be supported by the flow control mechanism.

A potential drawback of credit-based flow control is a one-to-one correspondence between flits and credits. For each flit sent downstream, a corresponding credit is eventually sent upstream. This requires a significant amount of upstream signaling and, especially for small flits, can represent a large overhead.

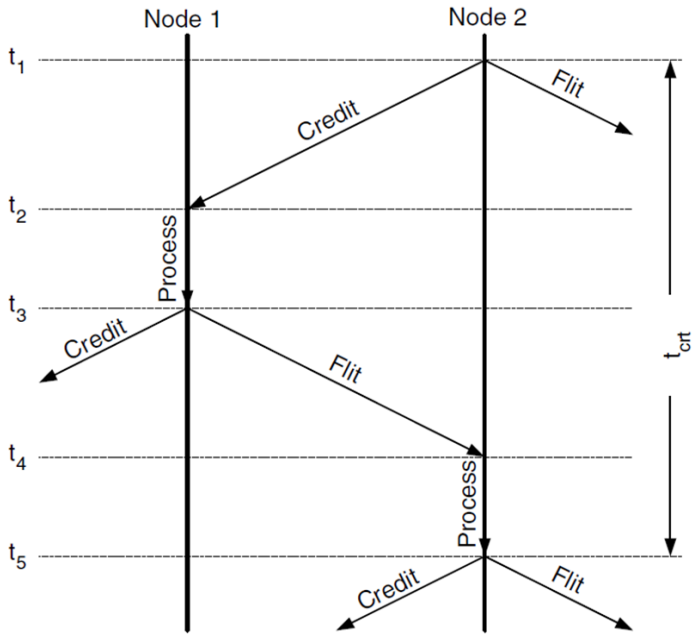


Figure 3.7 Timeline of credit-based flow control [3.9]

3.3.3.2 On/Off Flow Control

On/off flow control can greatly reduce the amount of upstream signaling in certain cases. With this method the upstream state is a single control bit that represents whether the upstream node is permitted to send (on) or not (off). A signal is sent upstream only when it is necessary to change this state. An off signal is sent when the control bit is on and the number of free buffers falls below the threshold X_{off}. If the control bit is off and the number of free buffers rises above the threshold

X_{on}, an on signal is sent. A timeline illustrating on/off flow control is illustrated in Figure 3.8. With an adequate number of buffers, on/off flow control systems can operate with very little upstream signaling.

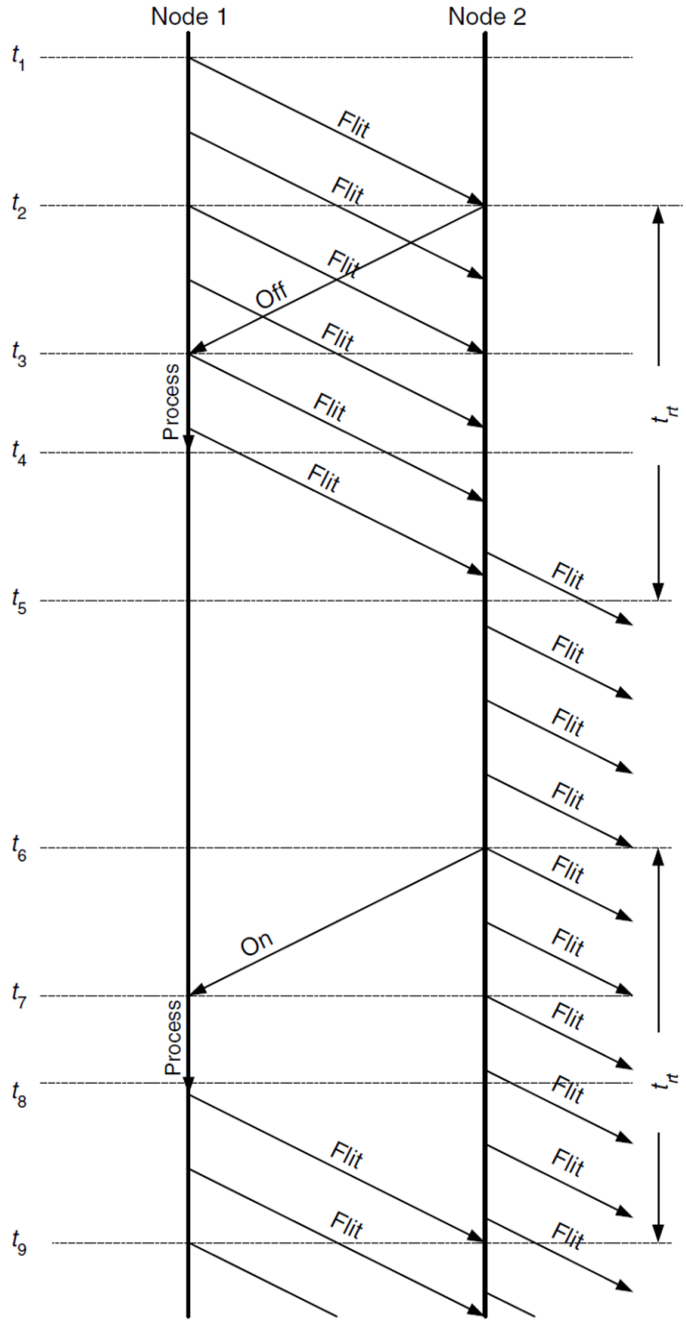


Figure 3.8 Timeline of on/off flow control [3.9]

3.3.3.3 Ack/Nack Flow Control

Both credit-based and on/off flow control require a round-trip delay t_{rt} between

the time a buffer becomes empty, triggering a credit or an on signal, and when a flit arrives to occupy that buffer. Ack/nack flow control reduces the minimum of this buffer vacancy time to zero and the average vacancy time to $t_{rt}/2$. Unfortunately there is no net gain because buffers are held for an additional t_{crt} waiting for an acknowledgment, making ack/nack flow control less efficient in its use of buffers than credit-based flow control. It is also inefficient in its use of bandwidth which it uses to send flits only to drop them when no buffer is available. With ack/nack flow control, there is no state kept in the upstream node to indicate buffer availability. The upstream node optimistically sends flits whenever they become available. If the downstream node has a buffer available, it accepts the flit and sends an acknowledge (ack) to the upstream node. If no buffers are available when the flit arrives, the downstream node drops the flit and sends a negative acknowledgment (nack). The upstream node holds onto each flit until it receives an ack. If it receives a nack, it retransmits the flit.

Because of its buffer and bandwidth inefficiency, ack/nack flow control is rarely used. Rather, credit-based flow control is typically used in systems with small numbers of buffers, and on/off flow control is employed in most systems that have large numbers of flit buffers.

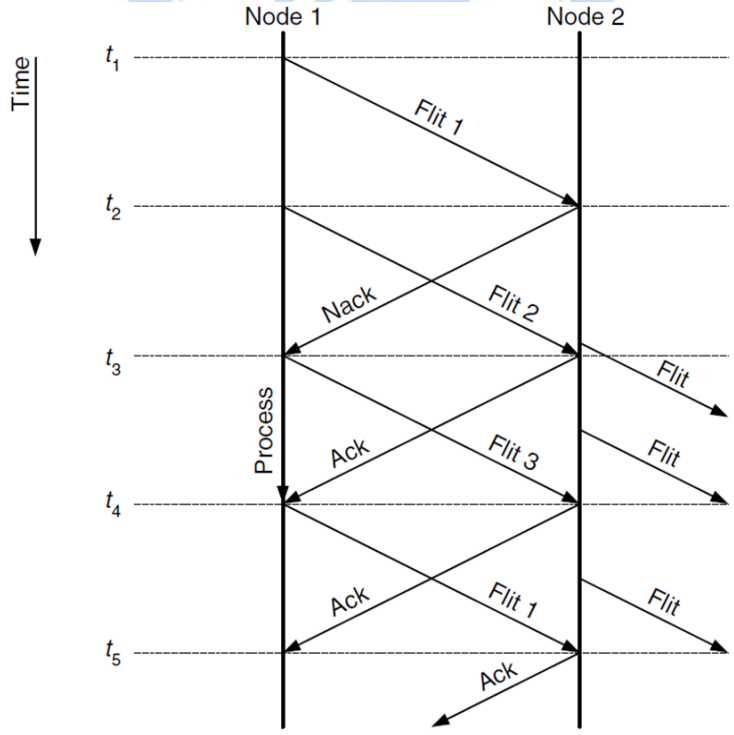


Figure 3.9 Timeline of ack/nack flow control [3.9]

3.3.4 Flit-Reservation Flow Control

While traditional wormhole networks greatly reduce the latency of sending packets through an interconnection network, the idealized view of router behavior can differ significantly from a pipelined hardware implementation. Pipelining breaks the stages of flit routing into several smaller steps, which increases the hop time. Accounting for these pipelining delays and propagation latencies gives an accurate view of buffer utilization.

The remaining time required to recycle the credit and issue another flit to occupy the buffer is called the turnaround time. Lower buffer utilization reduces network throughput because fewer buffers are available for bypassing blocked messages and absorbing traffic variations. Flit-reservation flow control can reduce turnaround time to zero and hide the flit pipeline delay in a practical implementation. Flit-reservation hides the overhead associated with a pipelined router implementation by separating the control and data networks. Control flits race ahead of the data flits to reserve network resources. As the data flits arrive, they have already been allocated an outgoing virtual channel and can proceed with little overhead. Reservation also streamlines the delivery of credits, allowing zero turnaround time for buffers. Of course, it is not always possible to reserve resources in advance, especially in the case of heavy congestion. In these situations, data flits simply wait at the router until resources have been reserved which is the same behavior of a standard wormhole router.

3.4 Memory-Centric On-Chip Interconnection Network for Heterogeneous Multi-Core SoC

NoCs contain very wide range of area that we describe in the chapter 2. Figure 3.10 shows the abstraction levels of NoCs. In this thesis we focus on transport (repacketization), switching circuit and arbitration. In this section we would describe the switching protocol between network interface and router (crossbar) and arbitration mechanism. Moreover, later we will describe network interface including buffers (queues), message transport (repacketization), switching policy between processor

element and network interface in the chapter 4.

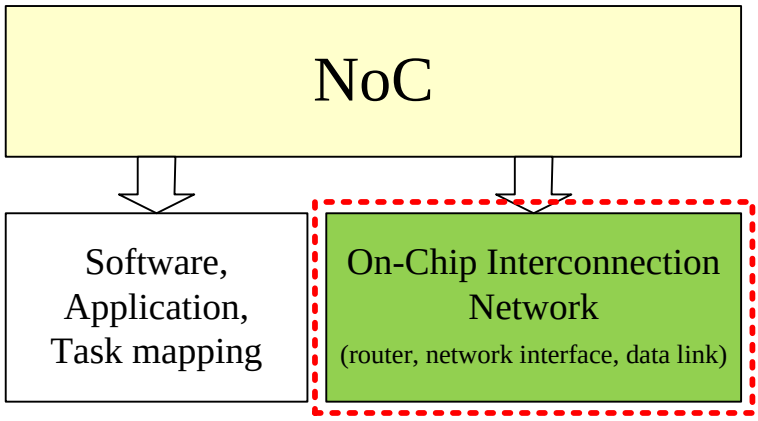


Figure 3.10 abstraction levels of NoC

The performance of a multi-core SoC [3.10] will be limited by the on-chip data communication, memory resource allocation, and memory data accessing. Therefore, a memory-centric on-chip interconnection network as shown in Figure 3.11 is presented for improving both the communication bandwidth and memory bandwidth. For each PE, a NI and a d-MMU with distributed memories is developed to communicate with the OCIN and on-demand memory subsystem, respectively. If a process element requires additional memory resources, the centralized memory resources, including centralized SRAMs and off-chip DRAM, can be utilized. The centralized memory resources are managed by a c-MMU that dynamically allocates and manages the memory resources based on different memory requirements. Therefore, the on-demand memory subsystem can efficiently handle all memory requests generated by PEs.

Memory-centric on-chip interconnection network includes a network interface (NI) for data switching between processor element and network interconnection (crossbar switch) and a network interconnection (crossbar switch) for data switching among each request with different priority from other node and routing and arbitration.

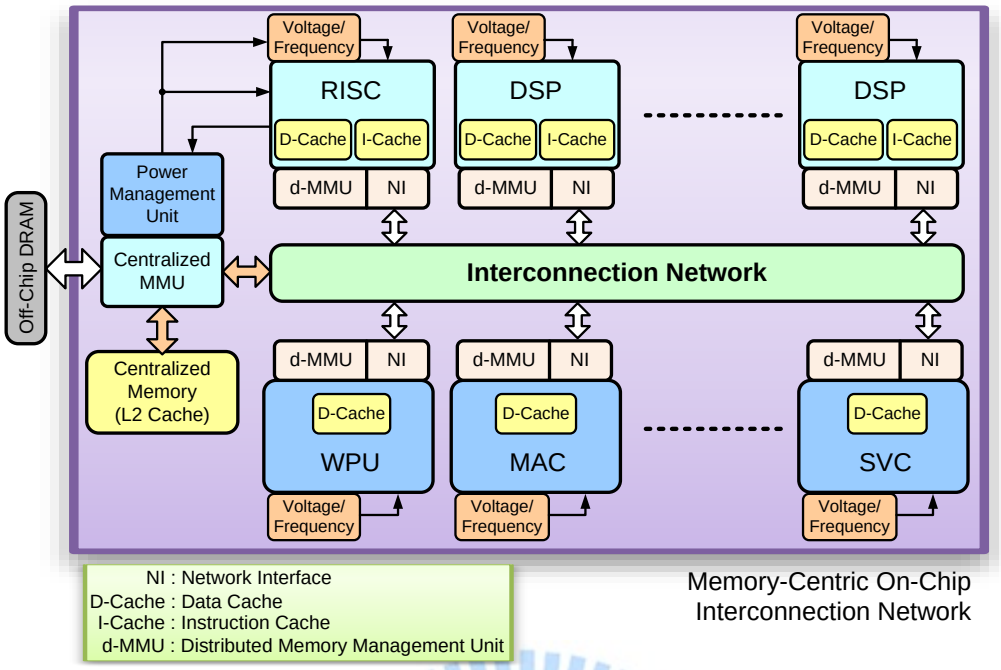


Figure 3.11 Memory-Centric On-Chip Interconnection Architecture.

We will introduce network interface (NI) later in the chapter 4.

3.4.1 Network Interconnection

3.4.1.1 Data Switching Protocol between Crossbar and NI

This router is a 4×4 crossbar. Crossbar switch receives transmit request from each network interface (NI) and decides which node can get grant to use. Table 3.1 shows input and output description between NI and interconnection network (crossbar).

I/O	Port name	Width	Description
Input	N_TX_REQ	1	Request transmit grant: Active high when transfer header flit
Input	N_TX_PRI	2	Packet priority: PRI signal represents the priority of transmitting data packet. It is used to determine the priority of network arbitration. The default value is 0 (highest priority)

Output	N_TX_GRANT	1	Grant to transfer: Active high when arbiter give the transmit grant
Input	N_DATA_RDY	1	Data ready: Active high when data is ready for transmit
Input	N_TX_DATA	34	Transmit data: include flit type (2bits) and data (32bits)
Output	N_STALL	1	Transmit stall: Active high when buffer queue of receiver end has no free slot under grant to transfer condition
Input	N_num_free	1	Number free slot: Active high when buffer queue of receiver end has free slot
Output	N_transmit	1	Transmit: Active high when transmit data to receiver
Output	N_type_out	2	Packet flit type: value 0 for header flit, value 1 for body flit and value 2 for tail flit
Output	N_data_out	32	Transmit data:

Table 3.1 Input and output description between network interface and crossbar

When requests from each network interface compete to get transmit grant, arbiter would decide which is winner according to packet priority. We would introduce the arbitration mechanism later in next section. Network interface sends transmit request and packet priority to crossbar and put header flit simultaneously. When crossbar is ready for next transmit, crossbar will give grant to winner node and receive header flit. When network interface gets grant to transmit, network interface activates data ready signal and puts data flit.

If free slot of buffer queue in the receiver end is not equal to zero, crossbar will transfer data in order to destination node. Otherwise, crossbar must activate stall signal to inform network interface suspend to transmit. In this case, crossbar will receive current data and transmit to destination node. When sender of network interface receives stall signal, network interface would suspend transmit until stall signal is released. Network interface would continue uncompleted transmit. The timing diagram is showed in the Figure 3.12. Interface definition is showed in Figure 3.13.

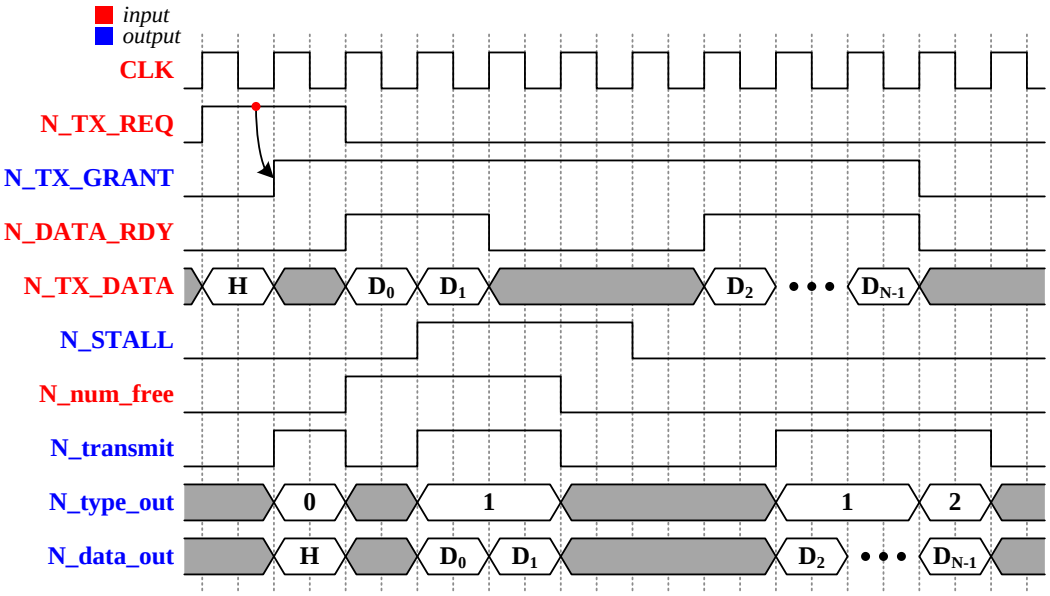


Figure 3.12 Timing diagram between crossbar and NI.

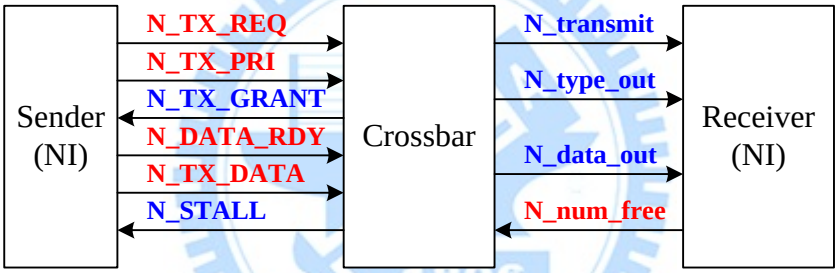


Figure 3.13 Interface definition between Network Interface and crossbar.

3.4.1.2 Arbitration Mechanism for Router (Crossbar)

When crossbar (interconnection network) receives request from network interface, crossbar must arbitrate which node get grant first. Our arbitration mechanism according to packet priority and grant order is very fair. Packet priority is defined by processor element and grant order is decided by present transmit. For example, we assume the current grant order is (node1, node2, node3). In the case that node1, node2 and node3 request to transfer to node0, we would give grant to node1 and set grant order to (node2, node3, node1). This setting can avoid the high priority node occupying transfer channel which leads to low priority node always cannot get grant to use channel.

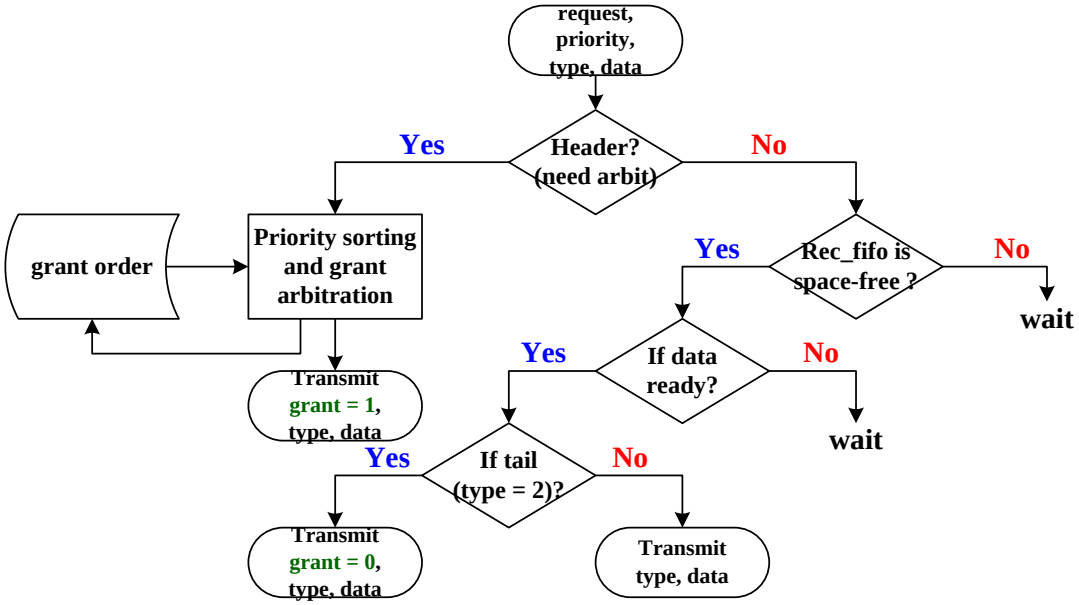


Figure 3.14 Flow chart of arbitration.

Figure 3.14 shows flow chart of arbitration. First, crossbar receives request from network interface (sender), arbiter would decide which node get grant and set new grant order for next arbitration. Then, crossbar would pass data from source node to destination node if no stall condition.

3.5 Summary

This chapter focuses on data switching policy between network interface and router and arbitration mechanism of router. We used wormhole switching policy to reduce the congestion condition and improve data transfer performance. About the arbitration mechanism, we consider a fairness policy to avoid high priority data occupying communication channel lead to starvation of low priority data.

Chapter 4
An Efficient Network Interface for Memory-Centric On-Chip Interconnection Network

The performance of a multi-core system-on-chip (SoC) will be limited by the on-chip data communication, memory resource allocation, and memory data accessing. A memory-centric on-chip interconnection network (OCIN) is presented to improve both the communication bandwidth and memory bandwidth. Moreover, network interfaces (NIs), one of the building block of OCINs, is a major factor in the performance. In this chapter, an efficient NI is proposed for the memory-centric on-chip interconnection network to reduce the data blocking by a borrowing mechanism. By considering the borrowed memory blocks and distributed memory management unit (d-MMU), the size of the output queue in NI can be dynamically scheduled.

4.1 Introduction

NI is designed as a bridge between a process element and a router. Additionally, NI is a major factor in the performance of OCINs, and implements the communication protocols of OCINs [4.1]. NI is implemented based on the message dependencies in shared–memory and message-passing communication paradigms. Therefore, a NI micro-architecture was proposed for the connection-then-credit (CTC) flow control protocol among both communication paradigms [4.2]. Moreover, an efficient NI was investigated to offer guaranteed services, shared-memory abstraction and flexible network configuration [4.3]. These two approaches increase design reuse and allow flexible instantiations in different design constraints. In addition to meet hare real-time constraints, a communication and configuration controller was developed to manage reconfiguration data-flows in NIs [4.4]. Furthermore, a high-speed NI was

proposed to support the serial-link packet-based transmission model [4.5].

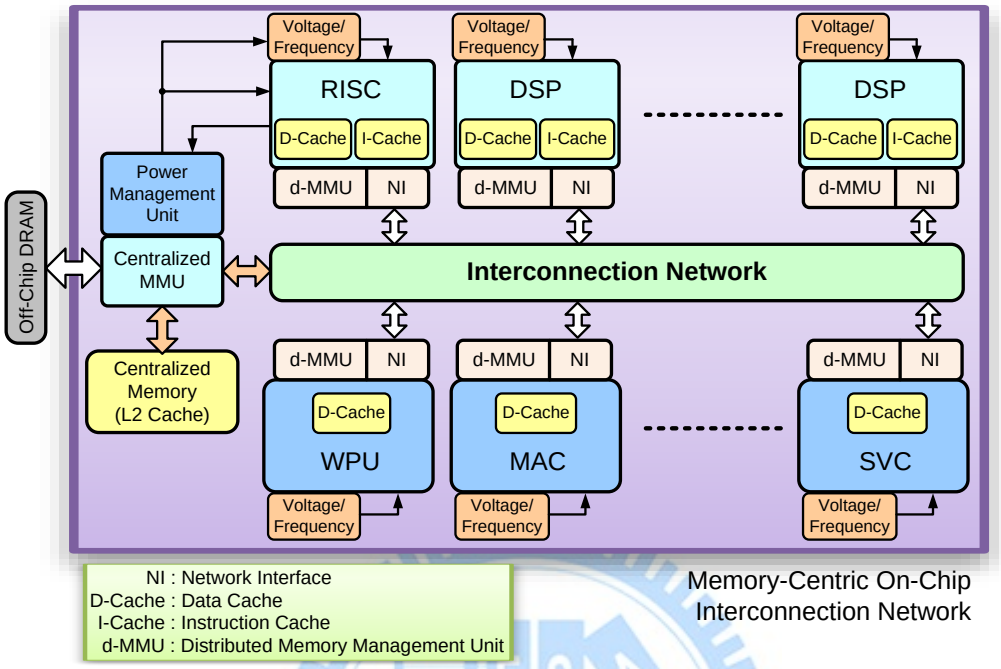


Figure 4.1 Memory-centric on-chip interconnection network (OCIN)

With the increasing PEs, data coherent of the shared–memory communication becomes one of the critical design challenges. Therefore, the message-passing communication paradigm is popular in multi-core systems [4.6][4.7]. Additionally, with increasing demands on ubiquitous wireless high-data-rate multimedia services, large amounts of high speed and low power memories are indispensable for a multi-core platform. Therefore, on-chip SRAM-rich SoCs and processors was proposed for multimedia devices, and a memory-access-specific OCIN was developed [4.8]. In view of this, a memory-centric OCIN as shown in Figure 4.1 will be an effective platform for the future wireless multimedia systems. The memory-centric OCIN is developed by a conventional OCIN and an on-demand memory sub-system, including distributed memories, centralized memories, distributed memory management units (d-MMUs) and a centralized MMU (c-MMU). In this paper, an efficient NI for the memory-centric OCIN is proposed to reduce the data blocking in NIs via the d- MMU. The d-MMU can dynamically allocate the memory resources for buffering the blocking network data.

4.2 Memory-Centric On-Chip Interconnection Network

4.2.1 Packet Definition

The message-passing communication paradigm is adopted among the PEs via the packet switching technique. Therefore, Figure 4.2 presents the packet definition of the memory-centric on-chip interconnection network. The header field in the header flit contains the information of this packet to route this packet to the destination in the OCIN. Additionally, “Mes” indicates this packet is routed for message-passing or on-demand memory subsystem. Therefore, the message-passing field or the on-demand memory field indicates the control signals between PEs or d-MMU and c-MMU. Furthermore, if both “Mes” and “Addr” are activated, the control signals between two PEs are extended in the address field. In additional, “R/W” indicates the access operation of this packet and “Pr” indicates the packet priority. “BL” indicates the burst length of packet, in other words, it represents the size of payload flits.

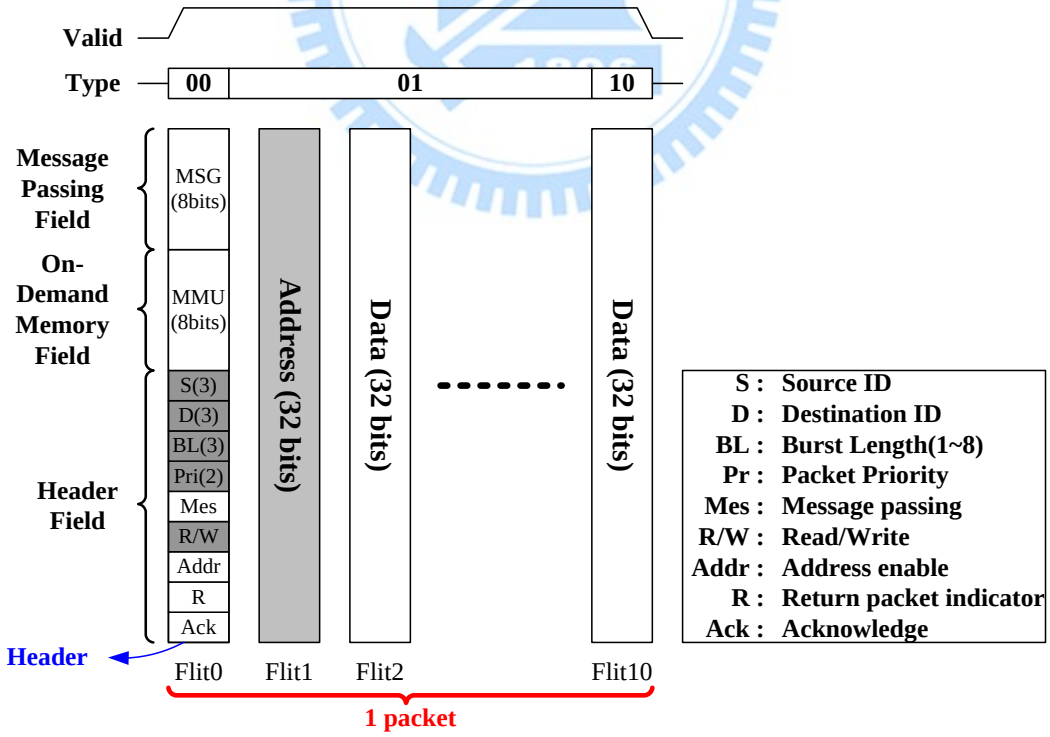


Figure 4.2 Packet definition of the memory-centric on-chip interconnection network

4.2.2 Data Communication Protocol

In on-demand memory platform, a specific memory access and data communication protocol has been defined to integrate all processor elements. Base on this protocol, processor element can use large memory space and communicate with other processor elements. This section will describe the detail information about the protocol, including I/O definition and all kind of operation behaviors.

In order to finish high level verification, a simple MMU behavior model has been established to simulate memory access and data communication behavior. Figure 4.3 shows the block diagram of the simulation model. In this diagram, MMU behavior model acts as memory system behavior including memory access, data transmit, data receive and special MMU operation. In this section, these will be described clearly. According to this protocol, processor element providers need to establish a wrapper to execute memory access, data transmit and data receive successfully.

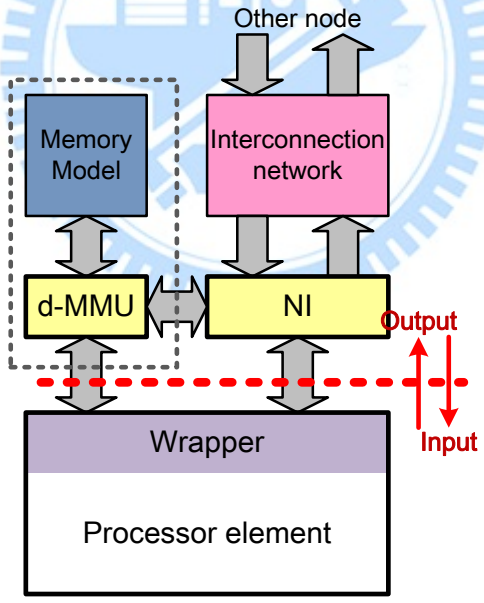


Figure 4.3 Simulation model block diagram

In the Figure 4.3, the red line indicates the I/O port interface and I/O direction (based on wrapper). The detail input and output port definition between d-MMU and wrapper will be introduced in the Table 4.1 and Table 4.2 respectively.

I/O	*Category*	*Port name*	*Width*	*Description*
Input	Memory R/W operation	M_IN_VALID	1	***Memory data input enable signal*** : Active high when memory data inputs are valid
		M_RDATA	32	***Memory data input*** : It will be read data from memory
		M_IN_BL	3	***Memory input data burst length*** : The burst length value is IN_BL + 1. For example, IN_BL = 3'b000 represents the burst length is 1, and 3'b111 represents the burst length is 8
		MA_READY	1	***MMU memory access ready signal*** : Active high when the d-MMU is ready to service memory access
Input	MMU operation	MU_READY	1	***MMU operation ready signal*** : Active high when the d-MMU is ready to service MMU operation
	Transmit operation	TX_READY	1	***MMU transmit ready signal*** : Active high when the d-MMU is ready to service transmit operation
	Receive operation	RX_IN_VALID	1	***Receive operation input enable signal*** : Active high when receive operation data input are valid
		RX_RW	1	***Receive operation Read/Write signal*** : Indicate the receive operation is receiving data or receiving request 0 : Receiving data 1 : Receiving request
		RX_IN_BL	3	***Receive data burst length*** : The burst length value is IN_BL + 1. For example, IN_BL = 3'b000 represents the burst length is 1, and 3'b111 represents the burst length is 8
		RX_SOURCE	2	***Input data source*** : Indicate the source of the receiving data. The node ID is shown below 0 : WPU 1 : MAC 2 : LT coding 3 : SVC The details will be showed in the chapter 5
		MSG_INFO_IN	8	***Message data information*** : The signal represents the receiving data type and information which are defined by each processor elements

		RX_DATA	32	***Receive data input*** : It will be packet data from other node
	Common	CLK	1	***Clock***
		RST	1	***Reset*** : Synchronous active high reset

Table 4.1 Input description of the wrapper

I/O	***Category***	***Port name***	***Width***	***Description***
output	Memory R/W operation	M_OUT_VALID	1	***Memory data output enable signal*** : Active high when memory access output signals are valid
		M_RW	1	***Memory Read/Write signal*** : Indicate the memory access is read or write 0 : write 1 : read
		M_OUT_BL	3	***Memory access output data burst length*** : The burst length value is IN_BL + 1. For example, IN_BL = 3'b000 represents the burst length is 1, and 3'b111 represents the burst length is 8
		M_ADDR	32	***Read/Write address*** : ADDR signal represents the start address of the continuous burst data
		M_WDATA	32	***Memory write data output*** : This signal will be memory write data.
	Transmit operation	TX_OUT_VALID	1	***Transmit output enable signal*** : Active high when transmit operation output signals are valid
		TX_RW	1	***Transmit operation Read/Write signal*** : Indicate the transmit operation is sending data or sending request 0 : Sending data 1 : Sending request
		TX_OUT_BL	3	***Transmit operation output data burst length*** : The burst length value is IN_BL + 1. For example, IN_BL = 3'b000 represents the burst length is 1, and 3'b111 represents the burst length is 8
		TX_DEST	2	***Transmit operation output data destination*** : Indicate the data destination. The node ID is show below

				0 : WPU 1 : MAC 2 : LT coding 3 : SVC The details will be showed in the chapter 5
		TX_DATA	32	***Transmit data output*** : This signal will be memory write data
		TX_PRI	2	***Packet priority*** : PRI signal represents the priority of transmitting data packet. It is used to determine the priority of network arbitration. The default value is 0 (highest priority)
		MSG_INFO_OUT	8	***Message data information*** : The signal represents the transmitted data type and information which are defined by each processor elements
	Receive operation	RX_CAP	4	***Data capacity*** : This signal indicates available space of the buffer which is utilized to store receiving data. Its maximum value is 8, which indicates available space is equal or larger than 8
	MMU operation	MU_VALID	1	***MMU enable*** : Active high when processor element want to use MMU signal to tell d-MMU specific information
		MU_INFO	8	***MMU information***

Table 4.2 Output description of the wrapper

In the transmit operation, processor elements can transmit data to another node. This operation need to provide destination ID which indicated the data destination. This information is represented by TX_DEST port. The transmit also uses burst transmit, so it need burst length information which is represented by TX_OUT_BL. In addition, specific message information which is predefined by these two nodes will be required to identify the packet data type, so processor element can use MSG_INFO_OUT signal to present it. Besides, PRI signal indicates the packet data priority to determine the packet priority in the interconnect network, and its default value is set to 0 (highest priority). Also, TX_READY signal indicates the MMU transmit state Processor elements need to check this signal first to know about

whether the MMU is ready to serve transmit operations. The detail timing diagram of transmit operation is shown in Figure 4.4 and Figure 4.5.

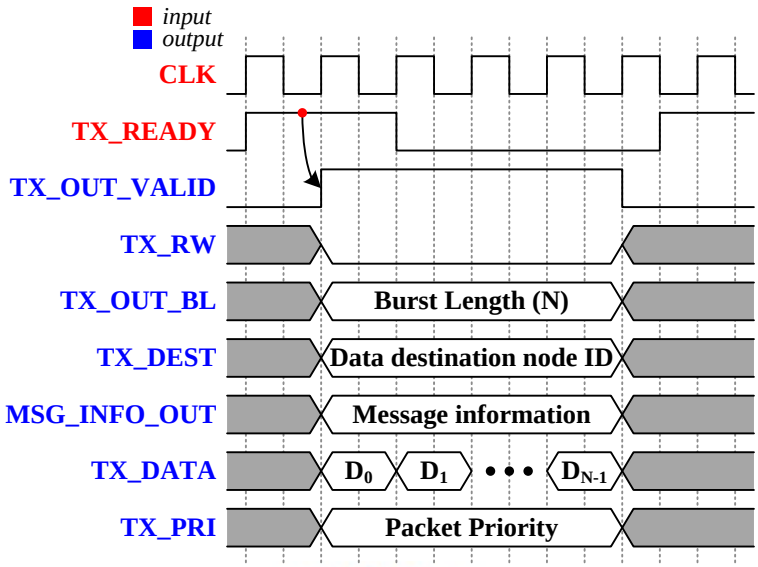


Figure 4.4 Timing diagram of send packet data to other node

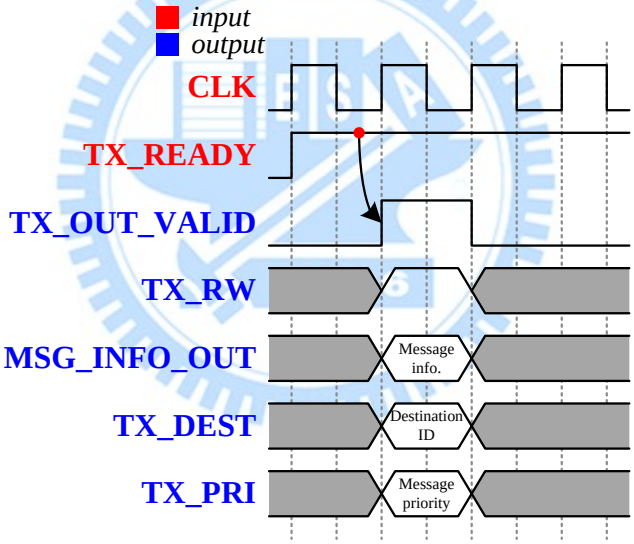


Figure 4.5 Timing diagram of send read request to other node

When another node transmits a packet data to the local node, d-MMU will execute receive operation, and then it forward the packet data to the processor element. In receive operation, RX_IN_BL signal indicates the transmitted packet burst length and RX_SOURCE signal indicates the source ID. Besides, MSG_INFO_IN signal is used to represent specific message information which is predefined by these two nodes. Processor element can use message information to identify received data. Moreover, in order that d-MMU want to check whether the processor element or wrapper has enough capacity to handle these data, RX_CAP signal indicate the

residual space of the received data buffer in the wrapper. D-MMU will check this signal first, and then decide whether the burst data can be send. If it cannot be send, these data will be stored in the d-MMU buffer until the processor element has enough capacity. The detail timing diagram is shown in Figure 4.6 and Figure 4.7.

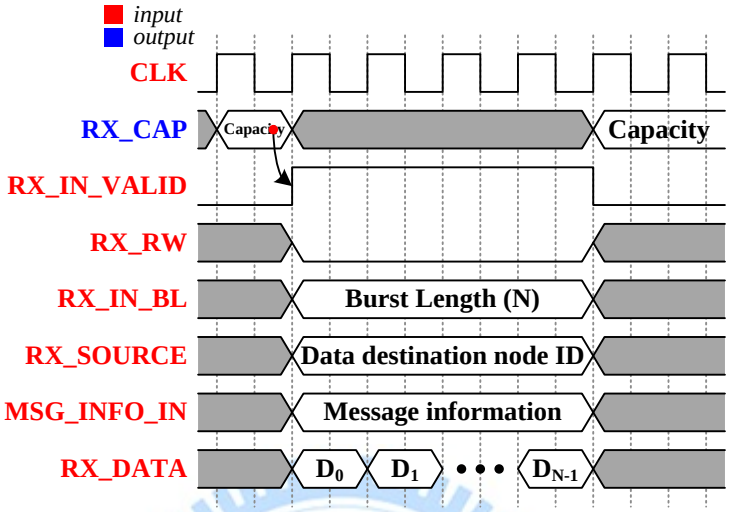


Figure 4.6 Timing diagram of receive packet from other node

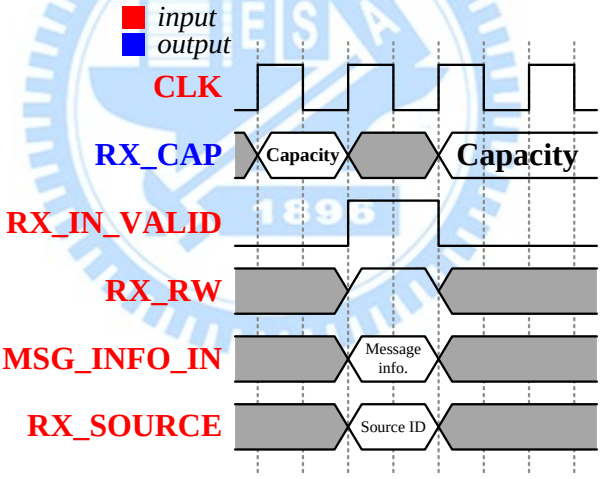


Figure 4.7 Timing diagram of receive request from other node

4.3 Efficient Network Interface for Memory-Centric On-Chip Interconnection Network

For the memory-centric OCIN, d-MMUs are designed for PEs to store the temporal data of their tasks. The d-MMU and distributed memories perform as a low level cache for the dedicated PE in the on-demand memory sub-system. Additionally, NI is designed as a bridge between the PEs and the OCIN. NI contains the input

queue and output queue for buffering packets. However, the sizes of the queues dominate the area and the performance of data transfer. If the buffer is insufficient, the PE will be stall until the head-of-line blocking releases. Therefore, if the utilization of the distributed memory is low, the d-MMU can borrow the memory resources for buffering the blocking packets from the PEs, and the PEs can keep computing for their tasks.

Depending on the buffering mechanism of the d-MMU, an efficient NI as shown in Figure 4.8 is proposed for the memory-centric OCIN. The NI uses a buffering control to generate a borrowing request to the d-MMU for borrowing memory resources. And thus, the d-MMU checks the valid table and generates the borrowing address for the NI.

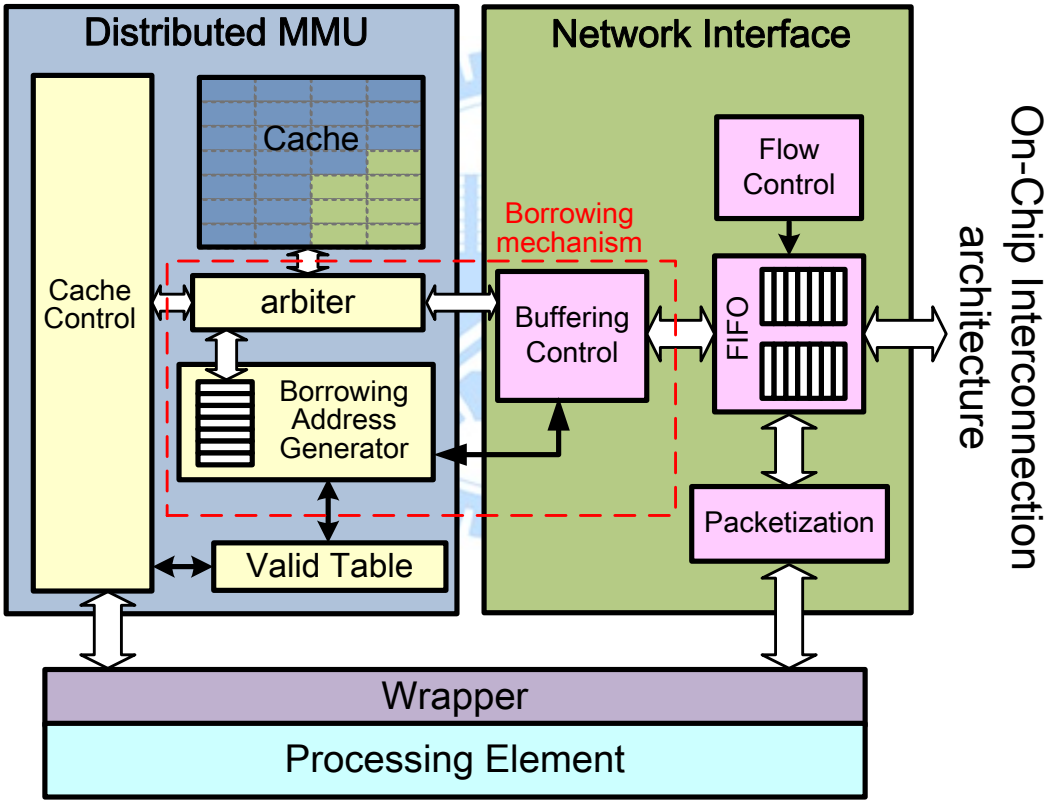


Figure 4.8 Efficient network interface with d-MMU.

Figure 4.9 presents the buffer borrowing interface between the NI and d-MMU. The operations of the buffer borrowing include write, read and release. For the write operation, the buffering control should send a buffer request to the d-MMU first, and send the blocking data until receiving a grant signal. However, the head-of-line blocking may release while waiting the grant from d-MMU or waiting the data from

PE. Therefore, a release operation can release the extension memory resources. While the blocking condition of output queue disappears, the buffering control would send data request to the d-MMU, and wait valid signal to read back the buffering packet from d-MMU.

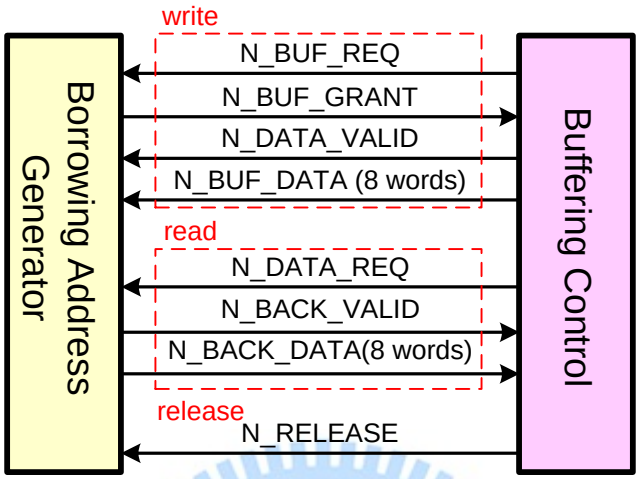


Figure 4.9 Buffer borrowing interface between the NI and d-MMU.

4.3.1 Borrowing Address Generator

When the NI requests an extend buffer to store the blocking packet, the borrowing address generator searches an empty space in the distributed memory via checking the valid table. This valid table is attached in the cache tables as shown in Figure 4.10. The distributed memories are divided into two banks with four-way association. The memories corresponding to the last associated table in bank 0 and bank 1 are infrequently used in opposition to others. Therefore, the d-MMU can borrow the empty spaces corresponding to this table. Moreover, each cache line in the four-way association contains 4 x 8 words. Therefore, the maximum payload of a packet can be stored in a memory block (8 words) in one cycle. If a memory block is borrowed, the d-MMU asserts the status bit that represents the borrowing data. Depending on the status bit, the cache control can mask the searching of this table in a searching operation.

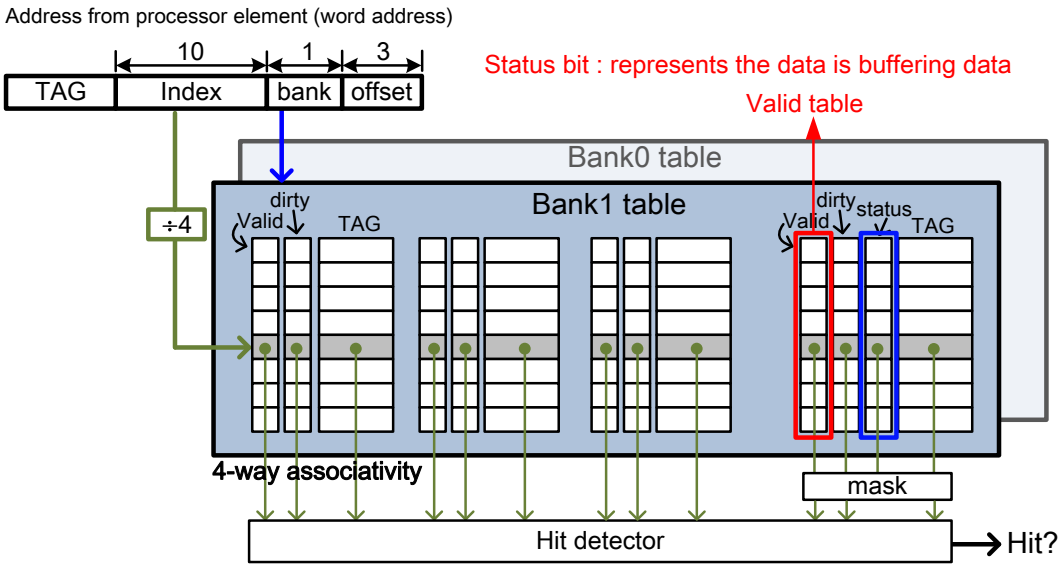


Figure 4.10 Borrowing mechanism in d-MMU

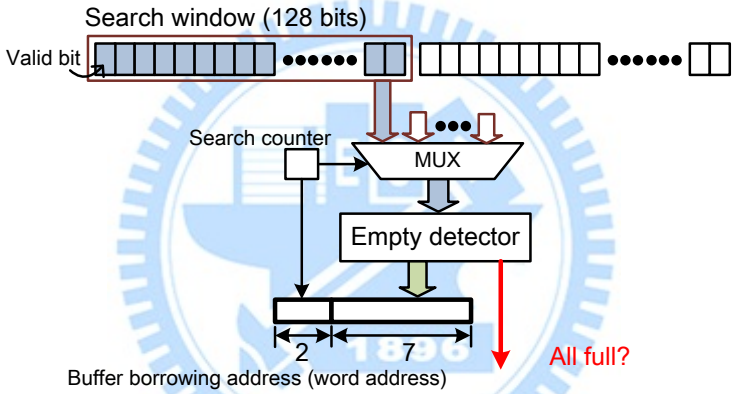


Figure 4.11 Architecture of the empty memory block searching.

After the NI send a borrowing request to the d-MMU, the NI should take 2-8 cycles for collecting the payload. Most packets contain 8 flits in their payloads, and the average size of payload is about 4 words. Therefore, the d-MMU has to search the empty memory block in 4 cycles. Additionally, the last associated tables in bank 0 and bank 1 contains 512 valid bits. To search the empty memory block, a 128-bit searching window is adopted. Figure 4.11 shows the architecture of the empty memory block searching. The searching window is controlled by a search counter. The empty detector detects an empty memory block and generates the borrow address with the search counter. If all memory blocks in a searching window are full, the searching windows will move to the next 128 bits. Figure 4.12 shows the searching flow chart of the borrowing mechanism. The flow can be divided into three steps, which are empty memory block searching, borrowing status setting, and data writing.

The operations of empty memory block searching and borrowing status setting are described above. While writing data in the borrowing memory block, the borrowing address should be stored in the address queue for reading operations. After writing the payload into the memory block, the grant signal is changed to 0 for the next borrowing request.

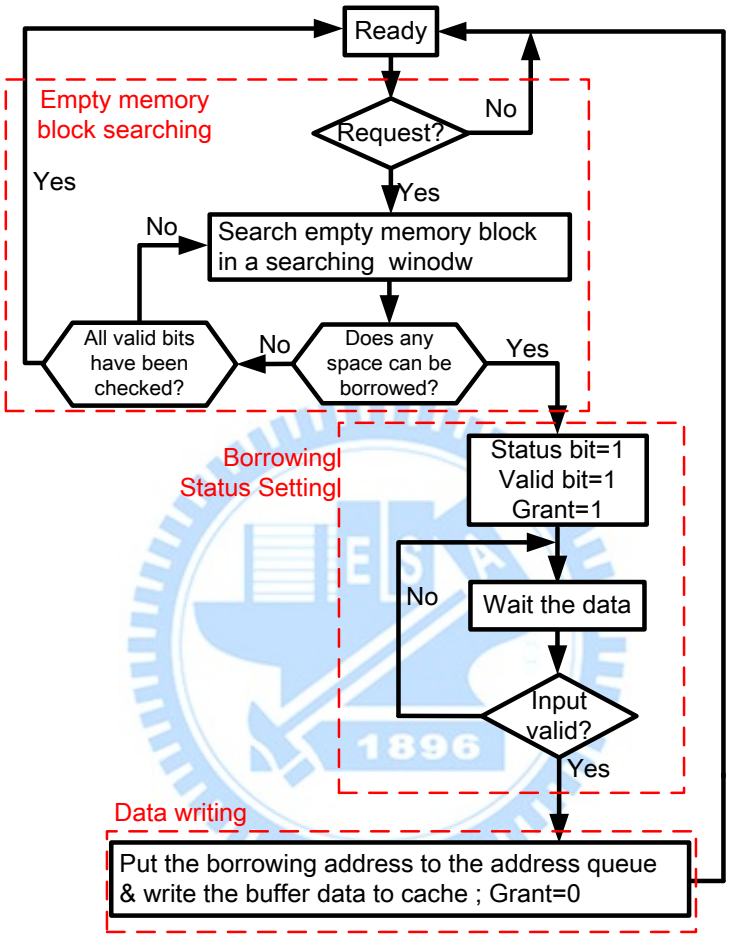


Figure 4.12 Searching flow chart of the borrowing mechanism in d-MMU.

4.3.2 Buffer Control

The buffering control in NI detects the empty size of the output queue and sends the borrowing request to d-MMU. For example, if the burst length of packet is less than empty size of output queue, a transfer is permitted. Figure 4.13 shows the block diagrams of borrowing mechanism in the buffering control. The buffering control sends the write, read, and release operations depending on an empty pointer of the output queue and a borrowing pointer of the borrowing header queue. The empty pointer and borrowing pointer indicate the number of the occupied buffers in the

output queue and borrowing header queue, respectively. If the borrowing header queue is not empty, it represents that there are still some blocked packets in the d-MMU.

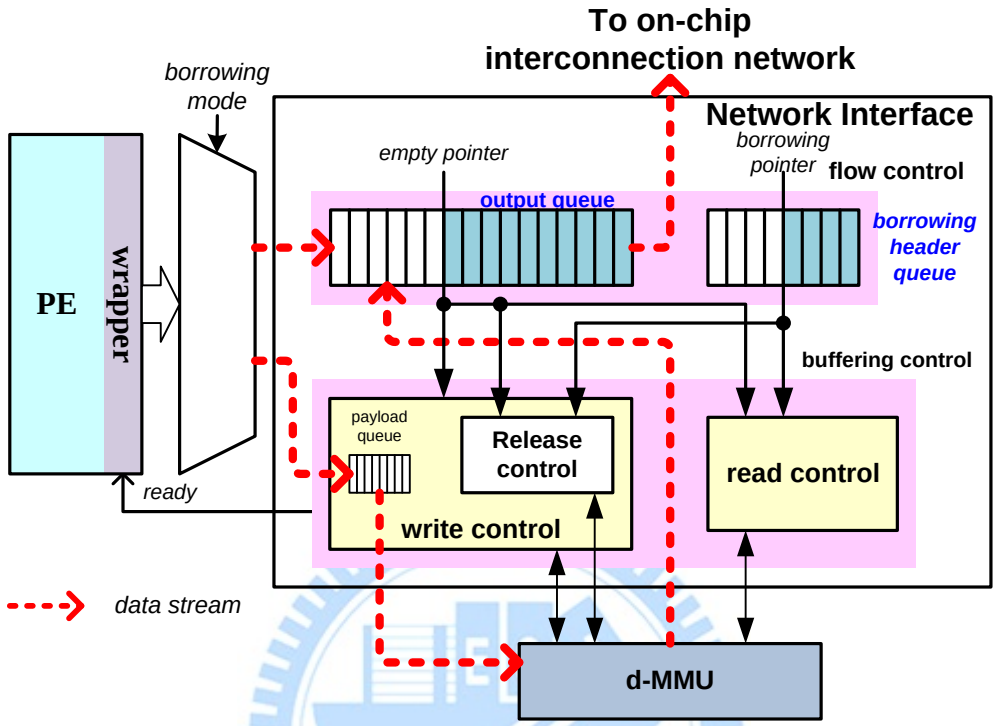


Figure 4.13 Block diagrams of borrowing mechanism in network interface

In addition, the write control contains a payload queue for collecting the payload, and then writing this payload to the borrowed memory block. The borrowing control policy of the buffering control is presented as shown in Figure 4.14. The borrowing mode indicates whether the channel is blocked and the blocking data should be stored in the d-MMU or not. Therefore, after receiving data from the PE, the data should be stored in the d-MMU in the borrowing mode. Otherwise, the data can be stored in the output queue when the size of the empty slots is larger than the current payload. While waiting the borrowing grant from d-MMU and collecting the payload, the head-of-line blocking may be released. Therefore, the borrowing mechanism can also be released if the borrowing mode equals to zero. The release signal will interrupt the search operation of d-MMU.

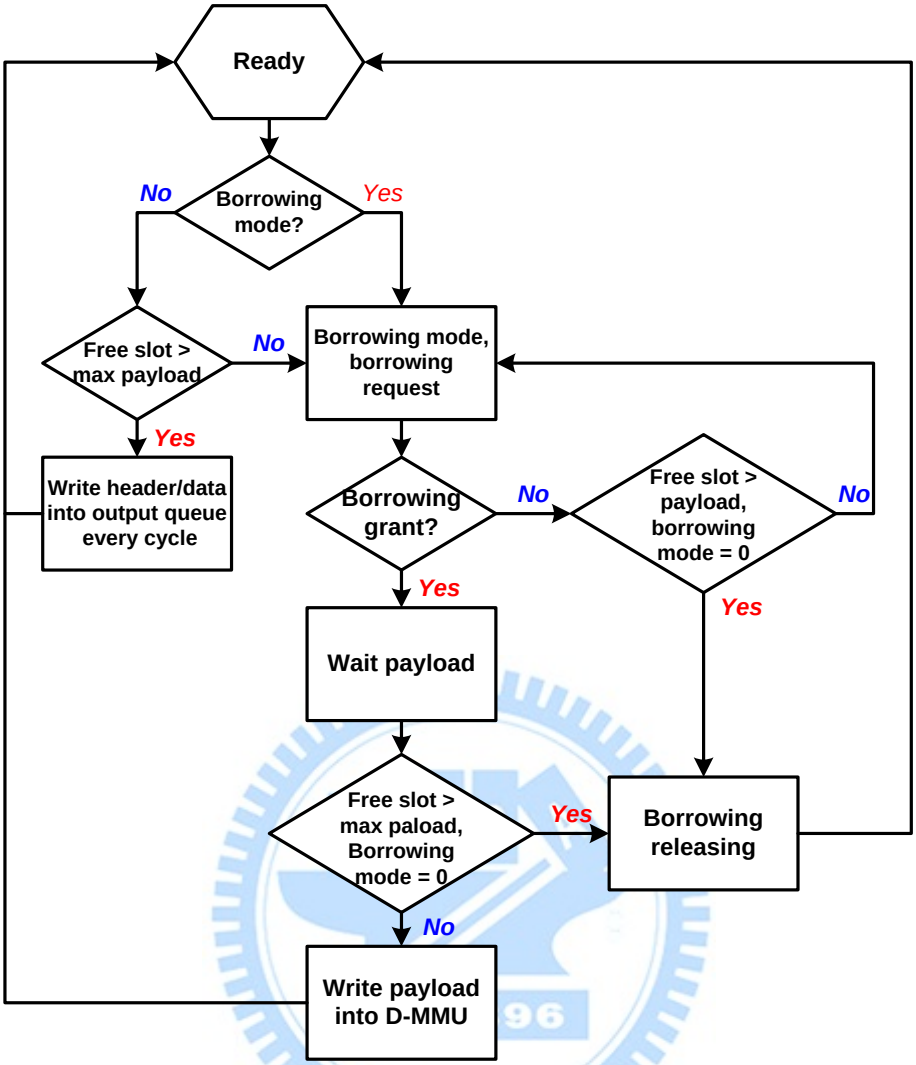


Figure 4.14 Borrowing control policy of the buffer control

For write operation, when receiving full payload from PE we would double check the empty size of output queue. If the head-of-line is released and there is no blocked data in the d-MMU, the received data can store into output queue and set borrowing mode to zero. This release operation is shown in the Figure 4.16. Otherwise, the received data should store in the d-MMU followed with the former blocked data. The timing diagram of write operation is shown in the Figure 4.15.

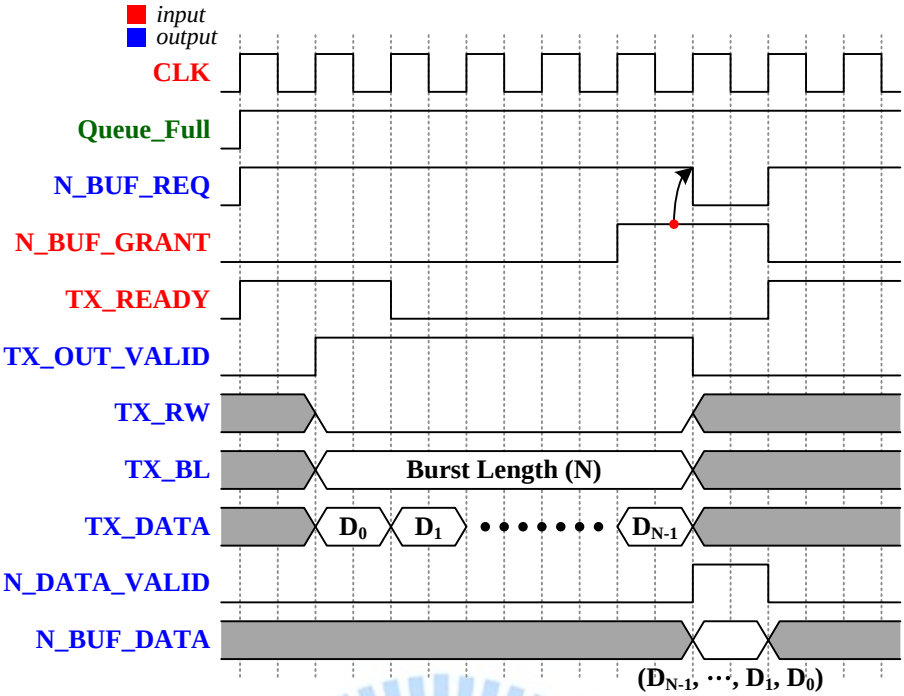


Figure 4.15 Timing diagram of writing blocked data in the d-MMU

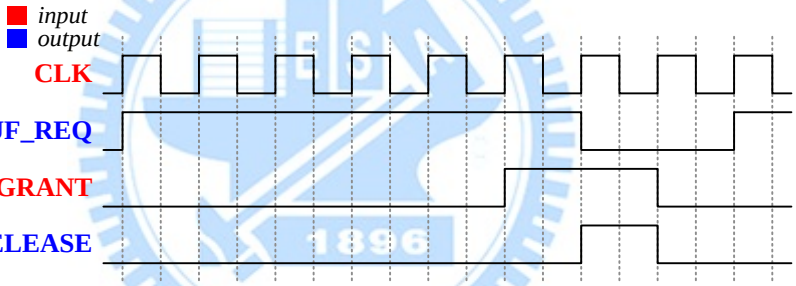


Figure 4.16 Timing diagram of releasing extension memory block

While the empty size of output queue is larger than the current packet which is stored in the d-MMU, buffer control would send request to d-MMU for reading back. We can access the header flit stored in the borrowing header queue to complete this work. Then, buffer control would receive blocked data and store into output queue while the data valid bit is activated. Timing diagram of read operation is shown in the Figure 4.17.

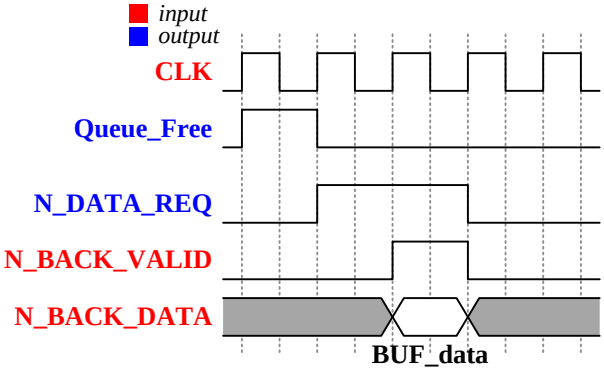


Figure 4.17 Timing diagram of read back blocked data.

4.4 Simulation Results

The proposed efficient NI and memory-centric OCIN are implemented in SystemC for the cycle-driven simulation. Thereby, the simulation environment is set as a 4x4 router with 4 PEs to evaluate the performance improvement via the efficient NIs which is shown in the Figure 4.18.

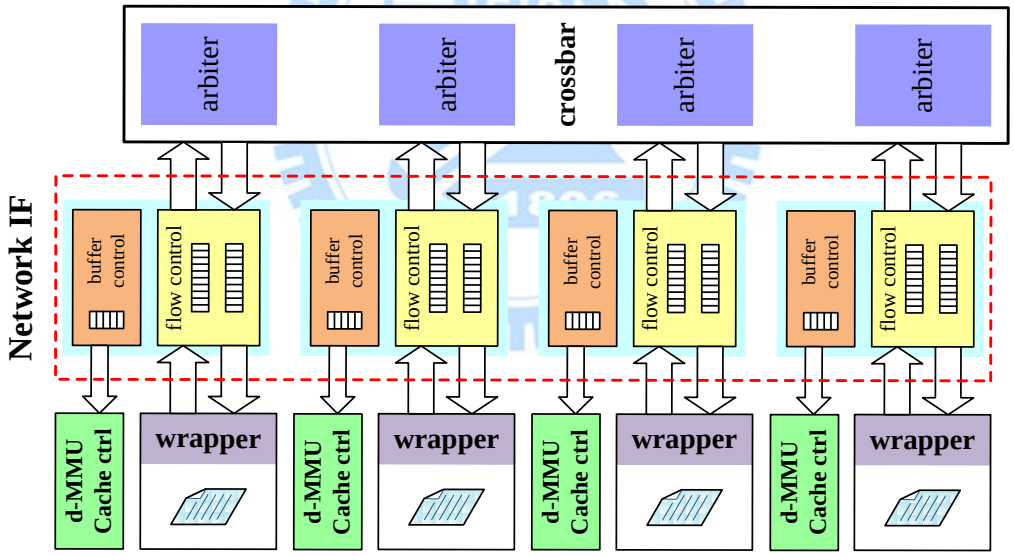


Figure 4.18 Simulation environment

The input pattern is generated from C++ code by random function. The access type of write/read equals 4. Write operation is the major loading of data transfer. Packet priority, burst length and destination address are all random.

Figure 4.21(a) shows the execution time of transferring 200000 random packets under various injection loads and output queue sizes of sender. With the increasing injection load, the execution time decreases because the transferred packets are fixed.

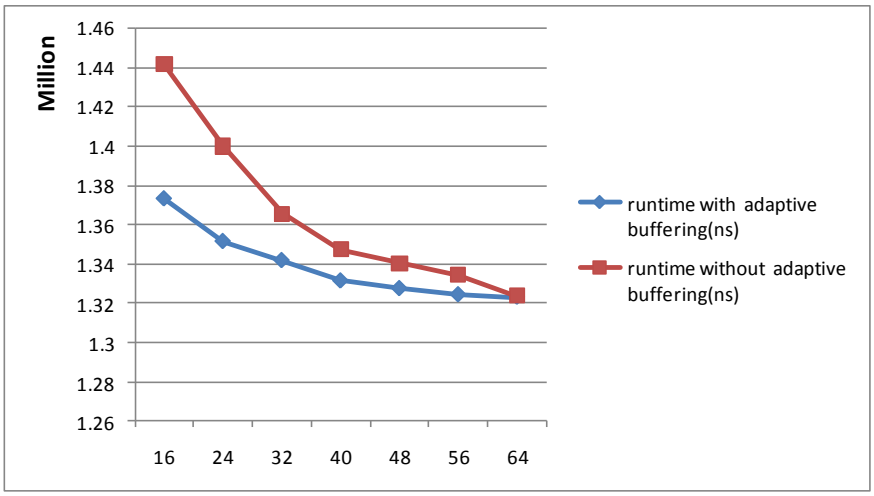


Figure 4.19 Execution under different output queue size (injection load = 20%)

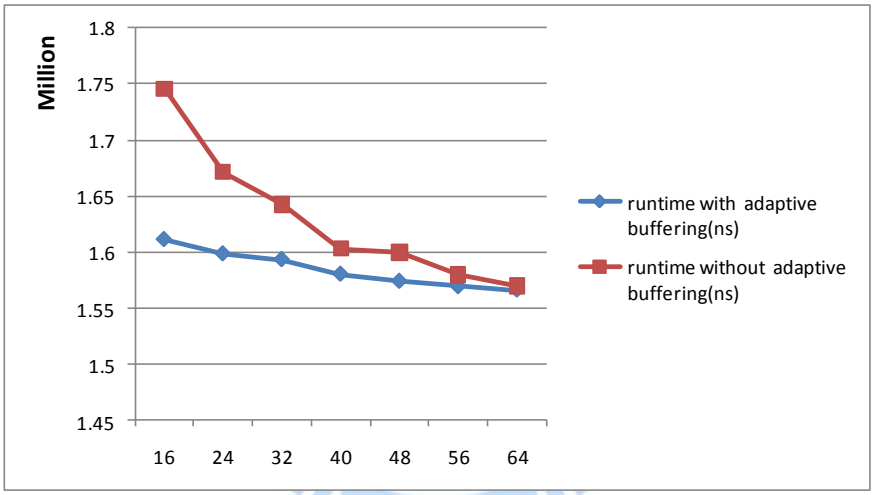


Figure 4.20 Execution under different output queue size (injection load = 15%)

The buffer size of sender and receiver in the NI can be dynamically adjusted according to the congestion of network interconnection [4.9][4.10]. By adjusting the size of buffer we can increase the data switching performance. In the same word, we can increase the maximum borrowing memory blocks of d-MMU to improve the performance of buffer borrowing mechanism and promote efficiency of data transfer in the same time.

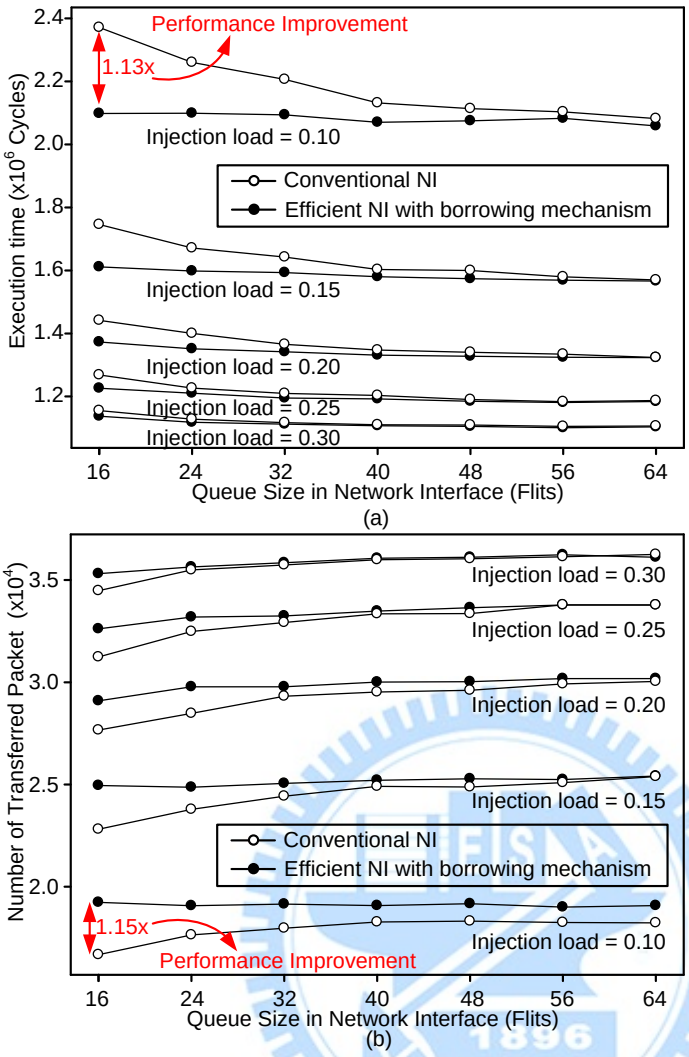


Figure 4.21 (a) Execution time under various injection loads and queue sizes (b) Transferred packet under various injection loads and queue sizes

Additionally, Figure 4.21(b) shows the number of transferred packets in 300000 cycles under various injection loads and queue sizes. Based on the simulation results, the proposed borrowing mechanism can achieve the similar performance with different queue sizes. Moreover, the proposed efficient NI can realize about 1.15x performance improvement compared to the conventional one with 16flits.

Under the same setting, Table 4.3 shows the number blocking condition under various injection rate and queue sizes. We can see this network interface improves data blocking condition 2%~5%.

injection rate = 35%	blocking cycle of PE within 1ms			
Data FIFO size (words)	**16**	**24**	**32**	
with adaptive buffering	38436	38623	38701	**blocking rate = 55%**
without adaptive buffering	39604	39619	39653	
reduction rate	3.0388%	2.5788%	2.4599%	
with adaptive buffering	38184	38687	38007	**blocking rate = 60%**
without adaptive buffering	39677	39715	39734	
reduction rate	3.9100%	2.6572%	4.5439%	
with adaptive buffering	38056	38530	38320	**blocking rate = 65%**
without adaptive buffering	39568	39833	39409	
reduction rate	3.9731%	3.3818%	2.8419%	
with adaptive buffering	38556	38654	38420	**blocking rate = 70%**
without adaptive buffering	40321	39864	40519	
reduction rate	4.5778%	3.1303%	5.4633%	
with adaptive buffering	40093	40019	40603	**blocking rate = 75%**
without adaptive buffering	41796	41252	41457	
reduction rate	4.2476%	3.0810%	2.1033%	
with adaptive buffering	44541	44212	44198	**blocking rate = 80%**
without adaptive buffering	45801	46492	45227	
reduction rate	2.8289%	5.1570%	2.3282%	
with adaptive buffering	52348	52729	52100	**blocking rate = 85%**
without adaptive buffering	54004	53017	53394	
reduction rate	3.1634%	0.5462%	2.4837%	

Table 4.3 Blocking condition reduction rate under various output queue size and blocking rate of receiver (size of borrowed memory blocks = 512 words)

4.5 Summary

A memory-centric on-chip interconnection network provides effective memory and communication bandwidths for on-chip SRAM-rich SoCs based on MMUs. Moreover, network interface is a primary element in the performance of on-chip interconnection networks. In this chapter, an efficient network interface is presented to reduce the data blocking by a borrowing mechanism. The d-MMU can dynamically allocate the memory resource for buffering the blocking network data. By considering the borrowed memory blocks and d-MMU, the size of the output queue in network interface can be dynamically scheduled. According to the cycle-driven simulation

results in SystemC, the proposed efficient network interface can achieve performance improvement by 1.15x compared to the conventional one. At the same time, this efficient network interface can reduce data blocking condition compared to conventional one. Therefore, the proposed efficient network interface can increase the performance of the memory-centric on-chip interconnection network.

Chapter 5
Memory-Centric On-Chip Data Communication for Wireless Video Entertainment Systems

In this chapter, a memory-centric on-chip data communication platform is developed for a wireless video entertainment system. First of all, the introduction and motivation of the wireless video entertainment system will be depicted in the section 5.1. Subsequently, section 5.2 will describe the concept of the memory-centric on-chip data communication platform. And then the development of the wireless video entertainment system will be introduced in the section 5.3. Finally, wireless video entertainment system will be constructed in memory-centric on-chip data communication platform, and it will be described in section 5.4.

5.1 Motivations

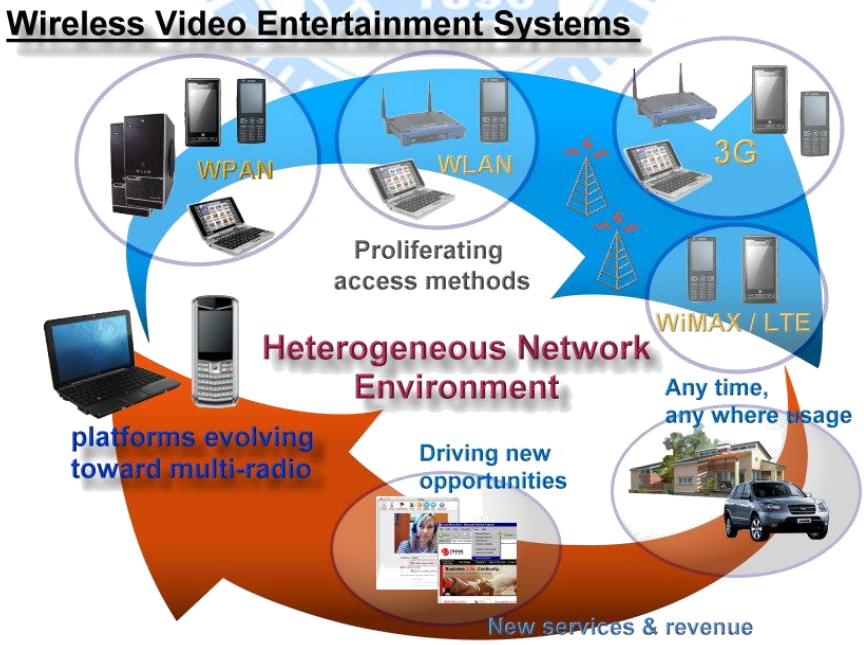


Figure 5.1 Wireless Video Entertainment Systems

With the advancements of the wireless communication and multimedia

techniques, various digital communication products are developed in our life. These modern electronic products provide more convenient communication environment and media enjoyment for humans than those before. However, with different applications or standards, a variety of devices would be needed. Figure 5.1 illustrates a heterogeneous network environment in our life. In recent years, merging different networks, electronic appliances and media devices into a heterogeneous integrated platform becomes an important issue that enables people enjoy their life in an more friendly and energy-efficient digital environment.

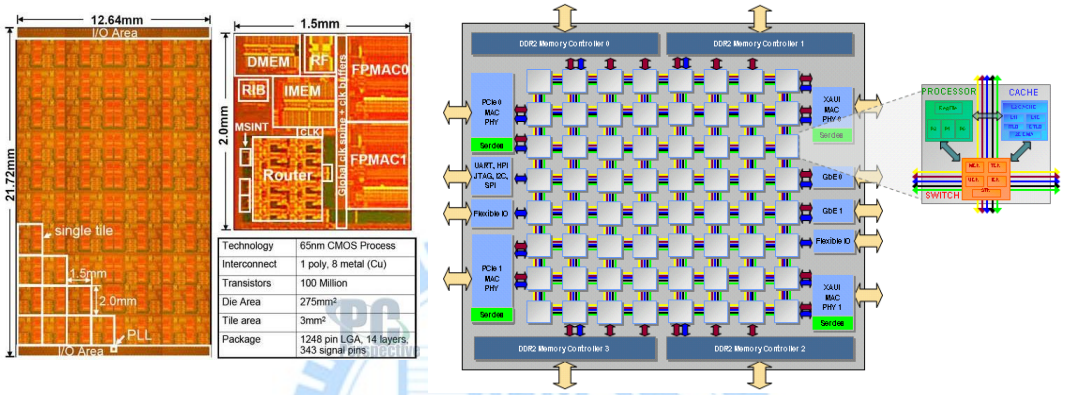


Figure 5.2 Homogeneous multi-core platform (a) Intel Polaris (b) Tilera TILEPro64^{TM} Processor

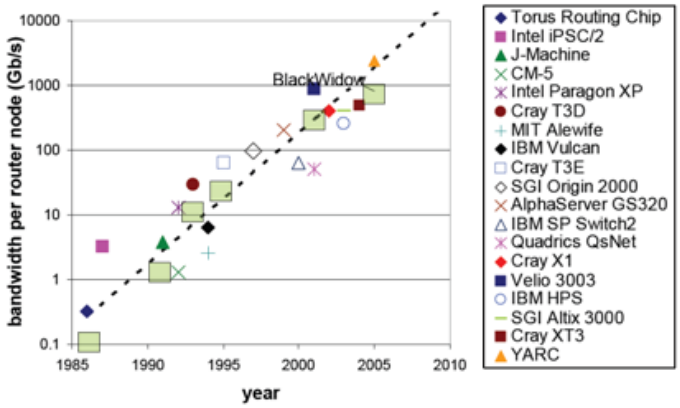


Figure 5.3 Trend of the data transmitting bandwidth

To integrate various applications into a system, a multi-task/multi-core concept provide a typical solution to build the system. The design of multi-core platform is a popular research area recently. Figure 5.2 shows two homogeneous multi-core platforms. Intel proposed an 80-core platform as shown in Figure 5.2(a) and Tilera [5.1] proposed a 64-core platform as shown in Figure 5.2(b). These multi-core

platforms can execute billions of operation per second. Furthermore, the data transmitting bandwidth for the multi-core platform is increasing year by year as shown in the Figure 5.3. However, the overall system performance could be limited by the task partitioning, task mapping, memory resource allocation, and memory data accessing. Figure 5.4 indicates the bottlenecks of multi-core platforms with insufficient memory bandwidth and memory capacity for supporting high communication efficiency in the multi-core systems. With ongoing development of multi-core or multi-task system, both the memory capacity and memory access bandwidth are required. Enabling multiple memory data access is necessary for improving the memory bandwidth. However, increasing the memory read/write ports not only increases the hardware complexity but also reduces the memory performance and noise immunity. Conventional memory access method cannot provide enough memory bandwidth for multi-core platform. Hence, the memory management in multi-core or multi-task platform will become more and more important. It is an essential issue that reducing additional memory access and increasing the memory bandwidth effectively. For these reasons, a memory-centric on-chip data communication platform will be proposed and introduced in the following section.

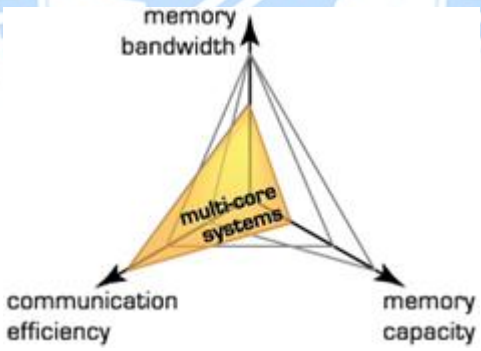


Figure 5.4 Comparison between memory bandwidth, memory capacity and communication efficiency in multi-core systems

5.2 Memory-Centric On-Chip Data Communication Platform

5.2.1 Overall Architecture

To solve the problems as mentioned above, a hierarchy memory-centric on-chip

data communication platform is proposed and the architecture is shown in Figure 5.5. Heterogeneous processing elements such as microprocessors and application-specific stream processors can be integrated in the platform. In this platform, each processor element owns distributed memory management unit (d-MMU). The d-MMU includes local cache (D-cache and I-cache) and cache controller which can efficiently handle all memory requests generated by the processor elements. It can dynamically allocate unused space in cache for buffering the transmitting data. If processor elements need additional memory resource requirements, the centralized memory resources including centralized cache and off-chip DRAM can be used. It is controlled by a centralized memory management unit (c-MMU). It can dynamically allocate and manage the memory resources according to different memory requirements.

For the data communication between processor elements, message-passing technique is applied for this platform. The processor elements transmit/receive the data to/from others through an on-chip interconnection network. Network interface is applied to packetize the transmitted data to interconnection and de-packetizes the received data from interconnection. Furthermore, in order to have better energy utilization for green computing, the power management unit can be applied to dynamically control the supply voltage and operating frequency of each processor element for saving energy consumptions.

In the heterogeneous multi-task platform, different processor elements would have quite different memory requirements with different specific functions in a platform. For instance, the memory requirement of the video decoding is larger than that of the wireless processing unit. Moreover, different system environment factors may affect memory utilizations for the applications in platform during runtime. Different qualities of wireless channels may have different memory behavior in a wireless video integrated system. Thus, a multilevel memory hierarchy on-demand memory system is applied for this platform. The memory system enables the processing elements to own different memory resources dynamically. In the following section, the concept of on-demand memory system will be introduced.

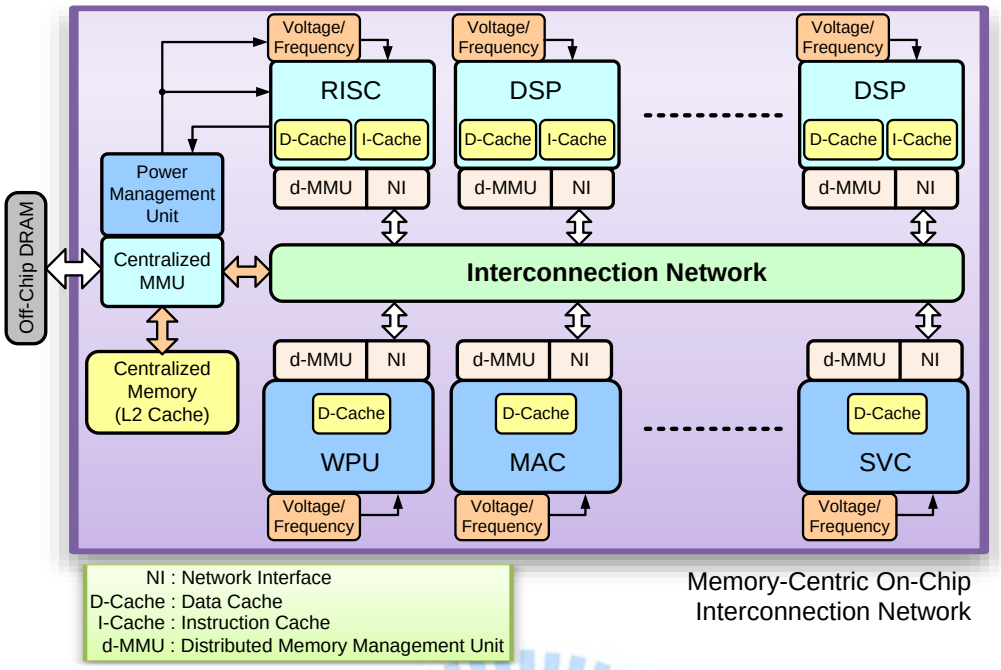


Figure 5.5 The architecture of memory-centric on-chip data communication platform

5.2.2 Concepts of On-Demand Memory System

In on-demand memory system, a three-level memory hierarchy is constructed, and the illustration is shown in Figure 5.6. For the first hierarchy level, distributed memory management unit (d-MMU) is applied to control the memory accesses. It includes distributed cache and cache controller for processor elements. Furthermore, in order to improve the transmitting efficiency for data communication, d-MMU can dynamically allocate unused space in distributed cache to store packet data so that the stall caused by data blocking can be prevented.

For the second level hierarchy of the on-demand memory system, centralized memory management unit (c-MMU) is constructed to provide more memory resources for processor elements. In c-MMU, a cache controller and centralized cache is included. In addition, the configuration of centralized cache can be dynamically adjusted according to the different memory requirement from processor elements. For example, if a processor element need larger memory requirement than others, it can own more centralized memory resources than other processor elements. Adaptive cache control in c-MMU controls the adaptive allocation and cache operation. In addition, unused memories can be power down to save memory power consumptions

for green computing.

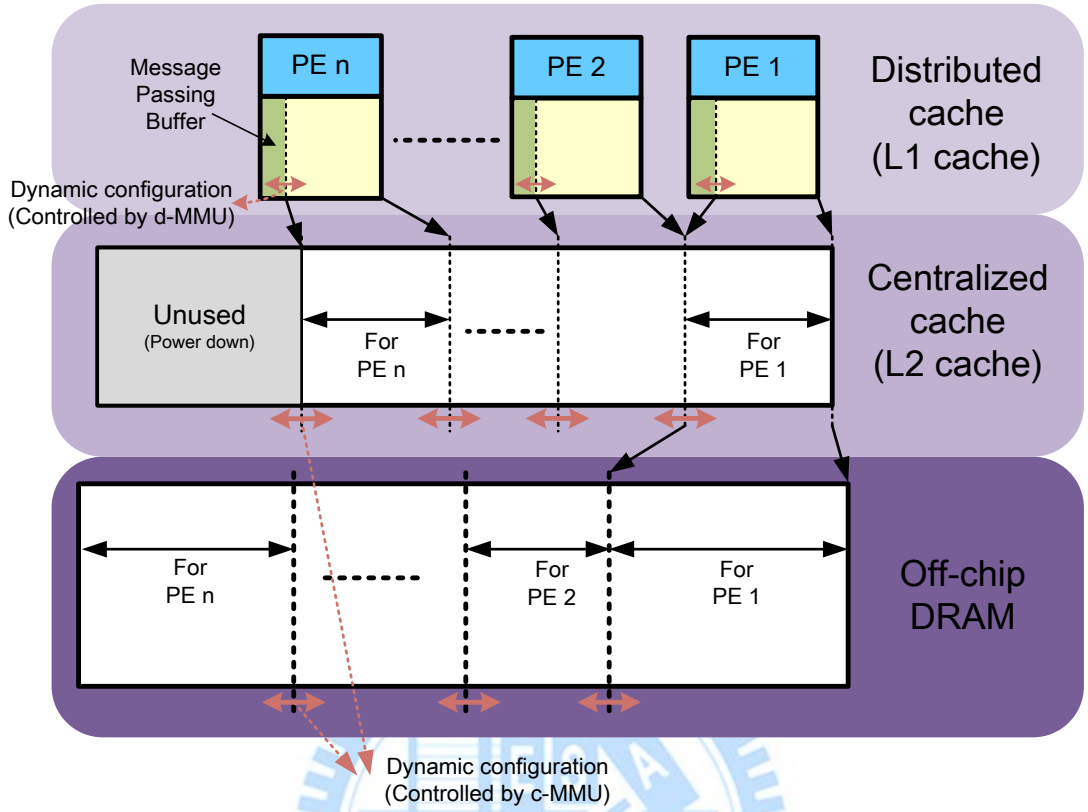


Figure 5.6 Illustration of the memory hierarchy in on-demand memory system

For supporting enough memory space, off-chip DRAM is applied, and it is the third memory hierarchy level in the system. DRAM controller is needed to access the off-chip DRAM devices. It includes an external memory interface and address translator to improve the memory access efficiency.

In the on-demand memory system, all processor elements own a private address space and can dynamically be allocated. For data switching between processor elements, message-passing mechanism is used. On-chip interconnection network in the platform is designed for data communication. Note that the thesis is focus on on-demand memory system. The design of interconnection network is not included in this thesis. In conclusion, adaptive memory resource allocation can be achieved and the memory utilization can be improved by the memory management units.

5.3 Wireless Video Entertainment System

With the ongoing advancement in digital and communication techniques, digital

home service becomes a trend nowadays. In the daily life, home is the personal headquarters for living, keeping personal assets and information. If the digital home services are applied, the residents will effectively participate in any events happening in the local, national and global communities without unnecessary travel. Digital home technique integrates wireless, wired physical transmission and multimedia real-time processes. With wireless communication technique, we can use mobile electronic product, such as cell phone, PDA or notebook, to transmit or receive the message by a certain sever. You can monitor and control the situation which something or somebody happens at home remotely or receive immediate video what you want. But nowadays, many kinds of communication protocol have been used such as WLAN, bluetooth, WiMAX or LTE techniques. In order to support a variety of protocols, a heterogeneous network system would be constructed. It provides an adaptable processing unit to process various communications.

Many researches try to integrate the communication device and entertainment platform into a system. However, the current technologies and systems cannot effectively meet the requirements of these digital homes for some reasons [5.2]. First, there are too many incompatible and not interoperable systems and standards, and each system only work for one particular application, using a particular physical transmission medium, and incompatible hardware and firmware. The Second one is the throughput of the future digital home system may require up to 10Gps (gigabit-per-second), but the current home networking technologies is below 1Gps. So the system bandwidth must be improved. Furthermore, the scalability, security and power are also the problems.

To solve these problems, we integrate the wireless sever and multimedia processing unit together, construct an integrated heterogeneous-processing platform to serve a variety of services. In order to serve various transmit channels, the multi-task wireless video entertainment system is shown in Figure 5.7. Analog front-end system receives and digitizes the wireless signals. Then the data will be processed by an integrated, high performance digital system.

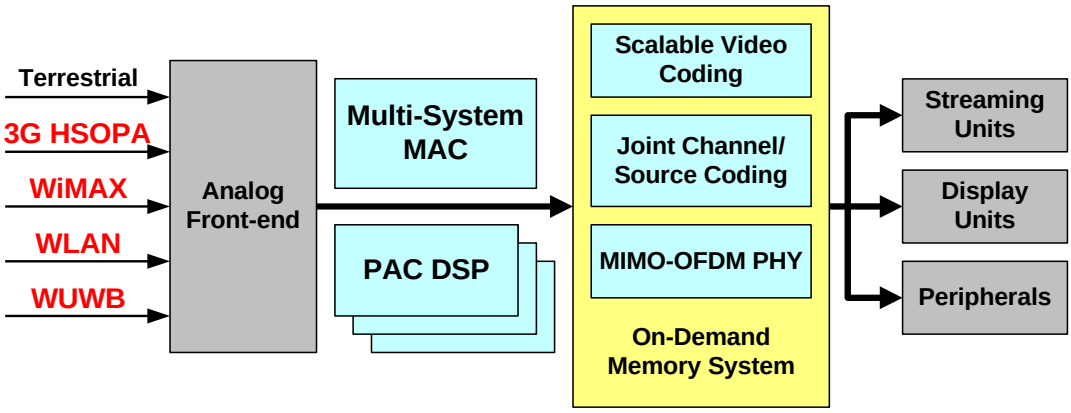


Figure 5.7 Multi-Task wireless video entertainment systems

In order to support various communication standards and have video entertainment for digital home service, a wireless video entertainment system is developed. It includes four functional blocks. The block diagram of wireless video entertainment system transceiver is shown in Figure 5.8. In this system, Scalable Video Coding (SVC), the extension of the H.264/AVC standard technique, is applied to provide spatial, temporal and quality scalability of the video sequences [5.3]. For the channel coding, Luby Transform (LT) coding, one kind of error correcting method, is applied to have high channel reliability. Media Access Control (MAC) module is the interface between application layer and the physical layer, and Wireless Processing Unit (WPU) handles the wireless signal processing including multi-standard baseband control and MIMO-OFDM. These functional blocks are grouped into a SoC system. At current development stage, receiver system is developed in an integrated on-demand memory system as which the red block in the Figure 5.8(b) represents. The system specification is listed in Table 5.1. Additionally, the details of WPU, MAC, LT coding and SVC coding will be described in the following sections.

	WPU (4x4)	MAC	LT Coding	SVC
Input data rate	160MBps (4Gbx12/s)	7.8MBps	7.8MBps	1333KBps
Output throughput	7.8MBps (with a 64-QAM modulation)	7.8MBps	7.8MBps	17.4MBps
Memory access bandwidth	222.4MBps	124.8MBps	124.8MBps	78.69MBps
Memory Size (Required)	6.25KB	2MB	1MB	11.34MB (a GOP)

Table 5.1 System Specification

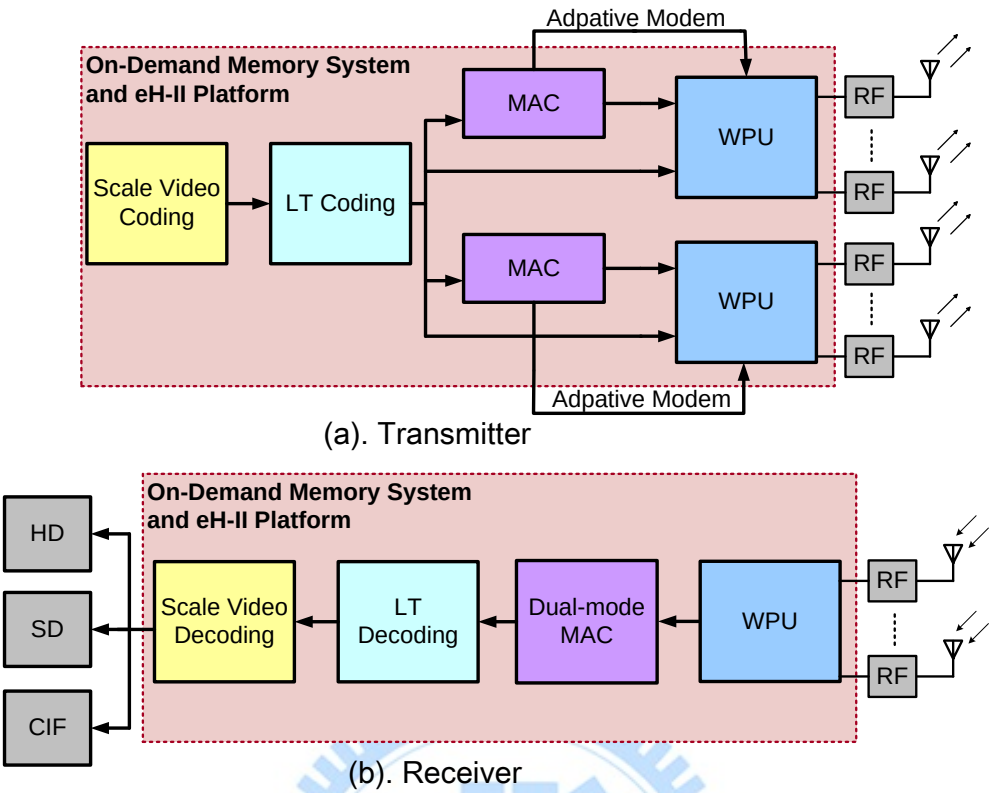


Figure 5.8 Transmitter and receiver block diagram

5.3.1 Wireless Processing Unit (WPU)

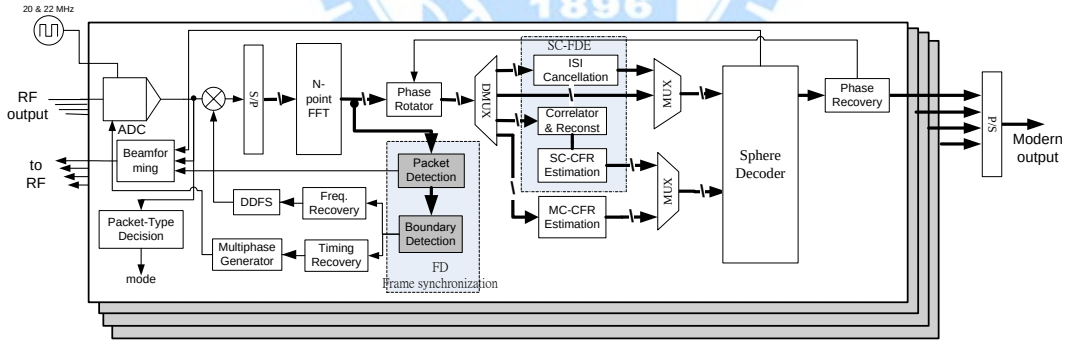


Figure 5.9 Single-FFT Architecture for MIMO Modem

The WPU is a designed as a frequency domain (FD) modem with the single-FFT architecture. Additionally, the single-FFT architecture for multi-standard baseband is suitable for IEEE802.11a/b/g/n/VHT and IEEE 802.15.3a/c. The architecture is shown as in Figure 5.9. There are three key components in this architecture, including frequency-domain (FD) synchronization, FD adaptive sampling and single carrier frequency domain equalizer (SC-FDE). The features of the three components are as

follows.

- Frequency-domain (FD) synchronization
  1. FD Adaptive Sampling
  2. FD Boundary Decision
  3. FD Anti-I/Q Phase Recovery
- Single carrier frequency domain equalizer (SC-FDE)
  1. Frequency-domain channel estimation (FD-CE)
  2. Frequency-domain ISI cancellation for DSSS non-CP SCBT
  3. Frequency-domain data decision
- FD adaptive sampling
  1. 6-symbol Lock
  2. 32 multiphase clocking
  3. Boundaryless
  4. Tolerance of -30,000~40,000 ppm SCO as shown in Figure 5.10.

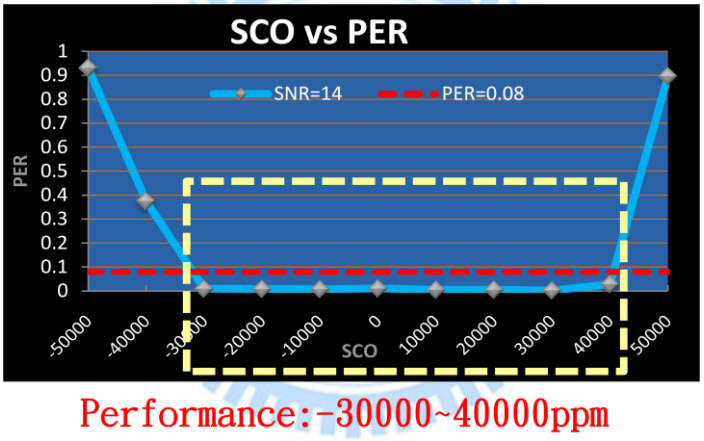


Figure 5.10 Single-FFT Architecture for MIMO Modem

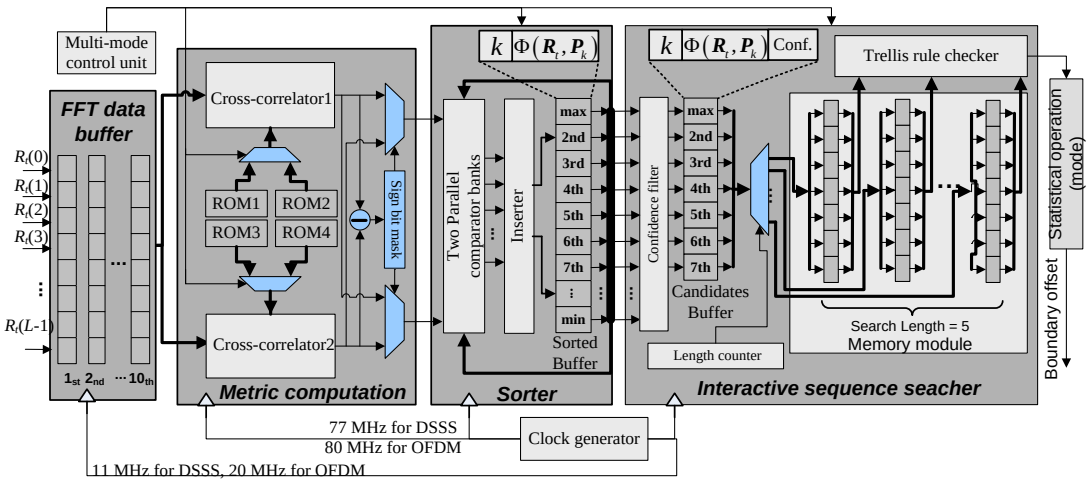


Figure 5.11 Single-FFT Architecture for MIMO Modem

Moreover, For the FD boundary decision, it contains the following features, only 1% detection error with low SNR (<5 dB) and gigh CFO tolerance. It is a trellis-based detector, and can be used both for DSSS and OFDM different systems. Figure 5.11 displays the architecture, and it contains 3 key components, including a metric computation, a sorter and an iterative sequence searcher. Additionally, for FD anti-I/Q phase recovery, it contain following features.

1. Pseudo CFO injection
2. Compatible with conventional method (Moose)
3. Robust in IQ mismatch
4. Gain error: 2dB
5. Phase error:20

5.3.2 Medium Access Control (MAC)

Medium Access Control (MAC) protocols play a very important role in wireless node-to-node communication, such as that between base stations and mobile terminals. This work concentrates on quick prototyping, early-stage verification and extensible design of multi-mode MAC layer systems. Starting from the integrated system of WiMAX/Wi-Fi dual-mode MAC, we apply Object-Oriented Analysis and Design (OOA&D) principle on both protocols, identifying the common and different components between both systems. By using divide-and-conquer and bottom-up design approaches, we are able to integrate WiMAX and WiFi MAC, and facilitate reuse and performance optimization of common components between the two systems.

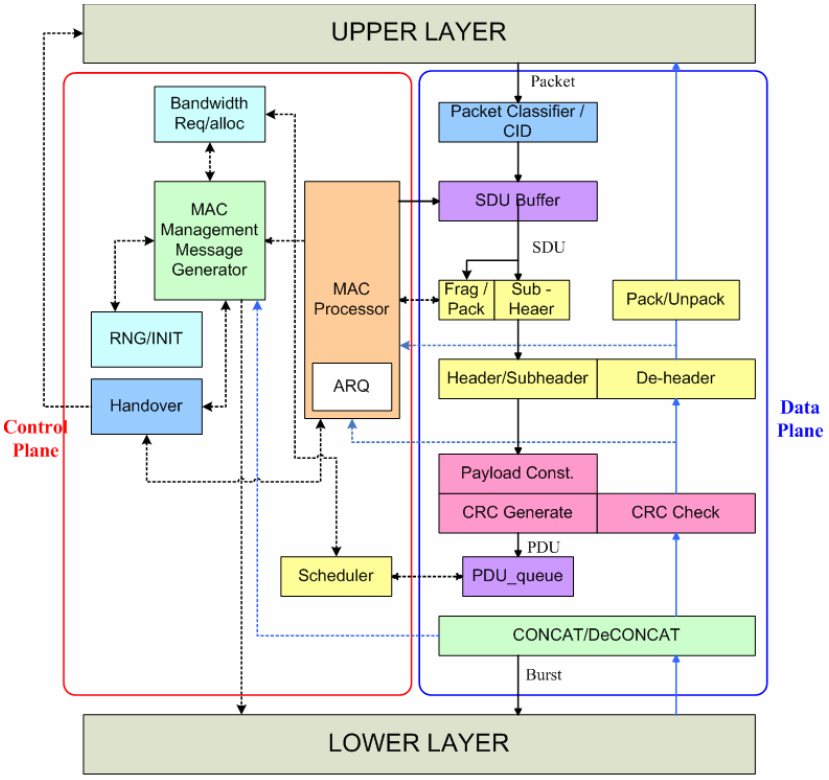


Figure 5.12 MAC Layer Architecture

As shown in Figure 5.12, the MAC protocol layer, in terms of implementation, could be separated in two parts: the Data Plane and the Control Plane. The main function of the Data Plane is production of MAC layer's protocol data units (PDUs). It could either be analyzed with electronic system level (ESL) methodologies, or realized by FPGA hardware solutions. The Control Plane takes control of the Data Plane according to various signal feedbacks. These feedbacks include PHY-to-MAC, Network-to-MAC and inter-BS or BS-to-MS signaling.

Besides data processing performance that directly relates to software/hardware co-design, there are other factors that have great impact or overall system performance. For example, the Request/Grant mechanism – the content of MS request shall be properly received and recognized by BS, and then properly responded, vice versa. Some MAC transmission mechanisms including auto retransmission request (ARQ), handover, uplink scheduling, external environmental mechanisms such as BS-end or MS-end channel condition, could deeply influence system performance. Unfortunately, it is difficult to analyze and verify the interaction of MAC functional interactions. The inter-node concepts cover a range even broader than system-level

design flows, and traditionally the verification of Control Plane begins at a later stage of design flow.

5.3.3 LT Coding

LT code is a class of rateless codes. Its performance is approximately close to channel capacities of arbitrary erasure channels. In theory, LT encoder generates infinite codewords. Each receiver starts decoding when sufficient codewords are collected. In spite of which codeword set is collected, the high recovery probability of source symbols is guaranteed. Consequently LT codes are channel independent and require no retransmission. For block codes, when there are too many codewords erased within a block, codewords in this block are undecodable and retransmission is needed. However, retransmission can jam the transmission and paralyze multicasting servers in multicasting. In comparison with block codes, LT codes are more suitable for multicasting. Recently, pre-codes concatenated with LT codes are standardized in 3GPP MBMS.

LT codes conduct BP algorithm as decoding scheme. The advantage of BP decoding is its low decoding complexity. It trades decoding ability for decoding complexity. The performance of LT codes are determined by two factors. One is the degree distributions derived based on BP algorithm. The other is the number of source symbols K. Theoretically, K approaches infinity and an LT encoder generates infinite codewords. In practice, with the same degree distribution, the performance of LT codes degrades with the decrement of K. BP decoding process fails when source symbols are not decoded completely but there are not codewords with degree one left. The information contained in these codewords is unable to be exploited by BP algorithm. This follows that the recovery probability of source symbols is not optimal. Codewords transmitted but not efficiently decoded results in the waste of transmission bandwidth.

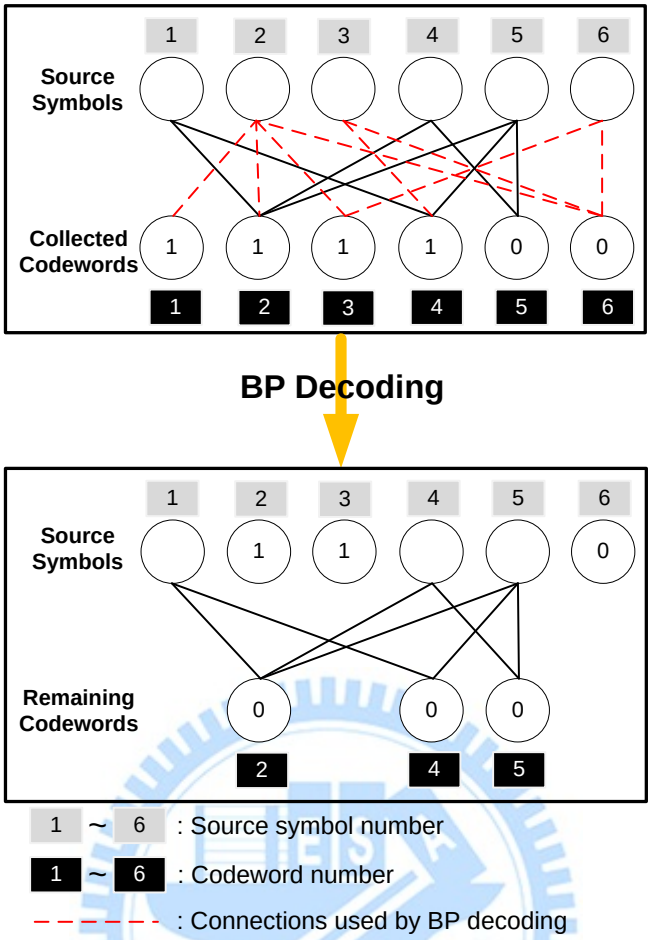


Figure 5.13 An example of decidable codewords which BP decoding fails to decode

Figure 5.13 is a simple example to show this condition. Now, there are six source symbols and six codewords. The red dash line stands for the connections that can be exploited by BP decoding. After BP decoding, codeword 2, 4, and 5 are left. Notice that, the source symbol 1 can be recovered by performing exclusive-or on codeword 2 and codeword 5. Similarly, source symbol 4 can be recovered by performing exclusive-or on codeword 2 and codeword 4. Finally, source symbol 5 is recovered by performing exclusive-or on codeword 2, codeword 4, and codeword 5. For rateless codes, decoding complexity is proportional to the total number of codeword degrees. After BP decoding, most of the codewords are removed. Besides, the average degree of remaining codewords is decreased. For example, with K=1000 and N=1120, the average degree of the received codewords is 43.6. After BP decoding, the average degree of remaining codewords is 8.3 and the corresponding degree distribution is shown in Figure 5.13. In addition, the average number of remaining codewords is 85.9. The total number of codeword degrees are (43.6×1120)/(8.3×85.9)=68.5 times

less after BP decoding. It is efficient to conduct more complicated decoding methods to recover the information in the remaining codewords.

5.3.4 Scalable Video Coding (SVC)

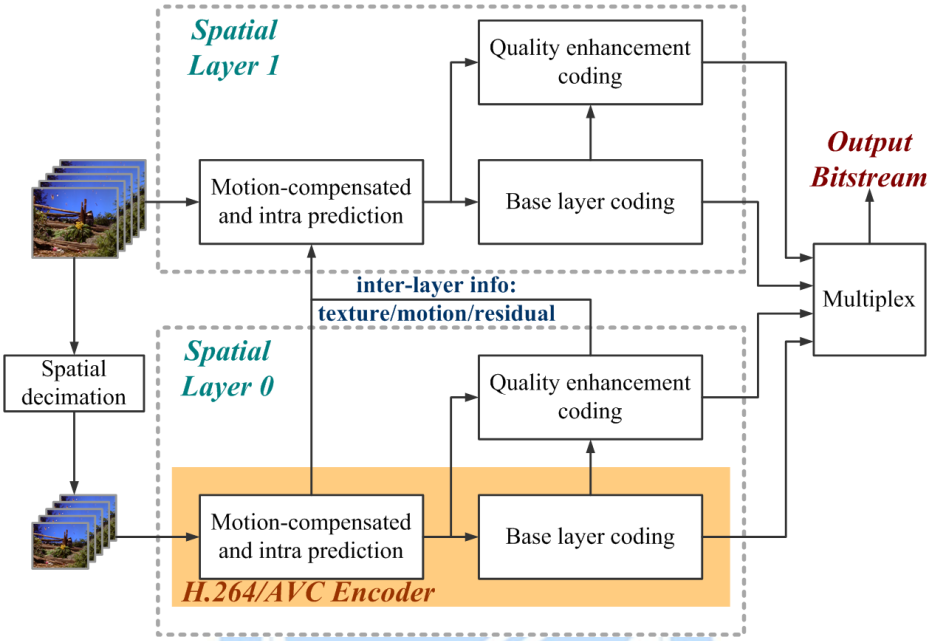


Figure 5.14 Architecture of an SVC encoder

Recently, with the prosperity of the Internet video, digital television, and portable devices, the demand of digital video becomes more and more diversified. To deal with those diversified video applications, Scalable Video Coding, the latest video coding standard inherited from the state-of-art H.264/AVC, is formed to provide different scalabilities (temporal, spatial, and quality) in a single bit-stream. Figure 5.14 shows an SVC encoder architecture with two spatial layers. To generate scalable bitstream, the input images are first downsampled to lower spatial resolution and encoded by H.264/AVC compatible video encoder. Afterward, the higher spatial resolution images are encoded by H.264/AVC encoder with additional advanced inter-layer prediction techniques to fully utilize the relationship between two consecutive spatial layers and consequently improve the coding performance. In addition, the quality and temporal scalabilities are achieved in each spatial layer by the approaches of Coarse Granular Scalability (CGS) and Hierarchical B structure, respectively. Finally, all generated bitstreams corresponding to different quality scalabilities are grouped into a single SVC bitstream. However, in addition to the primitive coding complexities of H.264,

the extra scalabilities of SVC also contribute significant computational complexity and memory requirement in hardware realization. Therefore, in order to minimize the computational complexity and memory requirement for realizing SVC codec, this project first analyzes the internal memory requirement and external memory access to find out the best coding method which can achieve best tradeoff between internal memory usages and external memory accesses and several efficient techniques are also proposed to improve the coding performance of SVC codec.

5.4 Memory-Centric On-Chip Data Communication Platform for Wireless Video Entertainment System

The designers try to meet efficient processing capability, merge multi-task system and use green computing concept in a system. However, when they try to integrate the heterogeneous functional blocks into a system, multiprocessing technique and multimedia process unit must be used. Furthermore, as the resolution of video processing applications becomes high, video signal processors should deal with a large amount of data within a tightly bounded time. Due to the huge data accesses, the system performance strongly depends on the memory bandwidth between processors and external memories. The system needs real-time and huge memory access requirement, but the speed gap of the memory and processor unit is large in the SoC system. Many researches are trying to minimize the speed gap. A well-organized memory management can significantly reduce the memory access latency. According to the data features of these applications, designer can find a well memory allocation method to reduce the number of memory access time and average access latency. Accordingly, for wireless video entertainment system, memory-centric on-chip data communication platform is applied to provide a high bandwidth and satisfy enough memory requirements.

According to the receiver system as mentioned in section 5.3, the processing sequence of these multiple tasks is generally step by step. Figure 5.15 shows the data stream of wireless video entertainment system. In memory-centric on-chip data communication platform, on-demand memory system can support heterogeneous and real-time memory requirement for wireless video entertainment system. MMUs in on-demand memory system enable the processor elements to have adaptive memory

resources. Base on different memory requirement of these processor elements, centralized MMU can dynamically allocate memory resources for processor elements. The architecture of the system is shown in Figure 5.16. The system components can be categorized into data computation part, data communication part and data storage part. For data computation, it includes WPU, MAC, LT coding and SVC processor elements. Subsequently, the other components will be introduced as follows.

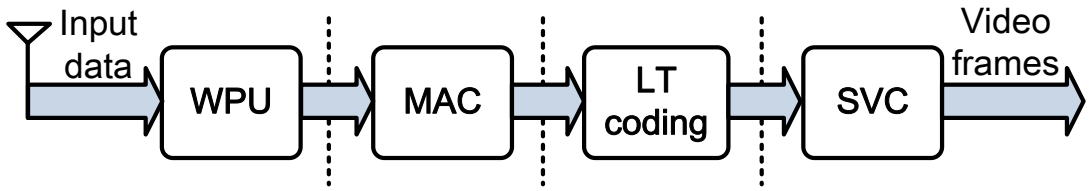


Figure 5.15 Data stream of wireless video entertainment system

For data communication, it includes network interface (NI) and interconnection network. In this system, message-passing mechanism is applied. With this mechanism, the transmitting data are packed into packets by network interface, and through the interconnection network using a pre-defined message-passing protocol. NI packetizes the transmitting data with a header indicating the data source, destination and some data information, and then transmits to the other node. It also de-packetizes the receiving data from the other processor elements. In addition, a packet queue is included in NI to store the blocking packet.

For data storage, each distributed processor element own a d-MMU, it includes a distributed cache (L1 cache) and cache controller for memory access. It also manages the cache usage. When packet queue size in NI is insufficient, d-MMU can borrow some unused cache block for NI. In addition, c-MMU is constructed for providing more memory resources. It includes centralized cache (L2 cache) and cache controller for processor elements. The cache controller can support dynamical cache re-organization for allocating different cache resources for different processor elements. In c-MMU, a DRAM controller is constructed to efficiently access off-chip DRAM. In DRAM controller, Address translator rearranges and translates address to have an efficient memory allocation, and the memory requests enter the memory interface with command scheduling to reduce memory access latency.

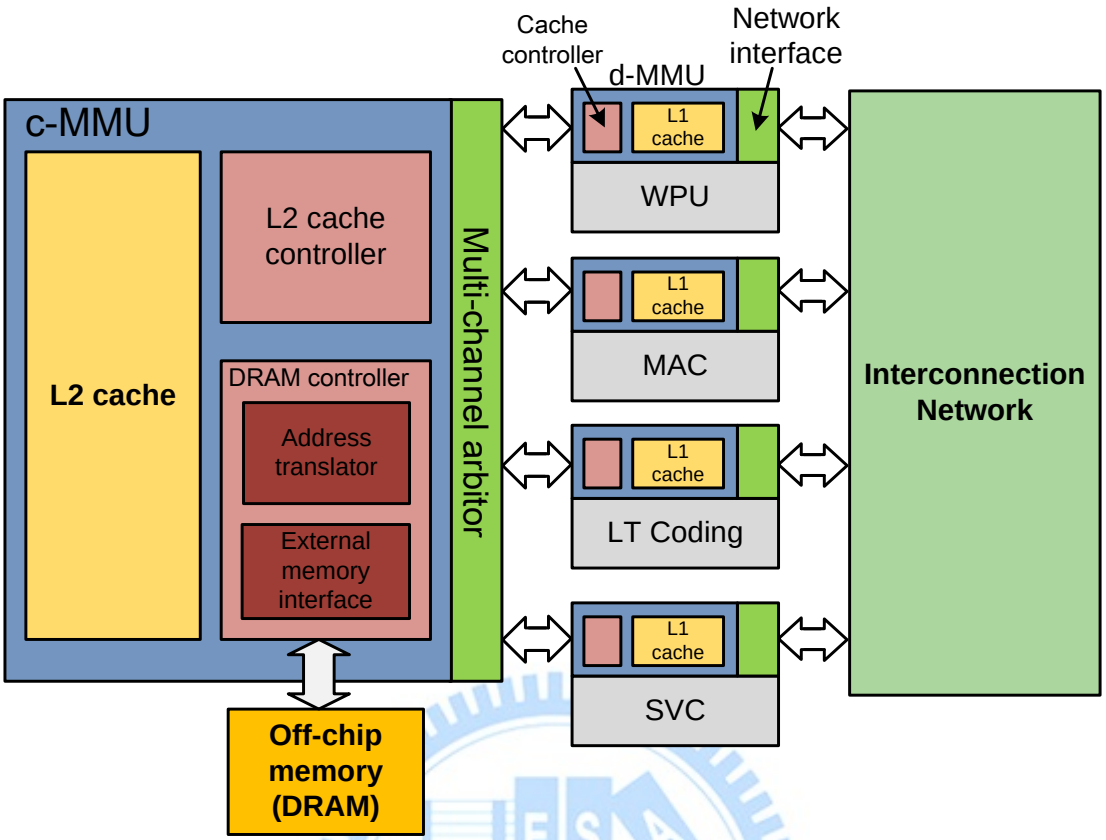


Figure 5.16 On-Demand Memory System architecture

5.5 Simulation Results

The proposed on-demand memory system is implemented in SystemC for the cycle-driven simulation. Thereby, the simulation environment is set as a hierarchy centralized MMU and a 4x4 router with 4 PEs (WPU, MAC, LT coding, SVC) and 4 distributed MMUs to evaluate the performance improvement via the efficient network interfaces which is showed in Fig. 5.16. We focus on the data transmit among PEs and data blocking condition of PEs. When the transmit data type is a large number and consecutive, this network interface can show more its value. The output data of this WPU and LT coding does not meet this type. Therefore, we do not discuss here.

For the MAC we detect the number of data blocking cycles in the output of wrapper under various output queue size of sender in NI. We dynamically adjust the size of output queue and input queue in the NIs and the size borrowed memory blocks for various blocking load in the receiver to see the data blocking condition in the output of processor elements.

Figure 5.17 shows the reduction rate under various blocking rate of receiver end. The size of output queue is 16 words in the sender and size of input queue in the receiver is 32 words. We can see if the size of memory borrowed blocks is enough; the reduction rate can increase linearly.

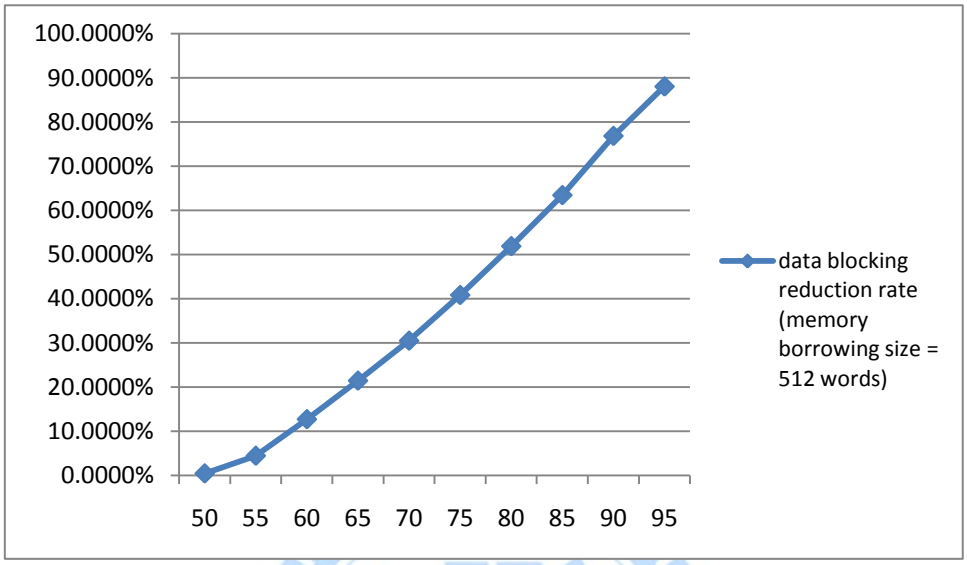


Figure 5.17 data blocking reduction rate under memory borrowing size = 512 words, size of output queue in the sender = 16 words, size of input queue in the receiver = 32 words

sender output FIFO size =16 words				
receiver FIFO size = 32 words	blocking cycle of MAC within 2000000cycle			
memory borrowing size (words)	**16**	**32**	**48**	
with adaptive buffering	11339	11339	11339	**blocking rate = 50%**
without adaptive buffering	11394	11438	11357	
data blocking reduction rate	**0.4827%**	**0.8655%**	**0.1585%**	
with adaptive buffering	11339	11340	11340	**blocking rate = 55%**
without adaptive buffering	11965	11962	11803	
data blocking reduction rate	**5.2319%**	**5.1998%**	**3.9227%**	
with adaptive buffering	11341	11340	11339	**blocking rate = 60%**
without adaptive buffering	13050	13191	12984	
data blocking reduction rate	**13.0958%**	**14.0323%**	**12.6694%**	
with adaptive buffering	11425	11345	11343	**blocking rate = 65%**
without adaptive buffering	14221	14398	14364	
data blocking reduction rate	**19.6611%**	**21.2043%**	**21.0317%**	
with adaptive buffering	12275	11342	11341	**blocking rate = 70%**

without adaptive buffering	16420	16369	16392	
data blocking reduction rate	**25.2436%**	**30.7105%**	**30.8138%**	
with adaptive buffering	15221	11428	11363	**blocking rate = 75%**
without adaptive buffering	19018	19042	19330	
data blocking reduction rate	**19.9653%**	**39.9853%**	**41.2157%**	
with adaptive buffering	19023	14440	11615	**blocking rate = 80%**
without adaptive buffering	23197	23390	23851	
data blocking reduction rate	**17.9937%**	**38.2642%**	**51.3018%**	
with adaptive buffering	26497	21118	16449	**blocking rate = 85%**
without adaptive buffering	30755	30804	31892	
data blocking reduction rate	**13.8449%**	**31.4440%**	**48.4228%**	
with adaptive buffering	42553	35869	30773	**blocking rate = 90%**
without adaptive buffering	47868	47983	47040	
data blocking reduction rate	**11.1035%**	**25.2464%**	**34.5812%**	
with adaptive buffering	84987	77393	68708	**blocking rate = 95%**
without adaptive buffering	93615	94141	95188	
data blocking reduction rate	**9.2165%**	**17.7903%**	**27.8186%**	

Table 5.2 data blocking reduction rate under various blocking rate of receiver and sizes of borrowed memory blocks

Table 5.2 shows the reduction rate of data blocking under various sizes of borrowed memory blocks and blocking rate of receiver.

Here, we limit the size of memory borrowed blocks to see the trend of reduction rate versus various blocking rate of receiver. For the same setting of size of input and output queue, the reduction rate will reach the maximum value under the memory borrowed blocks sets to 32 words when blocking rate of receiver equals 70% ~ 80%. In other words, when the blocking rate of receiver exceeds 80% under this setting, the reduction rate of data blocking condition would not increase. Figure 5.18 shows the results.

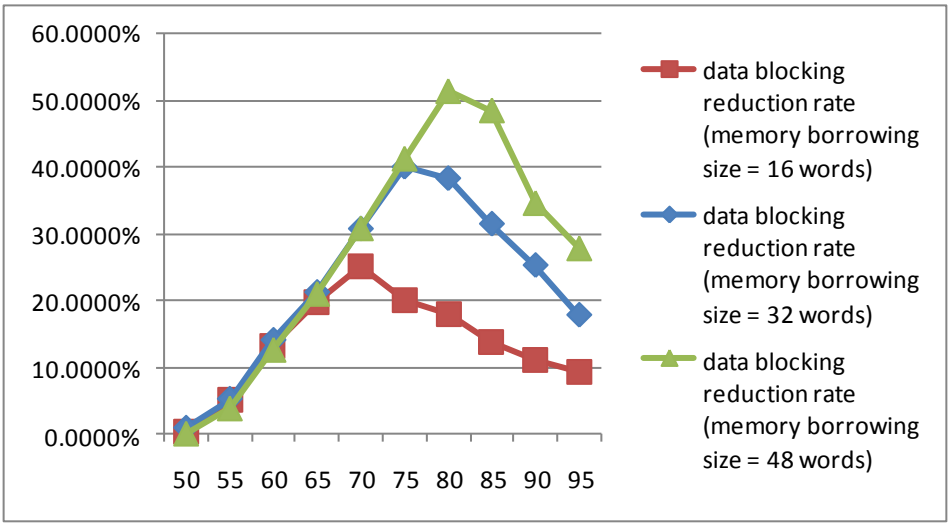


Figure 5.18 trend of data blocking reduction rate under various sizes of memory borrowing blocks

We can see the same trend when size of memory borrowed blocks equals 16 words and 48 words. When the blocking rate of receiver equals 70%, the reduction rate of data blocking condition would reach maximum under 16 words memory borrowed blocks. The improvement of data blocking reduction rate increases with the size of memory borrowed blocks. We can see the same situation under memory borrowed block equals 48 words.

5.5 Summary

We propose an on-demand memory system architecture shown in Figure 5.16 to verify the memory-centric on-chip data communication for wireless video entertainment system. The memory-centric on-chip interconnection network can reduce the data blocking condition and improve the performance. When the size of borrowed memory blocks increases, the data blocking reduction rate can have further improvement under limited size of borrowed memory blocks. Similarly, we also can adjust the size of output queue of sender and input queue of receiver to improve data transfer efficiency. For the MAC, the data blocking reduction rate can reach to 39.98% under the setting of output queue equals 16 words, input queue equals 32 words and maximum borrowed memory blocks equals 32 words.

Chapter 6
Conclusions and Future Work

6.1 Conclusions

A memory-centric on-chip interconnection network provides effective memory and communication bandwidths for on-chip SRAM-rich SoCs based on memory management units. Additionally, network interface is a major factor in the performance of on-chip interconnection networks. In this thesis, an efficient network interface for the memory-centric on-chip interconnection network is presented to reduce the data blocking via a borrowing mechanism. The d-MMU can dynamically allocate the memory resources for buffering the blocking network data. By considering the borrowed memory blocks and d-MMU, the size of the output queue for sender and the input queue for receiver in NI can be dynamically scheduled to improve the data transfer efficiency. According to the cycle-driven simulation results in SystemC, the proposed efficient NI can achieve performance improvement by 1.15x compared to the conventional one. Under different data injection load and output queue size we can reduce 2%~5% data blocking condition.

For the proposed on-demand memory system, the memory-centric on-chip interconnection network can efficiently reduce data blocking condition by adjusting the size of queue and borrowed memory blocks. Under limited size of borrowed memory blocks (16 words) and 70% blocking rate of receiver, the reduction rate of data blocking can reach 25%. Moreover, this OCIN provide a communication platform between various processing cores for wireless video entertainment systems. Therefore, the proposed efficient NI can increase the performance of the memory-centric on-chip interconnection network.

6.2 Future Work

For the future continued expansion of demand in the quantity and quality of multimedia service, multi-view 3D video technology becomes future star of global multimedia industry. At the same time, facing the growing mobile 3D multimedia services, the wireless communication infrastructure based on inherent Macrocell has become the bottleneck of service. How to integrate different network environment, home appliances, and video entertainment system in a heterogeneous integrated platform to establish a user-friendly and energy-efficient digital environment and system has become a very important topic. An efficient communication platform is essential.

For this memory-centric on-chip interconnection network, we use mesh-based router. We would extend to other various topologies to increase its scalability. We have already verified this memory-centric on-chip interconnection network by cycle-driven SystemC simulator in this phase. Later, we would use Verilog and SystemC for co-simulation to gain further improvement.

The eHome project is still going on. For eH-III project, a femtocell home multimedia center will be developed for supporting multi-view 3D video, high-speed MIMO OFDM and gigabit cross-layer RRM in a heterogeneous platform. The architecture is shown in Figure 6.1. In the future, in order to support huge memory bandwidth and data transmitting requirements, it will be necessary that constructing a heterogeneous memory-centric multi-core platform for multimedia center.

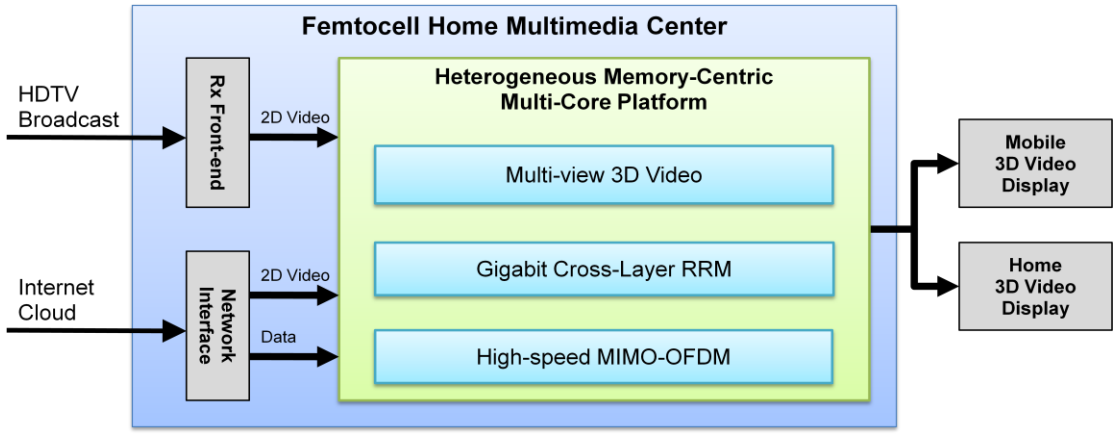


Figure 6.1 Architecture of femtocell home multimedia center

Bibliography

[1.1] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. Van Der Wijngaart, and T. Mattson, "A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS," *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC),pp.108-109, Feb. 2010.

[1.2] N. A. Kurd, S. Bhamidipati, C. Mozak, J. L. Miller, T. M. Wilson, M. Nemani, and M. Chowdhury, "Westmere: A family of 32nm IA processors," *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC), pp.96-97, Feb. 2010.

[1.3] J. L. Shin, K. Tam, D. Huang, B. Petrick, H. Pham, C.-K. Hwang, H.-P. Li, A. Smith, T. Johnson, F. Schumacher, D. Greenhill, A. S. Leon, and A. Strong, "A 40nm 16-core 128-thread CMT SPARC SoC processor," *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC), pp.98-99, Feb. 2010.

[1.4] Y. Yuyama, M. Ito, Y. Kiyoshige, Y. Nitta, S. Matsui, O. Nishii, A. Hasegawa, M. Ishikawa, T. Yamada, J. Miyakoshi, K. Terada, T. Nojiri, M. Satoh, H. Mizuno, K. Uchiyama, Y. Wada, K. Kimura, H. Kasahara, and H. Maejima, "A 45nm 37.3GOPS/W heterogeneous multi-core SoC," *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC), pp.100-101, Feb. 2010.

[1.5] C. Johnson, D. H. Allen, J. Brown, S. Vanderwiel, R. Hoover, H. Achilles, C.-Y. Cher, G. A. May, H. Franke, J. Xenedis, and C. Basso, "A wire-speed power™ processor: 2.3GHz 45nm SOI with 16 cores and 64 threads," *2010 IEEE International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC), pp.104-105, Feb. 2010.

[1.6] R. Marculescu, U. Y. Ogras, Li-Shiuan Peh, N. E. Jerger, and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*,vol.28, no.1, pp.3-21, Jan. 2009.

[1.7] R. Iris Bahar, Dan Hammerstrom, Justin Harlow, William H. Joyner Jr., Clifford Lau, Diana Marculescu, Alex Orailoglu, and Massoud Pedram, “Architectures

for Silicon Nanoelectronics and Beyond," *IEEE Computer*, vol. 40, no. 1, pp. 25-33, Jan. 2007.

[1.8] M. Moadeli, P.P. Maji and W. Vanderbauwhede, "Design and implementation of the Quarc Network on-Chip," *IEEE International Symposium on Parallel & Distributed Processing* (IPDPS), pp.1-9, May 2009.

[1.9] A. Avakian, J. Nafziger, A. Panda, and R. Vemuri, "A reconfigurable architecture for multicore systems," *IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum* (IPDPSW), pp.1-8, April 2010.

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

[2.1] K. Warathe, D. Padole and P. Bajaj, "A Design Approach to AMBA (Advanced Microcontroller Bus Architecture) Bus Architecture with Dynamic Lottery Arbiter," *2009 Annual IEEE India Conference* (INDICON), pp.1-4, Dec. 2009.

[2.2] Younjin Jung, Ok Kim, Byoungyup Lee, Hongkyun Jung, and Kwangki Ryoo, "SoC platform design with multi-channel bus architecture," *International SoC Design Conference* (ISOCC), vol.03, pp.III-48-III-49, Nov. 2008.

[2.3] M. Copploa, "Trends and trade-offs in designing highly robust throughput on chip communication network," *IEEE International On-Line Testing Symposium*, July 2006.

[2.4] V. Chandra, A. Xu, H. Schmit, and L. Pileggi, "An interconnect channel design methodology for high performance integrated circuits," *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, Vol. 2, pp. 1138 – 1143, 2004.

[2.5] H. Lekatsas and J. Henkel, "ETAM++: extended transition activity measure for low power address bus designs," *VLSID*, pp. 113-120, 2002.

[2.6] A. Ganguly, P.P. Pande and B. Belzer, "Crosstalk-Aware Channel Coding Schemes for Energy Efficient and Reliable NOC Interconnects," *IEEE Transactions on Very Large Scale Integration Systems*, vol.17, no.11, pp.1626-1639, Nov. 2009.

[2.7] Jih-Sheng Shen, Chun-Hsian Huang and Pao-Ann Hsiung, "PRESSNoC: Power-Aware and Reliable Encoding Schemes Supported Reconfigurable Network-on-Chip Architecture," *4th International Conference on Embedded and*

Multimedia Computing (EM-Com), pp.1-6, Dec. 2009.

[2.8] Jih-Sheng Shen, Chun-Hsian Huang and Pao-Ann Hsiung, "Learning-based adaptation to applications and environments in a reconfigurable Network-on-Chip," *Design, Automation & Test in Europe Conference & Exhibition* (DATE), pp.381-386, March 2010.

[2.9] P.P. Pande, Haibo Zhu, A. Ganguly, and C. Grecu, "Crosstalk-aware Energy Reduction in NoC Communication Fabrics," *IEEE International SOC Conference*, pp.225-228, Sept. 2006.

[2.10] M. Taassori and S. Hessabi, "Low Power Encoding in NoCs Based on Coupling Transition Avoidance," *Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, pp.247-254, Aug. 2009.

[2.11] Yi Liu, Yintang Yang, Yadong Jiao, and Ning Wang, "An Encoding Drive Approach to Reduce Signal Jitter of Interconnection Lines between NoC Routers," *2nd International Congress on Image and Signal Processing* (CISP), pp.1-4, Oct. 2009.

[2.12] S. R. Sridhara and N. R. Shanbhag, "Coding for reliable on-chip buses: fundamental limits and practical codes," *Proceedings of IEEE International Conference on VLSI Design*, pp. 417-422, Jan. 2005.

[2.13] F. Worm, P. Ienne, P. Thira, and G. DeMicheli, "A Robust Self-Calibrating Transmission Scheme for On-Chip Networks," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, pp. 126-139, 2004.

[2.14] Y. Thonnart, P. Vivet and F. Clermidy, "A fully-asynchronous low-power framework for GALS NoC integration," *Design, automation & Test in Europe Conference & Exhibition* (DATE), pp.33-38, March 2010.

[2.15] Y. Zhiyi and B. M. Baas, "A Low-Area Multi-Link Interconnect Architecture for GALS Chip Multiprocessors," *IEEE Transactions on Very Large Scale Integration Systems*, vol.18, no.5, pp.750-762, May 2010.

[2.16] Y. Thonnart, E. Beigne and P. Vivet, "Design and Implementation of a GALS Adapter for ANoC Based Architectures," *15th IEEE Symposium on Asynchronous Circuits and Systems* (ASYNC), pp.13-22, May 2009.

[2.17] M.A.U. Rahman, I. Ahmed, F. Rodriguez, and N. Islam, "Efficient 2DMesh Network on Chip (NoC) Considering GALS Approach," *Fourth International Conference on Computer Sciences and Convergence Information Technology*

(ICCIT), pp.841-846, Nov. 2009.

[2.18] E. Beigne, F. Clermidy, H. Lhermet, S. Miermont, Y. Thonnart, Xuan-Tu Tran, A. Valentian, D. Varreau, P. Vivet, X. Popon, and H. Lebreton, "An Asynchronous Power Aware and Adaptive NoC Based Circuit," *IEEE Journal of Solid-State Circuits*, vol.44, no.4, pp.1167-1177, April 2009.

[2.19] D. Gebhardt, J. You and K. S. Stevens, "Comparing Energy and Latency of Asynchronous and Synchronous NoCs for Embedded SoCs," *Fourth ACM/IEEE International Symposium on Networks-on-Chip* (NOCS), pp.115-122, May 2010.

[2.20] M. Drinic, D. Kirovski, S. Megerian, and M. Potkonjak, "Latency-Guided On-Chip Bus-Network Design," *IEEE Transactions on computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, Issue 12, pp. 2663-2673, Dec. 2006.

[2.21] Byungyong Kim and Chanho Lee, "High performance on-chip-network architecture with multiple channels and dual routing," *International SoC Design Conference* (ISOCC), vol.03, no., pp.III-33-III-34, Nov. 2008.

[2.22] A. M. Amory, K. Goossens, E. J. Marinissen, M. Lubaszewski, and F. Moraes, "Wrapper design for the reuse of a bus, network-on-chip, or other functional interconnect as test access mechanism," *Computers & Digital Techniques, IET*, Vol. 1,Issue 3, pp. 197 – 206, May 2007.

[2.23] J. Diemer, R. Ernst and M. Kauschke, "Efficient throughput-guarantees for latency-sensitive networks-on-chip," *Asia and South Pacific Design, automation Conference* (ASP-DAC), pp.529-534, Jan. 2010.

[2.24] J. Diemer and R. Ernst, "Back Suction: Service Guarantees for Latency-Sensitive On-chip Networks," *Fourth ACM/IEEE International Symposium on Networks-on-Chip* (NOCS), vol., no., pp.155-162, May 2010.

[2.25] D. Gebhardt, J. You and K. S. Stevens, "Comparing Energy and Latency of Asynchronous and Synchronous NoCs for Embedded SoCs," *Fourth ACM/IEEE International Symposium on Networks-on-Chip* (NOCS), pp.115-122, May 2010.

[2.26] P. Guerrier and A. Greiner, "A Generic Architecture for On-Chip Packet Switched Interconnections," *Proceedings of Design, automation and Test in Europe Conference and Exhibition*, pp. 250 - 256, Mar. 2006.

[2.27] A. Ivanov and G. De-Micheli, "Guest Editors' Introduction: The Network-on-Chip Paradigm in Practice and Research," *IEEE Design and Test of*

Computers, vol. 22, Issue 5, pp. 399-403, Sep. 2005.

[2.28] M. Dehyadgari, M. Nickray, A. Afzali-kusha and, Z. Navabi, “A new protocol stack model for network on chip,” *IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, March 2000.

[2.29] M. Tudruj and L. Masko, “Dynamic SMP Clusters with Communication on the Fly in NoC Technology for Very Fine Grain Computations**”,** *Parallel and Distributed Computing*, 2004. *Third International Symposium on/Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, pp. 97-104, 2004.

[2.30] P.G. Paulin, C. Pilkington, E. Bensoudane, M. Langevin and, D. Lyonnard, “Application of a multi-processor SoC platform to high-speed packet forwarding”, *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, volume 3, pp. 58-63, 2004.

[2.31] M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, S. K. Mathew, F. Sheikh, R. K. Krishnamurthy, and S. Borkar, "A 4.1Tb/s bisection-bandwidth 560Gb/s/W streaming circuit-switched 8×8 mesh network-on-chip in 45nm CMOS," *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC), pp.110-111, 7-11 Feb. 2010.

[2.32] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet and, N. Wehn, "A 477mW NoC-based digital baseband for MIMO 4G SDR," *International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC), pp.278-279, Feb. 2010.

[2.33] F. Karim, A. Nguyen and S. Dey, “An Interconnect Architecture for Networking Systems on Chips,” *IEEE Micro*, vol. 22, no. 5, pp. 36-45, Sept./Oct. 2002.

[2.34] D. Bertozzi and L. Benini, “Xpipes: a network-on-chip architecture for gigascale systems-on-chip,” *Circuits and Systems Magazine*, Volume 4, pp 18-31, 2004.

[2.35] G. Passas, M. Katevenis and D. Pnevmatikatos, "A 128 x 128 x 24Gb/s Crossbar Interconnecting 128 Tiles in a Single Hop and Occupying 6% of Their Area," *Fourth ACM/IEEE International Symposium on Networks-on-Chip* (NOCS), pp.87-95, May 2010.

[2.36] Y. Thonnart, R. Lemaire and F. Clermidy, "Distributed Sequencing for

Resource Sharing in Multi-applicative Heterogeneous NoC Platforms," *Fourth ACM/IEEE International Symposium on Networks-on-Chip* (NOCS), pp.233-240, May 2010.

[2.37] M. Saneei, A. Afzali-Kusha and Z. Navabi, "Low-power and low-latency cluster topology for local traffic NoCs," *IEEE International Symposium on Circuits and Systems*, 2006.

[2.38] S. Bourduas and Z. Zilic, "A Hybrid Ring/Mesh Interconnect for Network-on-Chip Using Hierarchical Rings for Global Routing," *First International Symposium on Networks-on-Chip*, pp. 195-204, 2007.

[2.39] Z. Marrakchi, H. Mrabet, C. Masson, and H. Mehrez, "Mesh of Tree: Unifying Mesh and MFPGA for Better Device Performances," *First International Symposium on Networks-on-Chip*, pp. 243-252, 2007.

[2.40] V. F. Pavlidis and E. G. Friedman, "3-D Topologies for Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.15, no.10, pp.1081-1090, Oct. 2007.

[2.41] F. A. Samman, T. Hollstein and M. Glesner, "Adaptive and Deadlock-Free Tree-Based Multicast Routing for Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration Systems*, vol.18, no.7, pp.1067-1080, July 2010.

[2.42] V. Chandra, A. Xu, H. Schmit and L. Pileggi, "An interconnect channel design methodology for high performance integrated circuits**",** *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pp. 1138-1143, 2004.

[2.43] K. Latif, T. Seceleanu and H. Tenhunen, "Power and Area Efficient Design of Network-on-Chip Router through Utilization of Idle Buffers," *IEEE International Conference and Workshops on Engineering of Computer Based Systems* (ECBS), pp.131-138, March 2010.

[2.44] R. S. Ramanujam, V. Soteriou, B. Lin and, Li-Shiuan Peh, "Design of a High-Throughput Distributed Shared-Buffer NoC Router," *Fourth ACM/IEEE International Symposium on Networks-on-Chip* (NOCS), pp.69-78, May 2010.

[2.45] V. Soteriou, R.S. Ramanujam, B. Lin, and Li-Shiuan Peh, "A High-Throughput Distributed Shared-Buffer NoC Router," *Computer Architecture Letters* , vol.8, no.1, pp.21-24, Jan. 2009.

[2.46] Zhe Zhang, Xiaoming Hu and Lili Cui, "A synthesizable on-chip wormhole router," *Asia-Pacific Conference on Computational Intelligence and Industrial*

Applications (PACIIA), vol.2, pp.193-196, Nov. 2009.

[2.47] L. Kangmin, L. Se-Joong, and Y. Hoi-Jun, "Low-power network-on-chip for high-performance SoC design," *IEEE Transactions on Very Large Scale Integration Systems*, , vol. 14, pp. 148-160, 2006.

[2.48] Liu Zheng, Cai Jueping, Du Ming, Yao Lei and, Li Zan, "Hybrid Communication Reconfigurable Network on Chip for MPSoC," *IEEE International Conference on Advanced Information Networking and Applications* (AINA), pp.356-361, April 2010.

[2.49] A. Ruadulescu, K. Goossens, G. De Micheli, S. Murali, and M. Coenen, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," *Proceedings of the 4th international conference on Hardware/Software Codesign and System Synthesis* (CODES+ISSS), pp. 130-135, 2006.

[2.50] T. D. Richardson, C. Nicopoulos, D. Park, V. Narayanan, X. Yuan, C. Das, and V. Degalahal, "A hybrid SoC interconnect with dynamic TDMA-based transaction-less buses and on-chip networks," *Held jointly with 5th International Conference on Embedded Systems and Design, 19th International Conference on VLSI Design*, 2006.

[2.51] Yun-Lung Lee, Jer Min Jou and Yen-Yu Chen, "A high-speed and decentralized arbiter design for NoC," *IEEE/ACS International Conference on Computer Systems and Applications* (AICCSA), pp.350-353, May 2009.

[2.52] X. Gao, Z. Zhang and X. Long, "Round Robin Arbiters for Virtual Channel Router," *IMACS Multiconference on Computational Engineering in Systems Applications*, pp. 1610-1614, 2006.

[3.1] Jose Duato, Sudhakar Yalamanchili and Lionel Ni, “Interconnection Networks – An Engineering Approach”.

[3.2] Yue Qian, Zhonghai Lu and Wenhua Dou, "Analysis of Worst-Case Delay Bounds for On-Chip Packet-Switching Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.29, no.5, pp.802-815, May 2010.

[3.3] I. Nousias and T. Arslan, "Wormhole Routing with Virtual Channels using Adaptive Rate Control for Network-on-Chip (NoC)," *First NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, vol., no., pp.420-423,

15-18 June 2006.

[3.4] L.I. Tabada and P.U. Tagle, "Shared Buffer Approach in Fault Tolerant Networks," *International Conference on Computer Technology and Development (ICCTD)*, vol.1, no., pp.235-239, 13-15 Nov. 2009.

[3.5] Xiao Canwen, Zhang Minxuan, Dou Yong, and Zhao Zhitong, "Dimensional Bubble Flow Control and Fully Adaptive Routing in the 2-D Mesh Network on Chip," *International Conference on Embedded and Ubiquitous Computing (EUC)*, vol.1, pp.353-358, 17-20 Dec. 2008.

[3.6] A. Pullini, F. Angiolini, D. Bertozzi, and L. Benini, "Fault Tolerance Overhead in Network-on-Chip Flow Control Schemes," *18th Symposium on Integrated Circuits and Systems Design*, pp.224-229, Sept. 2005.

[3.7] Teck Peow Lee and J. Siliquini, "Deficit round robin with hop-by-hop credit based flow control," *IEEE Region 10 Conference* (TENCON), pp.1-4, Oct. 30 2007-Nov. 2 2007.

[3.8] N. Concer, L. Bononi, M. Soulie, R. Locatelli, and L. P. Carloni, "CTC: An end-to-end flow control protocol for multi-core systems-on-chip," *3rd ACM/IEEE International Symposium on Networks-on-Chip* (NoCS), vol., no., pp.193-202, 10-13 May 2009.

[3.9] William James Dally and Brian Towles, “Principles and Practices of Interconnection Networks”.

[3.10] F. Gilabert, M. E. Gómez, S. Medardoni, and D. Bertozzi, "Improved Utilization of NoC Channel Bandwidth by Switch Replication for Cost-Effective Multi-processor Systems-on-Chip," *Fourth ACM/IEEE International Symposium on Networks-on-Chip* (NOCS), pp.165-172, May 2010.

[4.1] L. Benini and G. De Micheli, Network on Chips: Technology and Tools, Morgan Kaufmann, 2006.

[4.2] N. Concer, L. Bononi, M. Soulie, R. Locatelli, and L. P. Carloni, "The Connection-Then-Credit Flow Control Protocol for Heterogeneous Multicore Systems-on-Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.29, no.6, pp.869-882, June 2010.

[4.3] A. Radulescu, J. Dielissen, S.G. Pestana, O.P. Gangwal, E. Rijpkema, P. Wielage, and K. Goossens, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," IEEE

Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.24, no.1, pp. 4- 17, Jan. 2005.

[4.4] F. Clermidy, R. Lemaire, Y. Thonnart, and P. Vivet, “A Communication and Configuration Controller for NoC based Reconfigurable Data Flow Architecture,” *Proceedings of ACM/IEEE International Symposium on Networks-on-Chip*, pp. 153–162, May. 2009.

[4.5] Y.-L. Lai, S.-W. Yang, M.-H. Sheu, Y.-T. Hwang, H.-Y. Tang, and P.-Z. Huang, “A High-Speed Network Interface Design for Packet-Based NoC,” *Proceedings of IEEE International Conference on Communication, Circuits and Systems*, pp. 2667–2671, 2006.

[4.6] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. Van Der Wijngaart, and T. Mattson, "A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS," *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (ISSCC),pp.108-109, Feb. 2010.

[4.7] 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.

[4.8] H. Saito, M. Nakajima, T. Okamoto, Y. Yamada, A. Ohuchi, N. Iguchi, T. Sakamoto, K. Yamaguchi, and M. Mizuno, "A Chip-Stacked Memory for On-Chip SRAM-Rich SoCs and Processors," *IEEE Journal of Solid-State Circuits*, vol.45, no.1, pp.15-22, Jan. 2010.

[4.9] P. Giaccone, E. Leonardi and D. Shah, "Throughput Region of Finite-Buffered Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol.18, no.2, pp.251-263, Feb. 2007.

[4.10] P. Giaccone, E. Leonardi and D. Shah, "On the maximal throughput of networks with finite buffers and its application to buffered crossbars," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol.2, pp. 971- 980 vol. 2, 13-17 March 2005.

[5.1] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, Liewei Bao, J. Brown, M. Mattina, Chyi-Chang Miao, C. Ramey, D.

Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook, "TILE64 - Processor: A 64-Core SoC with Mesh Interconnect", *IEEE International Solid-State Circuits Conference*, 2008, ISSCC 2008, pp.88-598, 3-7 Feb. 2008.

[5.2] Yunxin Li, “Cognitive and Integrated Digital Home via Dynamic Media Access”, *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting* (BMSB), pp. 1-6, May 2009.

[5.3] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol.17, no.9, pp.1103-1120, Sept. 2007.

Vita

王湘斐
Shiang-Fei Wang

PERSONAL INFORMATION

Birth Date: April. 18, 1982

Birth Place: Tainan, **TAIWAN**

E-Mail Address: fila.ee94g@nctu.edu.tw

EDUCATION

09/2005 – 07/2010 M.S. in Institute of Electronics Engineering, National Chiao Tung University
Thesis: Memory-Centric On-Chip Interconnection Network for Wireless Video Entertainment System

09/2000 – 06/2005 B.S. in Department of Electronics, National Chiao Tung University

PUBLICATIONS

Po-Tsang Huang, Yung Chang, **Shiang-Fei Wang** and Wei Huang, "An Efficient Network Interface for Memory-Centric On-Chip Interconnection Network," IEEE Asia Pacific Conference on Circuit and Systems, APCCAS, Sept, 2010. (*Submitted*)