High Performance Computing Architecture with Security

by
He Zhou

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2015

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by He Zhou, titled High Performance Computing Architecture with Security and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

_____ Date: 07/29/2015
Janet M. Roveda, Ph.D.

_____ Date: 07/29/2015
Linda S. Powers, Ph.D.

_____ Date: 07/29/2015
Roman Lysecky, Ph.D.

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

_____ Date: 07/29/2015
Dissertation Director: Janet M. Roveda

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: He Zhou

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Janet M. Revoda, for the technical guidance, constant encouragements and supports she offered me for the past four years. I would like to thank you for your significant contributions to this work.

I would like to thank my doctoral dissertation committee members, Dr. Linda S. Powers and Dr. Roman Lysecky, for spending your precious time in reading my dissertation, providing me feedbacks, helping me in corrections, and attending my dissertation defense.

I would like to thank Dr. Kathleen Melde and Sungjong Yoo for your on-chip antenna model and data. I would like to thank Dr. Kathleen Melde for the technical antenna guidance you provided me.

I would like to thank my senior group member, Dr. Jin Sun, for your guidance and help to this work.

I would like to thank all of my group members in the VLSI lab and all of my friends in Department of Electrical and Computer Engineering. You guys helped me a lot during my Ph.D. study and research period. I had lots of fun and beautiful memories in this group, in this department.

Last, but not the least, I thank my family members, my mother, my father, and my grandparents. I could never become who I am now without your supports and your unconditional love.

TABLE OF CONTENTS

TABLE OF CONTENTS-CONTINUED

TABLE OF CONTENTS-CONTINUED

LIST OF FIGURES

LIST OF FIGURES-CONTINUED

LIST OF FIGURES-CONTINUED

LIST OF FIGURES-CONTINUED

LIST OF TABLES

ABSTRACT

Multi-processor embedded system is the future promise of high performance computing architecture. However, it still suffers low network efficiency and security threat. Simply upgrading to multi-core systems has been proven to provide only minor speedup compared with single core systems. Router architecture of network-on-chip (NoC) uses shared input buffers such as virtual channels and crossbar switches that only allow sequential data access. The speed and efficiency of on-chip communication is limited. In addition, the performance of conventional NoC topology is limited by routing latency and energy consumption due to its network diameter increases with the rising number of nodes.

The security concern has also become a serious problem for embedded systems. Even with cryptographic algorithms, embedded systems are still very vulnerable to side channel attacks (SCAs). Among SCA approaches, power analysis is an efficient and powerful attack. Once the encryption location in an instruction sequence is identified, power analysis can be applied to exploit the embedded system.

To improve on-chip network parallelism, this dissertation proposes a new router microarchitecture based on a new data structure called virtual collision array. Sequential data requests are partially eliminated in the virtual collision array before entering router pipeline. To facilitate the new router architecture, new workload assignment is applied to increase data request elimination. Through a task flow partitioning algorithm, we minimize sequential data

access and then schedule tasks while minimizing the total router delay. For NoC topology, this dissertation presents a new hybrid NoC (HyNoC) architecture. We introduce an adaptive routing scheme to provide reconfigurable on-chip communication with both wired and wireless links. In addition, based on a mathematical model which established on cross-correlation, this dissertation proposes two obfuscation methodologies: *Real Instruction Insertion* and *AES Mimic* to prevent SCAs power analysis attack.

CHAPTER 1 INTRODUCTION

1.1 Multi-processor Embedded System

Embedded systems possess properties of small size, rugged operating ranges, low power consumption, and low cost/unit. They are playing important roles in every day peoples' lives. As shown in Fig. 1-1, no matter the embedded system is used in which application, consumer electronics (smart watch, personal digital assistant, MP3 player, DVD player, videogame console, digital camera), communication systems (smart phone, modem, router, network switch, firewall gateway), medical equipment (vital signs monitor, electronic stethoscope, medical imaging equipment), or transportation systems (electric vehicle, hybrid car, global positioning system), it is always facing a common requirement: an insatiable demand for higher performance.

Fig. 1-1 Embedded system applications

In the past decades, this performance demand is majorly satisfied by Moore's Law: the number of components in an integrated chip (IC) will double every 18 to 24 months [1] [2]. As shown in Fig. 1-2 [3], till 2011, transistor count is doubling every two years. It can be roughly summarized as the performance demand is satisfied by "shrinking the silicon and increasing the clock speed" [3]. However, we cannot improve embedded system performance forever by just increasing IC's transistor count. Due to the immutable physical constants such as speed of light, the of electrons, and operating temperature limits of silicon, Moore's Law can no longer be relied on to provide extra performance [3].

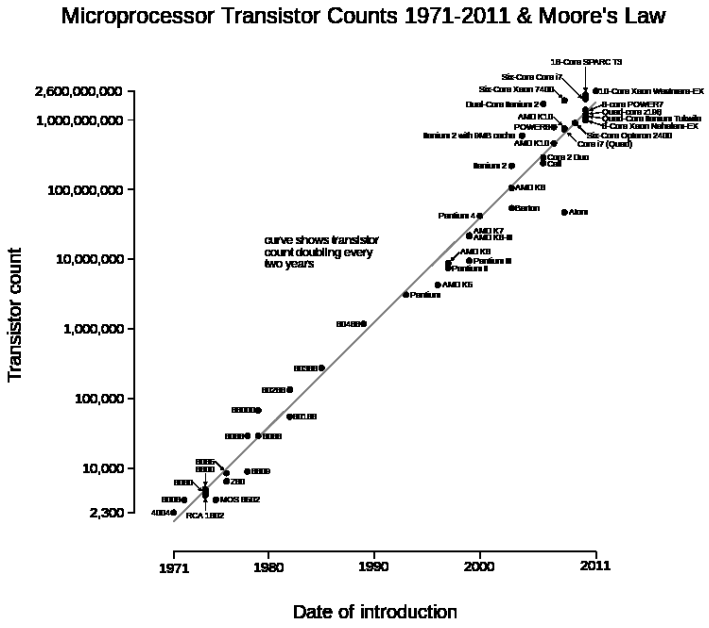


Fig. 1-2 Moore's Law [3]

As we can no longer improve embedded system performance by increasing IC's transistor count, does it mean that the embedded system has reached its heyday and can no longer performances better? The answer is absolutely no! Researchers have found the solution: multi-processors embedded system [4-6]. The previous successes achieved by IBM, Larrabee, Intel, and Tilera have already demonstrated 32 cores, 64 cores, 80 cores, and even 100 cores on a single die [7] [8]. Multi-processor embedded system has become a vital part for the high speed computing architecture.

1.2 Network-on-chip for Multi-core System

When an application is applied on a multi-core system, its task flow will be partitioned into several sub-flows with one or more tasks. These sub-flows will be assigned to and executed on different processors (this will be discussed in details in Chapter 2). Execution results of a task will be fed into the successor task as input data. Data transmission between different tasks on different processors is realized by network-on-chip (NoC). As shown in Fig. 1-3, the on-chip network has routers at every node. The router is connected to the central processor and neighbor nodes via short local interconnects.

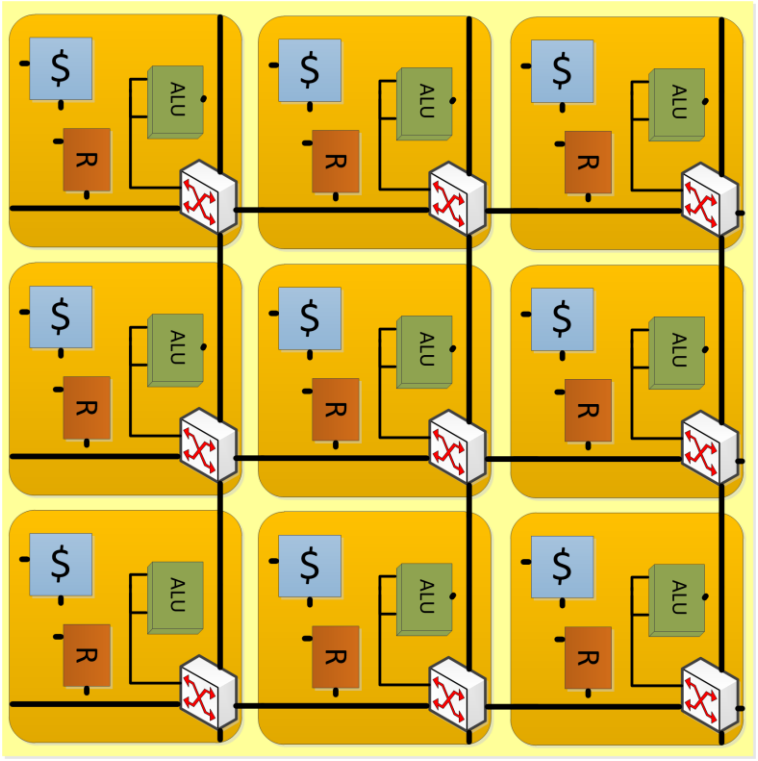


Fig. 1-3 Multi-core system with network-on-chip

A NoC architecture has four critical parameters: NoC topology, routing algorithm, flow control protocol, and router microarchitecture [2]. The NoC topology dictates the number of nodes in the platform. It also indicates how the nodes are connected by local links. While the NoC topology determines all possible paths a communication data can traverse through the network, the routing algorithm selects the specific path the communication data will take from its source to its destination. The flow control protocol determines the traversing detail for the communication data. For example, it specifies the timestamp of the communication data leaving a router and the timestamp of the communication data being buffered. Finally, the router

microarchitecture provides the router circuits' implementation. It realizes the routing and flow control protocols. In Chapter 2 and Chapter 3, we will discuss the router microarchitecture and NoC topology in details.

In recent years, the leading chip-industry companies, such as IBM, Larrabee, Intel, and Tilera, have released a wide variety of multi-core productions [7] [8]. It is not hard to find out the trend of the multi-core industry: the core count is getting higher and higher. "Big data" is another trend for the multi-core system. Nowadays, data transmission between different cores can reach hundreds of megabytes, even gigabytes. As the number of on-chip cores and transmitting data volume increase, a scalable and high-bandwidth network-on-chip becomes critically important [10].

1.3 Side Channel Attack and Embedded System Security

Due to the success in system-on-chip (SoC) architecture, embedded system based products are getting more and more popular in all aspects of people's lives. Along with the soaring prosperity of embedded system, side channel attack (SCA) also becomes a major concern for system security [11-15].

SCAs refer to the security attacks deployed by hackers to unlawfully obtain system secret keys. The hacker measures system properties such as power consumption [16], electromagnetic (EM) emission [17] [18], and execution time [19] from the embedded processor while it's

running cryptographic programs. Then the hacker can correlate these system properties with the program structure to predict system secret keys.

Side channel attacks are targeting at exploiting leakage from the physical implementation of cryptographic algorithms. Over the years, new effective SCA approaches, such as simple power analysis (SPA) [16, 20], differential power analysis (DPA) [15] [21-24], correlation power analysis (CPA) [25], mutual information attack (MIA) [26], and partitioning power analysis (PPA) [27], are ceaselessly suggested. Among all of the SCA approaches, DPA is the most efficient and powerful attack. We will discuss this SCA approach in details in Chapter 4.

1.4 Research Objective

The objective of this dissertation is to: 1) Discuss the defects and limitations of current NoC designs and SCA countermeasures; 2) Provide a new NoC router microarchitecture design and a new hybrid NoC topology design; 3) Provide a new power obfuscation model to defend power analysis in side channel attack.

Contributions of this work:

- Proposed a new router microarchitecture based on virtual collision array. This virtual collision array is placed right before the router input buffers to reduce the number of sequential data access to router pipeline.

- Designed new task flow partitioning and task scheduling algorithms facilitating the new router microarchitecture to achieve minimal router delay while considering performance constraints and system constraints.
- Based on the millimeter-wave four-element-array on-chip antenna, proposed a new hybrid NoC topology with both wired and wireless communication channels.
- Designed new adaptive routing scheme facilitating the new hybrid NoC topology, which is able to dynamically choose a proper transmission path, wired or wireless, for each data packet during the whole routing process.
- Introduced two power profile obfuscation ideas *Real Instruction Insertion* and *AES Mimic*.
- Introduced a new concept referred to as *Dissimilarity Level* to provide an explicit assessment of power profile difference between the one without obfuscation and the one with obfuscation.

1.5 Dissertation Outline

Chapter 2 introduces the virtual collision array concept and the new router microarchitecture based on this concept. Performance evaluation includes packet latency, system throughput, speedup, power, and energy consumption between the proposed design and the conventional virtual channel router is compared. In Chapter 3, backgrounds of millimeter-wave on-chip antenna are discussed. The new hybrid NoC topology and its routing algorithm are

proposed in this chapter. On-chip antenna performance, such as P_{RE}, input impedance, reflection coefficient, one antenna gain pattern, and two antennas gain pattern, are presented and analyzed. Network performance of the proposed design is compared with the conventional mesh, torus, and butterfly NoC topologies. In Chapter 4, the cross-correlation based encryption obfuscation model is designed. The concept of *Dissimilarity Level* and its impact on differential power analysis is presented. Two power profile obfuscation ideas and the optimization process are also presented in this chapter. In Chapter 5, the conclusion remarks of this dissertation are discussed.

CHAPTER 2 COLLISION ARRAY BASED ROUTER MICROARCHITECTURE DESIGN FOR NETWORK-ON-CHIP

Router microarchitecture is one of the most important parameters to define on-chip network architecture. Router microarchitecture realizes on-chip network routing algorithm and flow control protocol, and finally determines the implementation of network circuitry. The inner structure of router microarchitecture determines the per-hop packet latency and overall routing delay. Good router microarchitecture can provide relatively low per-hop packet latency. With the same routing algorithm (the same hop count), the overall routing delay for an inter-core communication could be lower on the network. In addition, since router microarchitecture determines network circuitry components and their activities, it can also influence network power and energy consumption.

With dozens or even hundreds of processors integrated on single chip, a sophisticated network-on-chip is desired for providing efficient and reliable on-chip communication [28-31]. However, the current state of art on NoC router microarchitecture can only provide limited on-chip communication performance. Novel router microarchitectures are urgently needed.

This chapter is first devoted to discuss the Amdahl's law and limitations of current NoC router microarchitecture. This chapter introduces the collision array concept for common sequential components. We then propose a new virtual collision array based NoC router

microarchitecture. Workload assignment and task scheduling for the new NoC platform is illustrated in this chapter. Finally, this chapter presents the result analysis of the new design.

2.1 Amdahl's Law and Limitations of Current NoC Router Microarchitecture

With the soaring prosperity of system-on-chip architecture, the modern multi-core system design is still governed by the Moore's Law: the number of components in an integrated chip will double every 18 to 24 months [1] [2]. The previous successes achieved by IBM, Larrabee, Intel, and Tilera have already demonstrated 32 cores, 64 cores, 80 cores, and even 100 cores on a single die [7] [8]. Nevertheless, putting hundreds of cores on a single chip is not an easy task. The most challenging part is not about how many cores can be integrated on a single chip, but how to efficiently activate those cores and achieve the proposed performance improvement. Even with multiple cores, the SoC architecture usually cannot provide speedups equal to the number of cores in use. So where is the bottleneck? Amdahl's law gives an accurate explanation: speedup achieved in parallel computing depends on how much sequential computing is required [32] [33]. Let's assume that each core takes one unit time (normalized) to complete one operation and that p is the percentage of this operation which could be executed in parallel by a multi-core system. Then for n cores, the parallel portion needs time p/n and the sequential portion needs time $1-p$. Therefore, it takes time $1-p+p/n$ to complete the operation by n cores. The speedup, defined by the ratio of sequential time to the parallel time, can be expressed as:

$$s = 1/(1 - p + p/n) \tag{2.1}$$

Fig. 2-1 illustrates the relationship between speedup and number of cores for various percentages of parallelism. Even with only 10% of sequential computation, the speedup is less than 8 times with 30 cores. For cases where 60, 70 and 80% of computations are parallel, we see little improvement in terms of speedup when we have 20 to 30 cores. The speedup saturates at 2, 3, and 4 times, respectively. For a multi-core system, the speedup bottleneck is the inability to respond to simultaneous requests with a shared, common resource that only allows sequential operations/computations.

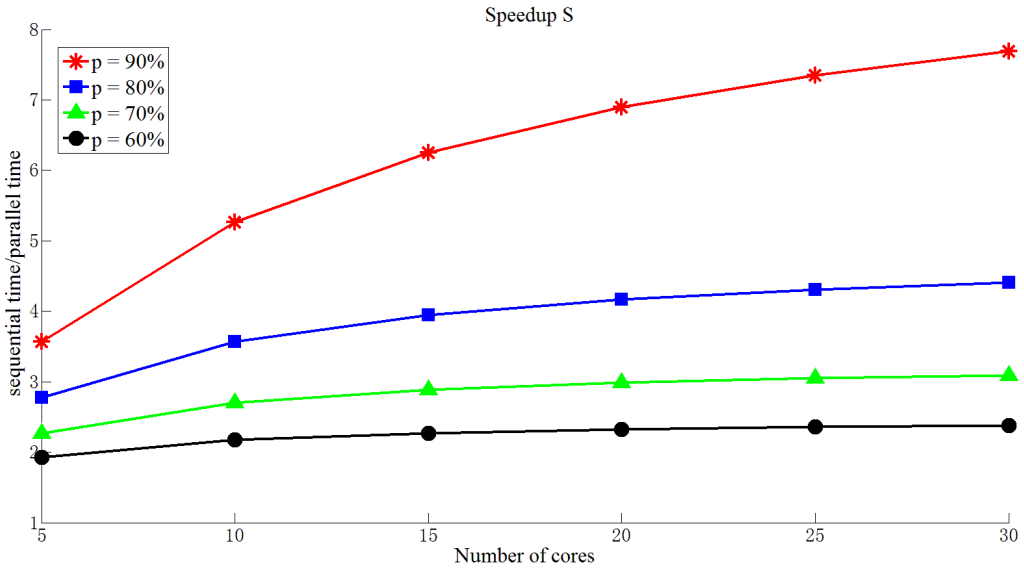


Fig. 2-1 Speedup vs number of cores

Following Amdahl's law, it is easy to understand why most multi-core systems have on-chip network I/O as their bottlenecks. With dozens or even hundreds of cores integrated on single chip, a sophisticated NoC is desired for providing efficient and reliable on-chip communication [28-31]. However, NoC router is not a concurrent structure. The conventional router employs shared input buffers such as virtual channels and crossbar switch that only allow sequential data access [9]. This leads to low speed and low efficiency of on-chip communication.

Modern on-chip networks use routers at every core to transmit data through the multi-core system. Fig. 2-2 depicts a 4x4 multi-core platform in torus topology. As shown in this figure, each router is connected to a core and four neighbor routers by short local interconnections. Therefore, for a torus topology on-chip network, each router has five input/output channels, transmitting data from/to north, south, west, east, and central nodes.

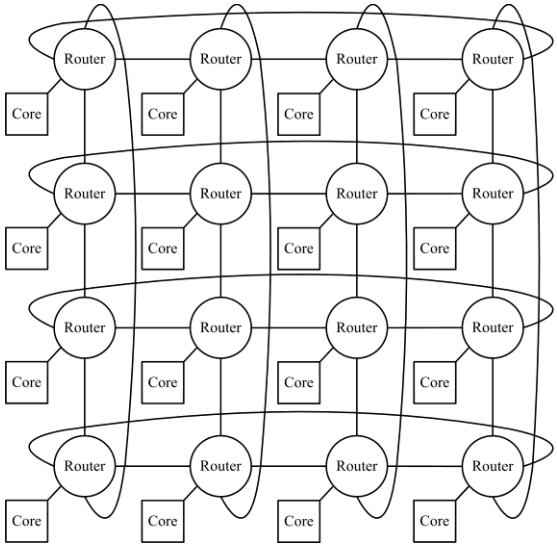


Fig. 2-2 4x4 multi-core platform in torus topology

Different types of routers have their unique data flow control mechanisms. For instance, the virtual channel router achieves an efficient physical link utilization and high throughput. As shown in Fig. 2-3 [9], the input buffer of each input port consists of multiple virtual channels. These virtual channels share the same physical links of one input channel. When a data packet holding one virtual channel gets blocked due to some unknown situations, the rest of the data packets can be transmitted using the physical links through other virtual channels. The virtual channel router follows a "push-model" style [34] where data packets are pushed from a source to a destination with one hop or multiple hops. At each hop, when the header flit of a data packet is available at the router input channel, an acknowledgement signal is set "high" if virtual channels have space to store the incoming data packet. Upon receiving this acknowledgement signal, the input buffer will receive a *write* request to initiate a *write* operation. When all virtual channels are occupied and have no more space to store new data packets, an acknowledgement signal is set to "low" and the *write* operation stops. The *router computation* block uses deterministic logic to analyze the header flit of data packets, computes the destination of data packets and selects output channels. If an output channel is available, and if the grant signals are set to "high" by the virtual channel allocator and switch allocator, the input buffer will receive *read* requests and initiate the *read* operations until the grant signals are set to "low" or the virtual channel becomes empty. A typical router pipeline includes buffer write and route computation stage (BW&RC), virtual channel allocation and switch allocation stage (VA&SA), and switch traversal stage (ST).

Each pipeline stage is sequential. Let t_p be the packet latency which is the duration for one data packet traverses through the router pipeline. The average router delay t_r can be calculated as:

$$t_r = H \cdot N \cdot t_p \tag{2.2}$$

where N is the total number of data packets and H is the hop count.

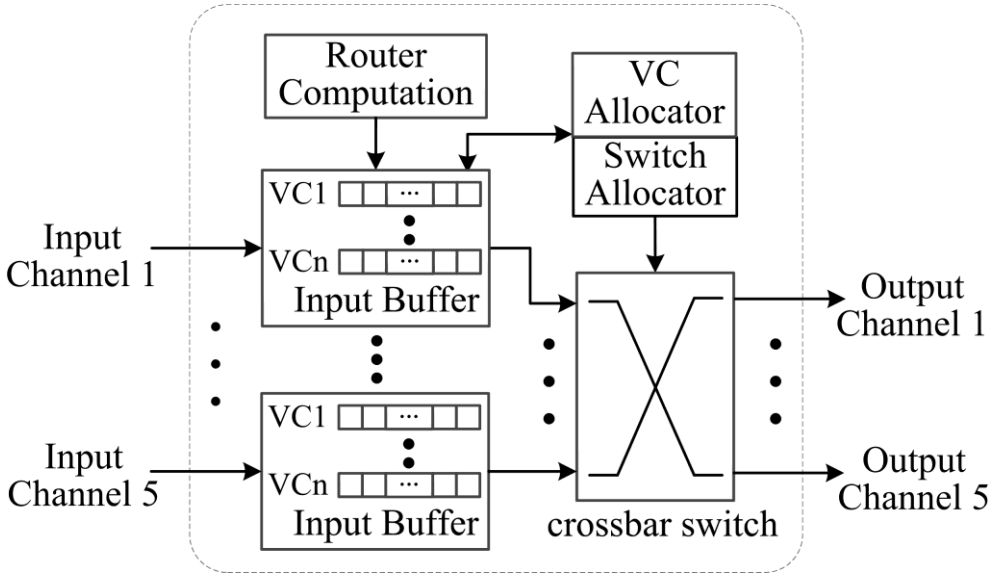


Fig. 2-3 Conventional virtual channel router architecture [9]

In summary, NoC routers put transmission data in queues to *write* into or *read* out from router input buffers. Each *write* or *read* operation for one data flit will update the router buffer in a sequential way while other data flits in the same packet are waiting. The same sequential operations also exist in the crossbar. Switch allocator coordinates the usage of input channels and output channels. Only one input channel can transmit data to one output channel each time. Data

packets from other input channels have to wait and try to bid for switch and output channels in the future cycles. As shown in Fig. 2-4, *On-chip Communication A* is sent from the central core to the east node, *On-chip Communication B* is sent from the south node to the north node. Even they use different input and output channels, they cannot be transmitted simultaneously. The router delay is literally the sequential addition for the two inter-core communications. Therefore, even each core in a multi-core system performs the same; the system throughput may still go down due to many communications among cores. Researchers realized that the only way to exploit multiple cores is through parallelism [35-39].

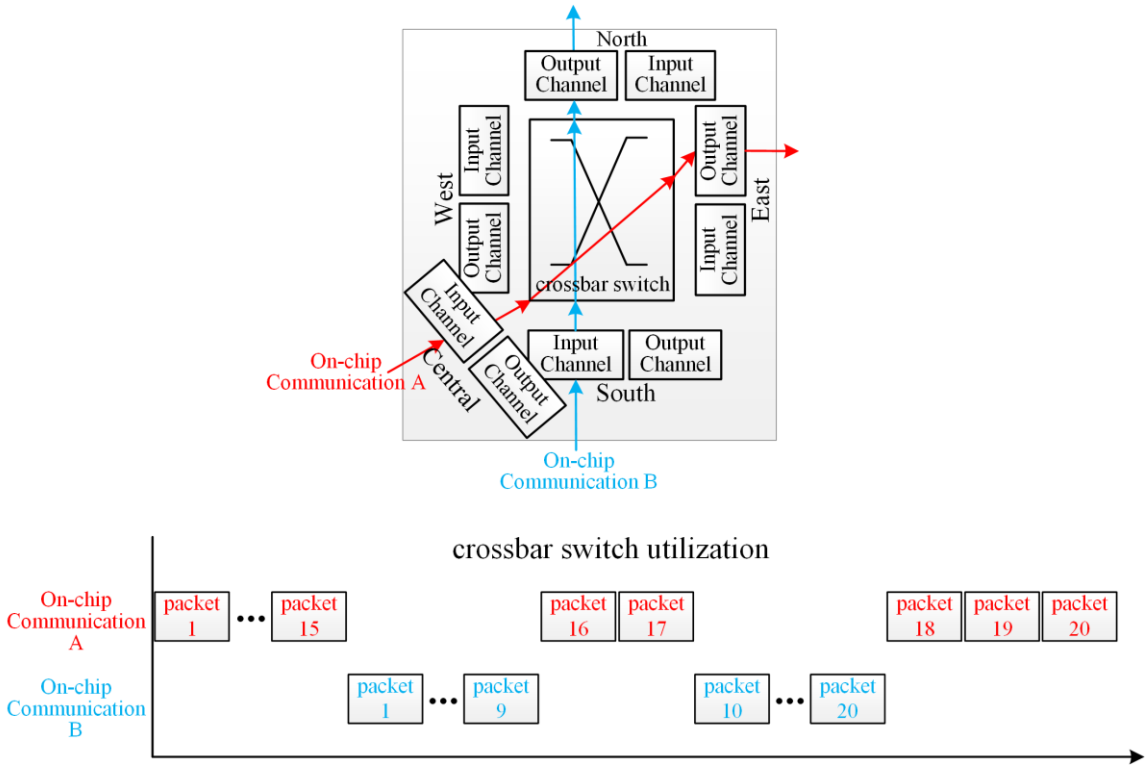


Fig. 2-4 Two on-chip communication bidding for switch and output channels

Moreover, the conventional router follows a "push-model" style [34], where data packets are pushed from the router input channel into the router output channel without investigating the data contents. In reality, with an application (e.g. FFT in image processing applications), it is possible to know data content through tasks and task flows. The existing router design cannot take advantage of the knowledge of task flows and their associated data requests. The routing speed is thus limited by the conventional "push-model".

To solve the above problems, this chapter introduces a virtual collision array based NoC router microarchitecture design. This new design integrates a virtual collision array structure at the input of router buffers to eliminate the sequential data access to the router pipeline. Instead of direct accesses to the router input buffers, data requests meet in the virtual collision array to cancel their opposite requests. The collision array concept was first introduced in [40] [41] for a single core with multiple threads. We will discussion this concept in the next section. For the first time, we introduce how to use this concept to redesign router for the NoC architecture.

2.2 Collision Array Concept for Common Sequential Components

Collision array was first introduced in [40] [41] to increase parallelism and reduce contentions when visiting common sequential components in single core multi-thread architectures. In this section, we first introduce this collision array data structure and then explain how we can apply the same concept in the NoC architecture of multi-core system.

Fig. 2-5 [41] presents a collision array structure for a common stack. There are two operations associated with a stack: *pop* and *push*. In a single core multi-thread architecture, each thread may try to gain access to this common stack. Such an attempt is counted as one request. If this request fails to be exercised because of the contention from other requests from other threads, then it backs-off into a structure called collision array. Here, a random location in the array is picked. If there is another request already stored in this location, requests may be eliminated if they have opposite operations. For example, one request wants to *pop* the data. The other request wishes to *push* the data. Then *push* exchanges its data with *pop*. This procedure is called request elimination. In this case, the stack is not really visited or accessed. If one request cannot exchange data with the request stored in the location picked (for example, *pop* cannot collide with *pop*, *push* cannot collide with *push*) or it does not find another request there, then this request would revisits the stack [40] [41]. For single core multi-thread architecture, this collision array structure can achieve a significant throughput improvement, compared with the "Back-off Only" algorithm. NoC based multi-core system also has shared structures, such as input buffers and crossbar in routers. Amdahl's law is also applicable to these cases. The collision array concept is not only limited for the single core multi-thread architecture. We extend it to the on-chip router to allow scalability of this type of architecture. Note that for the collision array concept in the previous researches, the collision array size or the collision array request holding/back-off duration (these definitions will be provided in the next section) is never considered as potential influential factor to the overall router delay. In addition, the data access

and operation requests are originally determined by the application itself. It is these tasks from one application that decide the need of data access and operation. Therefore, the “check-in” and “check-out” of collision array are not random, but fundamentally decided by task flows.

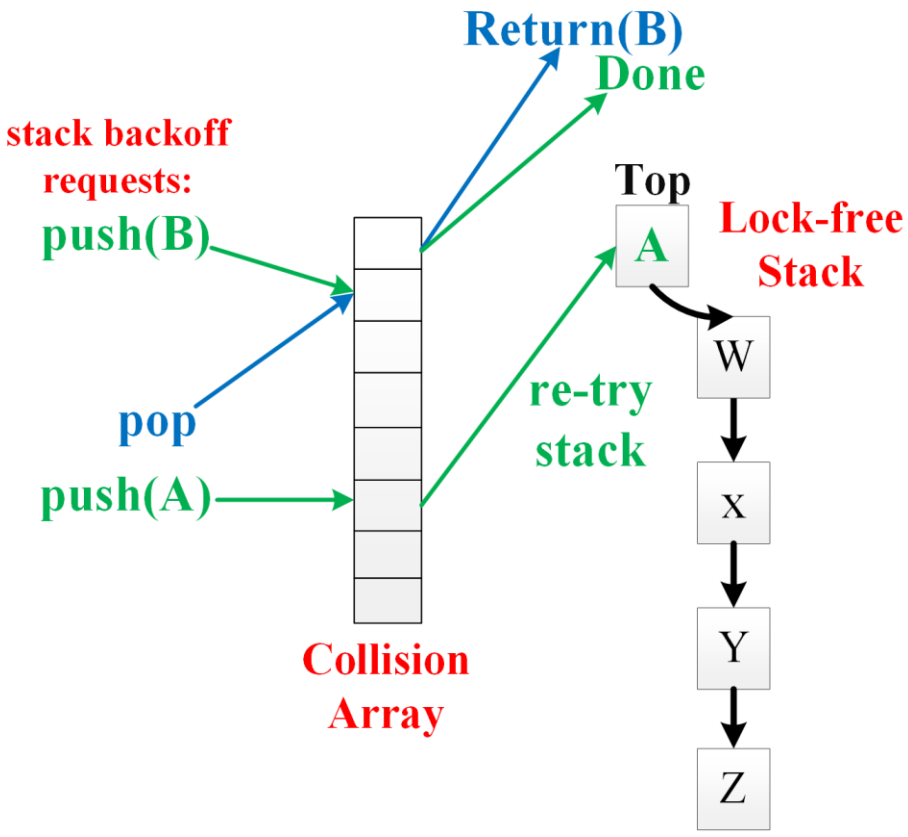


Fig. 2-5 Collision array structure for a common stack [41]

2.3 Virtual Collision Array for NoC Router

2.3.1 Communication Request Elimination in Virtual Collision Array

In this chapter, we propose a new NoC router structure. A new circuit "virtual collision array" is implemented in each input port of the conventional virtual channel router. This is to eliminate sequential data access to the router pipeline as well as to improve parallelism of the router resource utilization. To illustrate, instead of direct access to the input buffers, data requests first visit the virtual collision array to potentially collide with other requests. We provide the following two definitions for virtual collision array to help elaborate the models related to this new design.

Definition 2: ***Virtual Collision Array*** *is a virtual data array (also referred to as elimination array) where data requests may be eliminated before they access the shared structure.*

Definition 3: ***Collision Array Size (L)*** *is the length of collision array. It indicates the maximum number of write requests that can be stored in the collision array at the same time.*

Before we discuss how to design the new virtual collision array, let us first talk about how this array works. NoC platforms are driven by real applications. It is well known that application provides task flow. Different applications have different task flow topologies and data volumes. Each task flow has a number of tasks. It has been shown in [42] that tasks belonging to different task flows have little dependency, but tasks belonging to the same task flow are highly related. Fig. 2-6 illustrates a real application task flow example represented as a Directed Acyclic Graph (DAG). $T = \{T_1, T_2, \ldots, T_{10}\}$ is a set of nodes with each node representing one task. We also

assume that these tasks are fine grain type. Task weight w_i beside each node represents the execution time of task T_i on a single core. The router delay is directly impacted by data and its volume. We use data tokens [43], the number on each arc between two nodes in Fig. 2-6, to represent the number of data packets transmitted between two tasks. A task flow needs to be partitioned and assigned to different cores before it can be executed on a multi-core platform. When we process task flow on a multi-core platform, two directly connected tasks T_i and T_j may be partitioned and assigned to two different cores *A* and *B*. Then the data packets between T_i and T_j need to be transmitted from *Core A* to *Core B* through on-chip routers. The data delivery between two directly connected tasks executed by different cores is referred to as *Inter-core Communication*.

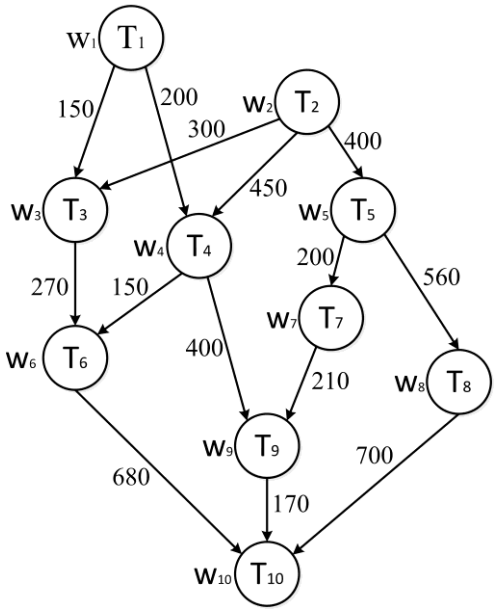


Fig. 2-6 Real application task flow example

Definition 1: ***Inter-core Communication*** *refers to data delivery between two directly connected tasks executed by different cores.*

Given an application, we have task flows and their associated tasks. As shown as an example in Fig. 2-7, when we process task flow on a multi-core platform, tasks T_1 and T_2 are assigned to *Core A*, tasks T_3 and T_4 are assigned to *Core B*. Then the data packets between T_1 and T_3 need to be transmitted from *Core A* to *Core B* through on-chip routers. Note that task flow partitioning and task scheduling are performed ahead of the routing process. Through task flow partitioning, we can acquire the source and destination information of inter-core communication data packets. Through task scheduling, we can increase the chance of sequential data access elimination. We will discuss the workload assignment and task scheduling in the next section. The virtual collision array is placed before router virtual channels in each router input port. The new design has a random address generator, a counter, a state analyzer and an L-size virtual collision array. During task flow partitioning and task scheduling stage, tasks are assigned to different cores and inter-core communications are generated. Fig. 2-7 demonstrates the process of request elimination in the virtual collision array. We will discuss the *write* and *read* request generations after elaborating how the new virtual collision array works. When a *write* request arrives router input port, the random address generator provides one random address for this data request in the L-size virtual collision array. If the associated array location of this random address is available, the data request is stored in this location. If this location is occupied by another data request, the random address generator will provide a different address for this new

data request. For example, *write*(*i*) request in Fig. 2-7 is first assigned to virtual collision array *location*(*1*). However, this location has already been occupied by *write*(*b*) request. So *write*(*i*) request gets a new address from the random address generator and is stored there. When a *read* request arrives, the random address generator also provides a random address in the same virtual collision array. The state analyzer records virtual collision array status. If there is a *write* request associated with the same data, these two opposite requests collide. Both requests are thus eliminated. The associated data packet will bypass the router pipeline and directly reach output channel for its destination. For example, as shown in Fig. 2-7, *read*(*c*) meets with *write*(*c*) in virtual collision array *location*(*0*). These two requests are eliminated and *packet*(*c*) will no longer traverse through the router pipeline, but be directly sent from input channel to output channel. If the *read* request cannot meet with its correspondent opposite request, it backs off and waits t_b cycles for the next available access to virtual collision array, while its corresponding data packet is still stored in the memory of the core or buffer of the previous router (*read*(*d*) request in Fig. 2-7). We would like to point out that each data request has independent access to the virtual collision array. Therefore, an effective way to increase data collision possibility and reduce router delay is by raising the number of accesses. That is, we allow back-offs and revisits to this virtual collision array. It is worth declaring here that back-offs do not degenerate the network performance. Without back-offs, data packet is sent to the conventional router pipeline when just one collision does not happen. However, the conventional router pipeline is a totally sequential structure. Data packets will be accumulated in the input queue, still without be transmitted. To

the contrary, back-offs allow requests revisit the virtual collision array, increase request eliminations. In addition, back-offs are not infinite. Each *write* request is stored in a random location of virtual collision array for a certain duration t_h. When the counter records t_h cycles, if the data request is still not eliminated, it leaves virtual collision array and returns to input buffers. The associated data packet is going to traverse the router pipeline without bypass. For example, *write*(*a*) in Fig. 2-7 is un-eliminated after storing in virtual collision array for t_h cycles. *packet*(*a*) will be sent to the router input buffer. The proposed virtual collision array works in the packet level. All data flits in a same packet will follow the same transmission path.

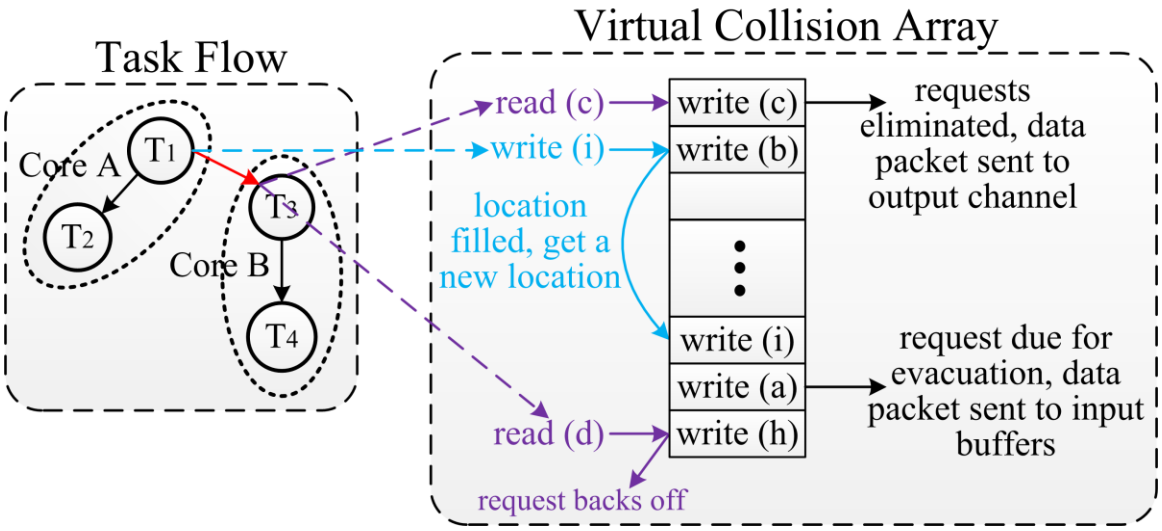


Fig. 2-7 Request elimination in virtual collision array

Definition 4: ***Read Request Back-off Duration*** **(t_b)** *is the time interval that a read request has to wait before it revisits the virtual collision array.*

Definition 5: ***Write Request Holding Duration (t_h)*** *is the longest cycles that the virtual collision array can store each write request. Once the holding duration expires, write requests that are not eliminated have to leave the virtual collision array. And the associated data packet will traverse the router pipeline without bypass.*

2.3.2 Virtual Collision Array based NoC Router Structure

Now let us discuss how the *write* and *read* requests are generated in the new architecture. Fig. 2-8 depicts the new router design with the proposed virtual collision array and its control circuits. When an inter-core communication needs to be transmitted by a router, data packets are injected into one of router input ports from the memory of a core or buffer of the previous router. If the virtual collision array in this input port has space to store data requests, an acknowledgement signal is set to "high" by the state analyzer and is sent to the input channel. For each data packet, a *write* request is issued and is sent to the virtual collision array. At the same time, routing computation extracts the destination address of a data packet in the header flit and decides the output channel for the each data packet. If the output channel is available for receiving data, it sends an acknowledge signal back to input port, and a *read* request is issued. This *read* request will visit the virtual collision array and will try to collide with its *write* request already waiting there. If request elimination occurs, the data packet will bypass the router pipeline and reach the output channel. As mentioned above, the new virtual collision array eliminates data requests that are sent to the router pipeline. Thus, data packets can bypass the

router pipeline to save routing time and energy. It is worth emphasizing that the *write* and *read* requests are generated and analyzed inside one router. For each data packet, there is one unique pair of *write* and *read* requests. Therefore, there is no confusion or confliction in the request elimination.

Our proposed structure allows data requests to visit the virtual collision array simultaneously, therefore it introduces parallelism to the sequential router structure. However, we also need to ensure there is no deadlock in the case that multiple requests are trying to update the virtual collision array at the same time. We implement the virtual collision array as a lock-based structure. The lock is realized by a special synchronization operation called CAS in the state analyzer. CAS is widely used in today's multi-core systems [41]. When a *read* request arrives at random location of the virtual collision array, CAS compares the address information of this *read* request and the *write* request stored in the location. If the two requests are associated with the same data packet, CAS outputs a Boolean signal to lock the virtual collision array and only lets this pair of requests update the virtual collision array. With the CAS lock, although multiple requests can visit the virtual collision array simultaneously, only one of them can update the virtual collision array each time. Therefore our proposed virtual collision array is deadlock-free.

Elimination rate, router delay, and power/energy consumption are critical performance metrics for the virtual collision array, and will be discussed in details in the following sections.

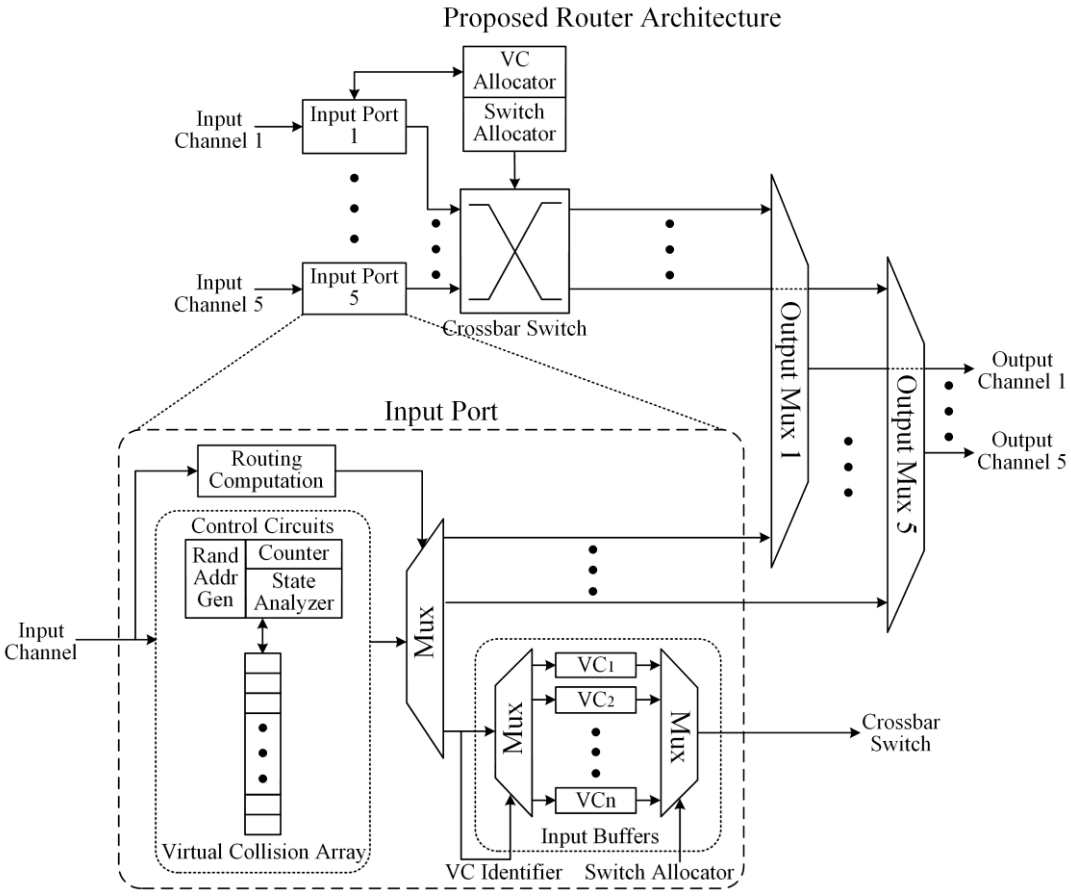


Fig. 2-8 Virtual collision array based router structure

2.3.3 Virtual Collision Array Elimination Rate

The introduction of our virtual collision array for NoC router is to eliminate data requests before them entering the conventional router pipeline. Therefore, the elimination rate of virtual collision array becomes a critical parameter for evaluating the performance. For the *L*-size virtual collision array, without loss of generality, let us assume that some *write* requests are already stored inside. Each *read* request can visit any location with a possibility of *1/L*. Thus the

probability of successful elimination is *1/L*. And the probability of elimination failure is 1-*1/L*. Let t_b be the *read* request back-off duration when a failure of elimination happens. And t_h denotes the *write* request holding duration. The ratio of t_h/t_b indicates the upper bound to the number of times that one *read* request may visit the collision array. The elimination rate can be express as:

$$ER = 1 - (\frac{L-1}{L})^{t_h/t_b} \tag{2.3}$$

As described by (2.3), the elimination rate *ER* is proportional to the ratio of t_h/t_b, and is reversely proportional to the increase of *L*. To verify the proposed virtual collision array model, we implemented it in C++. So, performances of the virtual collision array can be simulated. Fig. 2-9 verifies the elimination rate changing trend with different holding/back-off duration ratios for various virtual collision array sizes. It can be observed that larger elimination rate is achieved by using higher t_h/t_b ratio and smaller *L*. Let *N* be the total number of data packets. The number of data packets that are associated with failed elimination attempts in the collision array is:

$$N_{fail} = N \cdot (\frac{L-1}{L})^{t_h t_b} \tag{2.4}$$

And the number of data packets that will bypass router pipeline is:

$$N_{succ} = N \cdot (1 - (\frac{L-1}{L})^{t_h t_b}) \tag{2.5}$$

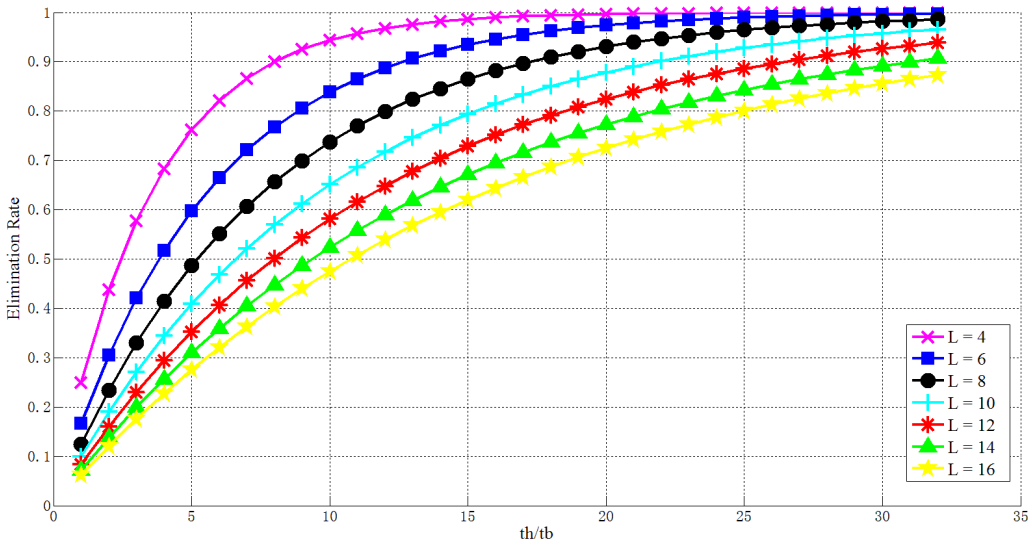


Fig. 2-9 Elimination rate vs t_h/t_b

Despite elimination rate is a critical performance metric for the virtual collision array, high elimination rate does not necessary lead to the low router delay or low router energy consumption. In the following sections, we will discuss further how to design virtual collision array to optimize router delay and router energy consumption.

2.3.4 Virtual Collision Array based Router Delay

(2.2) shows the router delay of an N-packet inter-core communication traveling through H hops. (2.4) provides the estimation of the number of data packets that will use the router pipeline. Therefore, the total router pipeline delay for this inter-core communication can be calculated as the product of number of data packet using router pipeline and router pipeline delay:

$$t_{pipeline} = N_{fail} \cdot t_p \tag{2.6}$$

Let k_i be the number of visit counts for the *i*-th *read* request. This k_i is bounded by $1 \le k_i \le t_h / t_b$. The delay to eliminate this *i*-th *read* request and its correspondent *write* request in virtual collision array is therefore $k_i \cdot t_b$. The total delay needed to successfully eliminate requests in virtual collision array can be calculated as:

$$t_{succ\_eli} = \sum_{i=1}^{N_{succ}} k_i \cdot t_b \tag{2.7}$$

where N_{succ} is estimated in (2.5). The total delay spent in virtual collision array for failed eliminated requests can be calculated as:

$$t_{fail\_eli} = N_{fail} \cdot t_h \tag{2.8}$$

Therefore, the total delay at the virtual collision array can be derived as:

$$t_{array} = \sum_{i=1}^{N_{succ}} k_i \cdot t_b + N_{fail} \cdot t_h + N_{succ} \cdot t_w \tag{2.9}$$

where t_w is the wire delay for one data packet traveling from the router input channel to the output channel. As described from (2.7) to (2.9), delay of the virtual collision array is proportional to *L*, and is reverse proportional to the ratio of t_h/t_b. This can be explained as follows. If one *write* request is stored in virtual collision array for long time, it deprives other requests' colliding opportunities. In addition, higher t_h/t_b ratio implies that each *write* request can be stored

in virtual collision array for a longer time. The delay of virtual collision array is thus increased. On the other hand, bigger L provides more space in the virtual collision array. The delay of the virtual collision array is reduced with bigger L since more requests can be processed in the virtual collision array. Fig. 2-10 verifies the relationships among the delay of virtual collision array, the size of virtual collision array, and the holding/back-off duration ratio t_h/t_b. We assume that the inter-core communication has 5000 data packets. Before L reaches 8, the virtual collision array delay drops dramatically when L increases. When L reaches 8, this dropping trend slows down, and begins to saturate when L exceeds 10. When the t_h/t_b ratio reaches 8, the virtual collision array delay is 33K cycles at $L=2$. The delay becomes 13K cycles at $L=16$. With different applications, these numbers would be different, but they follow that same theoretical explanation. The total router delay of the new architecture can be calculated as:

$$t'_r = H \cdot (t_{pipeline} + t_{array} - t_{overlap}) \tag{2.10}$$

where $t_{overlap}$ is the overlap of router pipeline delay and virtual collision array delay. (2.6) and (2.9) show that $t_{pipeline}$ and t_{array} are inverse proportional to each other. Fig. 2-11 demonstrates the relationship of $t_{pipeline}$ and t_{array} with the increase of holding/back-off duration ratio. Again, we assume that the inter-core communication has 5000 data packets with a packet size of four 64-bit flits. We simulated this example on an on-chip network of 4x4 torus topology with 16 virtual channels per physical channel, 8 flits per virtual channel. Fig. 2-11 shows that along with the increase of t_h/t_b ratio, the router pipeline delay decreases and collision array delay increases. We can explain the observations as follows. The virtual collision array is placed before the router

virtual channels. When a data packet is processed by the virtual collision array, a previous data packet is processed by the existing router pipeline at the same time. Therefore, the virtual collision array delay and the router pipeline delay are highly overlapped. According to evaluation, the overlap reaches 95% during the whole routing process. Therefore, for optimizing the total router delay, we need carefully design the virtual collision array properties to balance the virtual collision array delay and router pipeline delay.

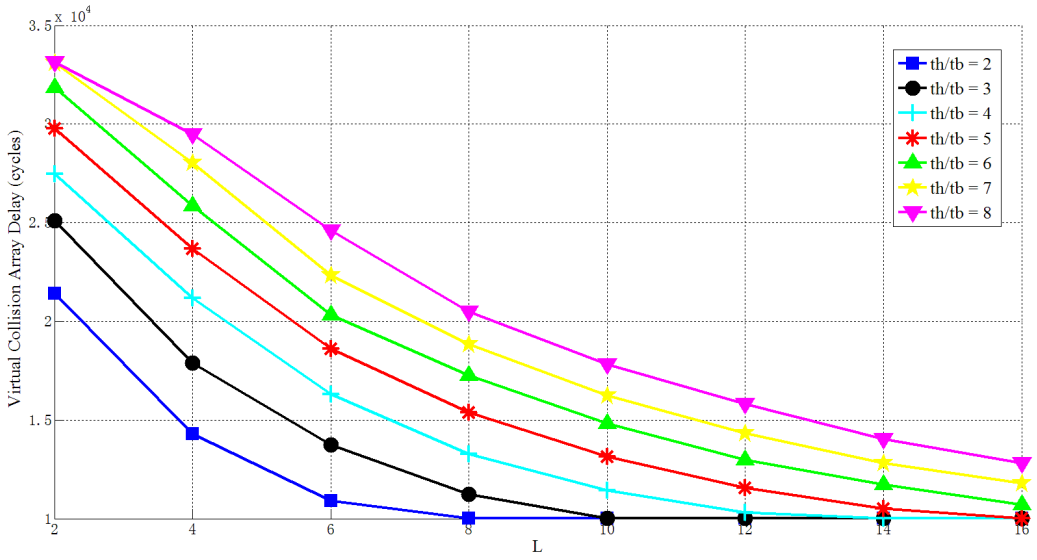


Fig. 2-10 Virtual collision array delay vs *L*

2.3.5 Virtual Collision Array based NoC Platform Power and Energy

We would like to point out that the proposed collision array is a virtual data structure. Data packets are never truly stored or passed through the virtual collision array. However, the new

design introduces more hardware, for example additional multiplexers and control circuits of the virtual collision array, which can introduce extra power consumption to the NoC router. On the other hand, due to the request elimination in the virtual collision array, the number of data packets transmitted by router pipeline is reduced. Therefore, the energy consumed on router pipeline is saved. For the conventional router structure, let P_{unit} be the original power consumption of one virtual channel router. It includes the power consumed by the input buffer, crossbar switch, allocators, arbiters, and wired links. The average power consumption of one router during the whole routing process can be calculated as:

$$P_r = \frac{P_{unit} \cdot t_p \cdot N}{t_r} \tag{2.11}$$

where t_p is the packet latency, and N is the number of data packets processed by this router. t_r is the total router delay for transmitting these N data packets. The average power consumption of the whole NoC platform can be calculated as:

$$P_{NoC} = \sum_{i=1}^{h} P_{ri} = \sum_{i=1}^{h} \frac{P_{unit} \cdot t_p \cdot N_i}{t_r} \tag{2.12}$$

where h is the number of routers, N_i the number of data packets processed on the i-th router. The total energy consumed on the NoC platform can be calculated as:

$$E_{NoC} = P_{NoC} \cdot t_r = \sum_{i=1}^{h} P_{unit} \cdot t_p \cdot N_i \tag{2.13}$$

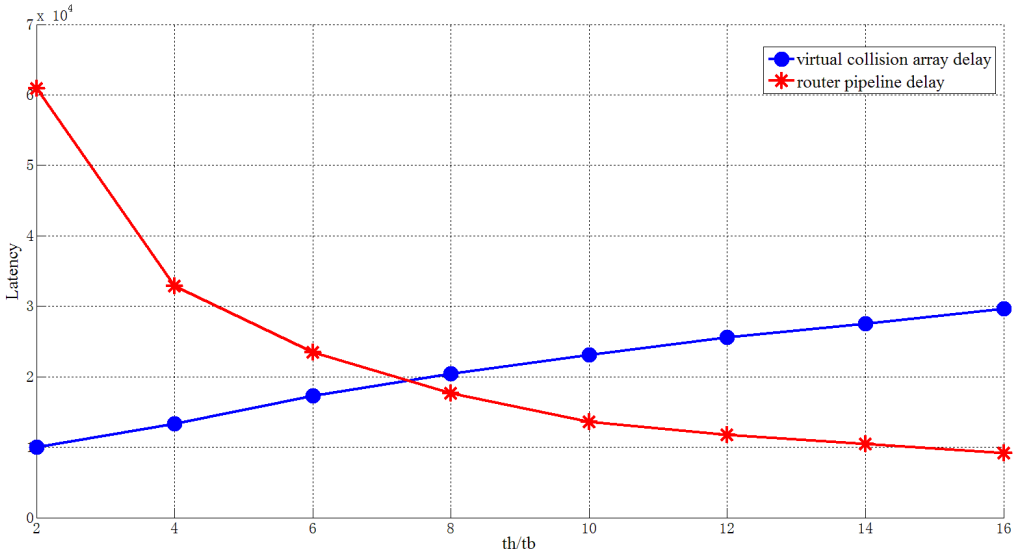


Fig. 2-11 t_{array} and $t_{pipeline}$ vs t_h/t_b

For the new virtual collision array based router structure, let P_{array} be the power consumption of the virtual collision array. It includes the power consumed by the random address generator, counter, and state analyzer in the control circuits and the additional multiplexers. The average power consumption of the whole NoC platform can be calculated as:

$$P'_{NoC} = \sum_{i=1}^{h} P'_{ri} = \sum_{i=1}^{h} \frac{(P_{unit} + P_{array}) \cdot t'_{ri}}{t'_r} \tag{2.14}$$

where t'_r is the total router delay of the new structure and t'_{ri} is the individual delay of the i-th router. The total energy consumed on the platform can be calculated as:

$$E'_{NoC} = \sum_{i=1}^{h} (P_{unit} + P_{array}) \cdot t'_r \cdot \alpha_i \tag{2.15}$$

where α_i is the duty rate of the i-th router. t'_{ri} and α_i are determined by the workload assignment algorithm and task scheduling algorithm. We will discuss these algorithms in the following section.

Due to the extra hardware added into the virtual channel router, the average power consumption of the proposed NoC platform is larger than which of the conventional NoC platform ($P'_{NoC} > P_{NoC}$). However, the new technology reduces router delay. The total energy consumed on the NoC platform is saved ($E'_{NoC} < E_{NoC}$).

2.4 Workload Assignment and Task Scheduling

As shown in (2.6)-(2.10), the total router delay of the proposed virtual collision array based router is a function of data packets number: $t'_r(N)$. Task flow partitioning determines the number of data packets transmitted by routers among the whole network. The proposed task flow partitioning algorithm aims to minimize the total router delay by assigning workloads to multiple cores with the minimum data communication. Capacity Rate is introduced as an indication of how much workload one core can sustain during workload assignment. Workloads are fundamentally composed of sets of tasks, named task flows. Tasks from different flows tend to have very few or no dependencies among them. Therefore, to reduce total routing delay, we limit the process of one task flow to a group of cores physically adjacent to each other. Such a group of cores is defined as a partition. The partitioning algorithm aims at mapping a task flow onto

one particular partition. Task scheduling will accordingly be done within the partition that is designated to this task flow.

Generally a task flow is represented by a DAG $G = (T, E)$ with T denoting a set of tasks to be executed and $E = \{(i, j)\}$ specifying precedence relationships among tasks. Each task T_i is associated with a task weight w_i representing its execution time. The workload of this task flow can be evaluated by the structure of the task graph and the task weights [44]:

$$Workload = \frac{\sum_{i} w_i \forall T_i \in T}{\sum_{j} w_j \forall T_j \in WCP} \tag{2.16}$$

where WCP is the group of tasks in the worst-case path. When performing partitioning, the partition should be able to accept the workload evaluated based on the given flow. On the other hand, as capacity rate specifies the upper bound of workload on each core, the sum of core capacity rates in one zone reflects the maximum total workload one zone can accept. This summation must exceed the estimated workload resulted from the flow assigned to this zone. In addition, for the whole multi-core system, there exists a power consumption boundary P_{up_total}. The summation of power consumption P for each core should be less than P_{up_total}. Taking all these constraints into consideration, the partitioning problem for a particular task flow can be described as:

$$minimize \quad \sum_{i,j} t'_r(N) \quad \forall T_i, T_j \in Z_k$$

$$subject\ to \quad \sum CR \geq Workload$$

$$\sum P \le P_{up\_total}$$

$$variables \quad Z_k$$

where T_i and T_j are two directly connected tasks assigned to different cores. Z_k is the partitioning result consisting of an optimal set of adjacent cores, such that the router delay is minimized. The search for an optimal partition was done following a greedy heuristic. As a prerequisite, we rank the cores according to the capacity rates, and identify the strongest 20% of the cores (with highest capacity rates). The partitioning algorithm starts with an identified strong core to initialize a partition in rectangular shape, and calculates the total router delay within this partition. The next step is to make slight adjustments to this initialized zone for exploring a better partition, following a heuristic procedure. For each adjustment, the heuristic compares the router delay with the delay for the initialized partition. The heuristic accepts changes that improve router delay following a simulated annealing algorithm. This searching procedure is repeated until the termination condition is satisfied. All tasks from the task flow will be assigned to the cores included in the obtained partition.

After partitioning, the next step is task scheduling within the partition designated to a task flow. The goal is determine the task-core mapping results to achieve the minimum routing delay. We formulate the task scheduling problem as a mixed-integer program (MIP), by introducing a set of binary variables $\{M_{ij}$'s$\}$ to represent all task-core mapping results. Each variable M_{ij} is a decision variable of binary type, which is set to "1" if task T_i is mapped onto the j-th core and "0" as the opposite situation. Suppose a flow of n tasks is to be executed within a partition of m

cores. The total number of decision variables is thus (*n* x *m)*. Apparently for each task T_i the definition of this binary variable imposes the following constraint:

$$\sum_{j=1}^{m} M_{i,j} = 1 \tag{2.17}$$

which indicates that one task can be assigned to only one core. Using these binary variables, task allocation can be conveniently determined.

In addition to the binary mapping variables, another set of decision variables are the starting times for all tasks, denoted by d^s. Task starting times are combined with the task-core mapping variables to determine the job sequences of all involved cores. An optimal schedule can be fully specified by determining the binary variables M_{ij}'s and task starting times d^s: the decision variables related to *j*-th core indicates a set of tasks to be executed on it, and the starting times help determine the execution sequence of all assigned tasks. This introduces the second set of constraints in task scheduling, which is for the purpose of keeping the precedence relationships among tasks. For each arc (*i, j*) on the task graph, the precedence relationship requires that task T_i must be finished before the execution of its successor T_j:

$$d_i^s + d_i^e + d_{i,j}^{comm} \le d_j^s \tag{2.18}$$

where d_i^s and d_i^e represent the starting time and execution time for task T_i , respectively, while $d_{i,j}^{comm}$ represents the communication time between task T_i and T_j, which is determined by the router delay.

There are two other constraints for each core in the scheduling framework: workload constraint imposed by core capacity rate and upper bound limit of power consumption. Capacity rate is included as an upper bound limit for workload assigned to each core. A core with low capacity rate indicates that light workload should be assigned to this core. The workload on a core depends on task allocation and frequency scaling ratio. The estimation of core workload can be detailed in [44]. When performing task scheduling the real workload assigned to a core cannot exceed a pre-evaluated bounding value provided by capacity rate; otherwise, it may break the performance limits and cause reliability issues.

The last concern in this scheduling framework is the optimization objective. For the purpose of optimizing total routing delay, we need to minimize the number of hops for each inter-core communication. Therefore, we should have the shortest Manhattan Distance among the cores with tasks mapping on, in order to achieve minimum routing delay. To sum up, task scheduling in this proposed router design can be generalized by the following mixed-integer program:

$$\textit{minimize} \qquad \sum_{i,j} MD(i,j) \quad \forall T_i, T_j \in M_k$$

$$\textit{subject to} \qquad d_i^s + d_i^e + d_{i,j}^{comm} \le d_j^s$$

$$CR_k \ge Workload_k$$

$$P_k \le P_{upk}$$

$$\textit{variables} \qquad M_k, d^s = \{d_1^s, d_2^s, \ldots, \ d_n^s\}$$

where M_k denotes the set of mapping variables, which exert a constraint in this MIP to guarantee the uniqueness of task-core mapping relationship. d^s is the set of task starting time. *MD(i,j)* is the Manhattan Distance between two tasks T_i and T_j having precedence relationship, and the objective function is the summation of all *MD(i,j)*'s. The execution time of Task T_i must be completed before the execution of T_j, including the communication time. For each core processing a part of the task flow, its capacity rate CR_k must be larger than the workload of these tasks. Additionally, there is an upper boundary P_{upk} for each core's power consumption P_k.

It is worth emphasizing that although the optimal scheduling result can be obtained by solving the MIP described above, its computational complexity could be a serious issue when applied to real systems and applications. The size of the search space when solving MIP is proportional to the number of tasks and cores. To address this issue, the scheduling algorithm searches task-core mapping variables with simulated annealing to obtain a near optimal solution.

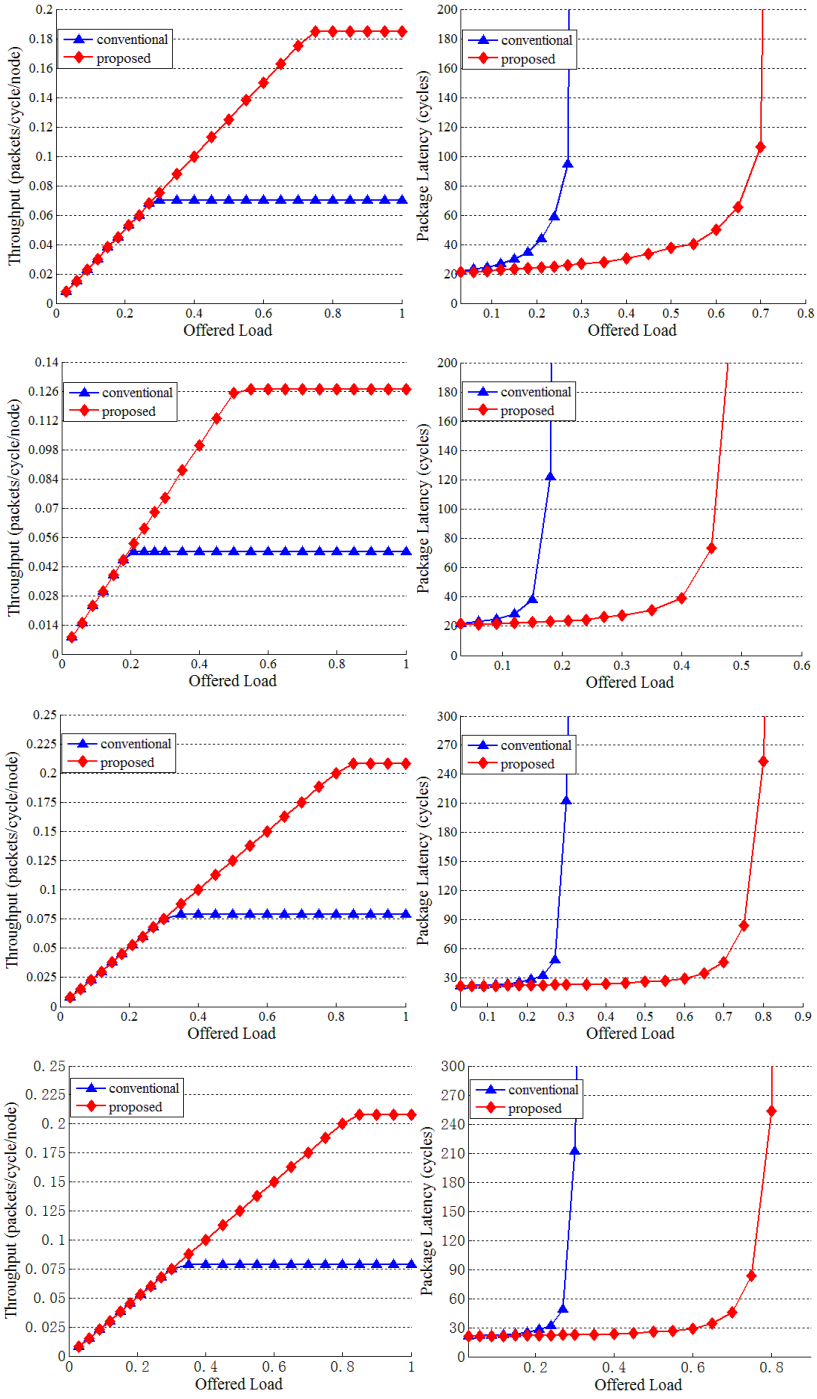
2.5 Result Analysis

In this section, we compare the proposed virtual collision array based router design with the existing virtual channel router design. The performance evaluation includes packet latency, system throughput, speedup, power, and energy consumption. A data packet size of four 64-bit flits is used in all experiments. We use Booksim2.0 [45], a cycle-accurate interconnection network simulator, to simulate an on-chip network of 4x4 torus topology with 16 virtual channels per physical channel, 8 flits per virtual channel. To keep the simplicity and ensure the

deadlock freedom, we choose dimension-ordered routing (DOR) as the routing algorithm. We use virtual-channel flow control to improve link utilization. To establish a fair comparison, we simulate different data traffic and generate different task graphs using real applications from four benchmark suites: MediaBench [46], MiBench [47], NetBench [48], and EEMBC [93]. Tasks of real embedded application and network application are generated by profiling particular benchmarks from benchmark suits. Then an application task flow is created by artificially assigning precedence relationships among the tasks. The profiling work was done by a previous group. Here we just use their task flow data in the experiments. The power and energy results for the on-chip network components are estimated by Xilinx ISE Design Suite 13.2 and Xilinx Power Analyzer with the Xilinx Virtex-6 library [49-51]. All the experiments are running on a computer with Intel Core i5-3230M CPU at 2.6GHz and Linux operating system.

For proving the adaptivity of the proposed virtual collision array router, we simulate eight different data traffic patterns: Uniform, Matrix Transpose, Neighbor, Tornado, Random Permutation, Shuffle, Bit Reversal, and Bit Complement [52]. Fig. 2-12 demonstrates the new structure's significant packet latency and system throughput improvement for the 4x4 torus network on all of the eight data traffics. Traffic 1 is Uniform traffic. Network throughput of the conventional virtual channel router saturates at 0.3 offered load. Network throughput of proposed virtual collision array based router saturates at 0.75 offered load. It shows an increase of 160% in throughput. Traffic 2 is Matrix Transpose traffic. Network throughput of the conventional virtual channel router saturates at 0.21 offered load. Network throughput of proposed virtual collision

array based router saturates at 0.55 offered load. It shows an increase of 159% in throughput. Traffic 3 is Neighbor traffic. Network throughput of the conventional virtual channel router saturates at 0.35 offered load. Network throughput of proposed virtual collision array based router saturates at 0.85 offered load. It shows an increase of 163% in throughput. Traffic 4 is Tornado traffic. Network throughput of the conventional virtual channel router saturates at 0.35 offered load. Network throughput of proposed virtual collision array based router saturates at 0.85 offered load. It shows an increase of 164% in throughput. Traffic 5 is Random Permutation traffic. Network throughput of the conventional virtual channel router saturates at 0.27 offered load. Network throughput of proposed virtual collision array based router saturates at 0.7 offered load. It shows an increase of 158% in throughput. Traffic 6 is Shuffle traffic. Network throughput of the conventional virtual channel router saturates at 0.24 offered load. Network throughput of proposed virtual collision array based router saturates at 0.6 offered load. It shows an increase of 166% in throughput. Traffic 7 is Bit Reversal traffic. Network throughput of the conventional virtual channel router saturates at 0.21 offered load. Network throughput of proposed virtual collision array based router saturates at 0.5 offered load. It shows an increase of 163% in throughput. Traffic 8 is Bit Reversal Complement. Network throughput of the conventional virtual channel router saturates at 0.35 offered load. Network throughput of proposed virtual collision array based router saturates at 0.85 offered load. It shows an increase of 163% in throughput.


conventional
proposed
Throughput (packets/cycle/node)
Package Latency (cycles)
Offered Load

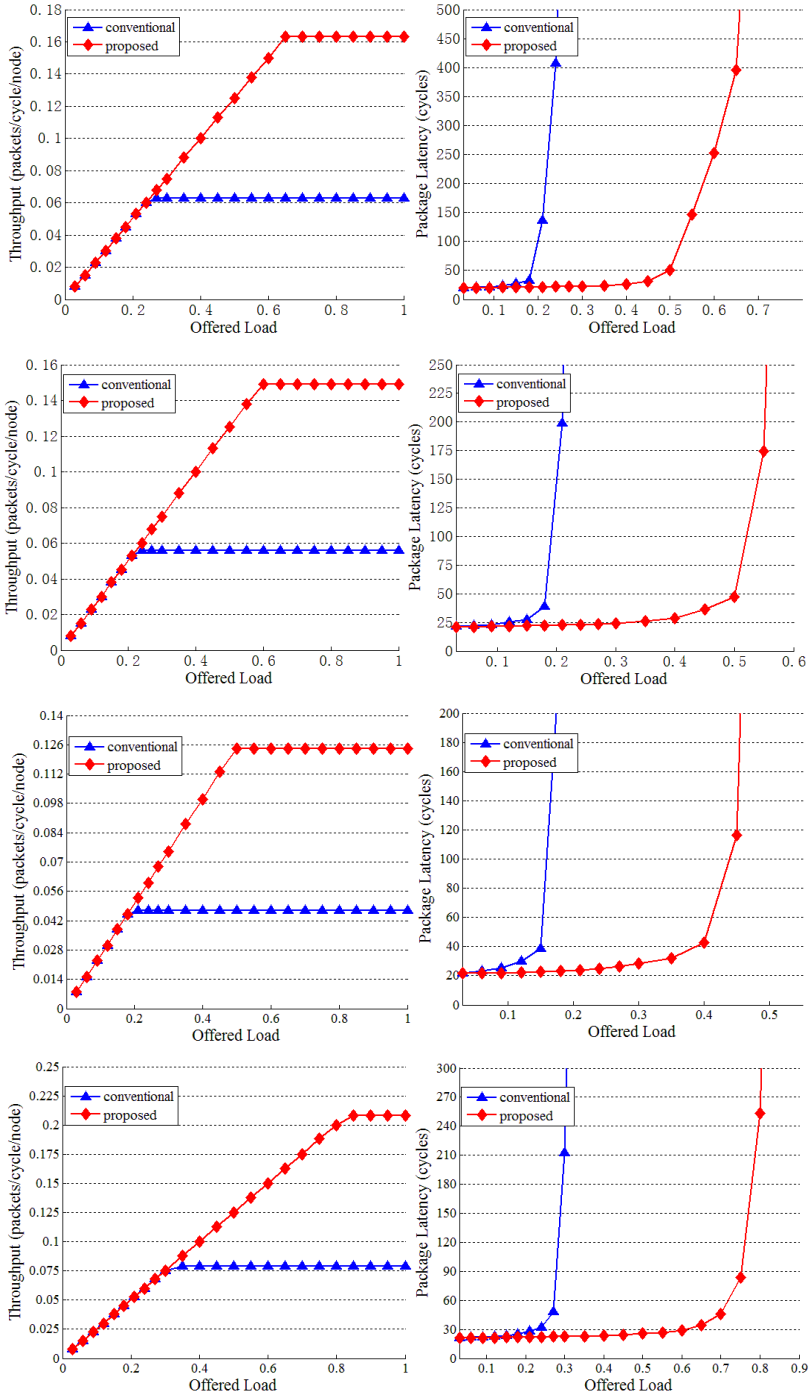


Fig. 2-12 Throughput and latency for different data traffics

The latency plots in Fig. 2-12 show the network saturates at the same point as the throughput plots for all of the eight traffic patterns. As long as the increase of offered load, the proposed virtual collision array based router has consistently lower packet latencies than the conventional virtual channel router. At the same time, packet latencies of the proposed structure saturate much slower than which of the conventional structure.

To evaluate the router delay deduction and measure the speedup of proposed technology, we generate different task graphs using real applications from four benchmark suites: MediaBench [46], MiBench [47], NetBench [48], and EEMBC [93]. Table II-1 lists fifteen real applications as well as design and performance parameters, with and without using the proposed technology. The first column lists the name of each real application. The second column provides the total number of tasks in each application task flow. The third column shows the number of cores in use after applying task flow partitioning algorithm. The fourth column provides the total number of data packets needs to be transmitted among the network. The fifth column shows the total router delay for the conventional virtual channel router. The sixth column is the designed virtual collision array size. The seventh column is the optimal holding duration over back-off duration ratio. The eighth column shows the elimination rate provided by the chosen virtual collision array size and holding duration over back-off duration ratio. The last column shows the total router delay by using the proposed technology. From Table II-1, it can be observed that by optimizing virtual collision array properties and applying workload assignment and task

scheduling algorithms, an average of 74.9% data transmission in the conventional router pipeline can be eliminated. Total router delay is significant saved by using the proposed technology.

Table II-1 Fifteen applications

Application Name	Number of Task Nodes	Number of Cores in Use	Number of Data Packets	Conv. Router Delay (cycles)	*L* Size	t_h/t_b	Elimination Rate	Prop. Router Delay (cycles)
A2TIME01	7	4	404	8897	8	11	71.5%	2458
AIFFTR01	14	7	640	14426	10	18	73.8%	3704
AIIFFT01	11	5	502	10979	8	15	76.7%	2878
bc	12	8	691	15610	10	17	76%	3624
CACHEB01	24	5	1094	25885	14	24	75.3%	5822
cjpeg	25	6	1140	27041	14	26	74.1%	6664
dhNetbench	25	4	1322	32720	16	28	75.8%	7176
djestra	25	4	1143	27513	14	26	75.9%	6332
djpeg	21	7	1058	25308	14	26	74.3%	6036
drrNetbench	21	7	1037	24702	14	25	75%	5746
epicUnoptimized Encode	14	6	680	15355	10	17	74.1%	3698
mpegDecode	18	3	926	21530	12	21	73.8%	5157
mpegEncode	23	5	982	23048	12	22	76.1%	5376
pegwitdecode	24	7	1163	28075	14	26	75.2%	6624
pegwitencode	20	4	973	22594	12	21	75.4%	5280

Fig. 2-13 shows the proposed technology speedup normalized by the conventional technology on the fifteen real applications. The highest speedup of 4.5 times is achieved on the *dhNetbench* application, which has the most task nodes number of 25 and most inter-core communication data packets of 1322. The lowest speedup of 3.6 times is achieved on the *A2TIME01* application, which has the least task nodes number of 7 and least inter-core communication data packets of 404. This is because the proposed virtual collision array eliminates sequential data request to the router pipeline and increase the system parallelism. The more complicated application with the more data packets transmitted among the network, the

more decent performance the proposed technology can provide. For average, a 4.18 times of speedup can be observed from the figure. The speedup is majorly achieved by the reduced number of data packets accessing router pipeline. It is also achieved by the reduced packet latency.

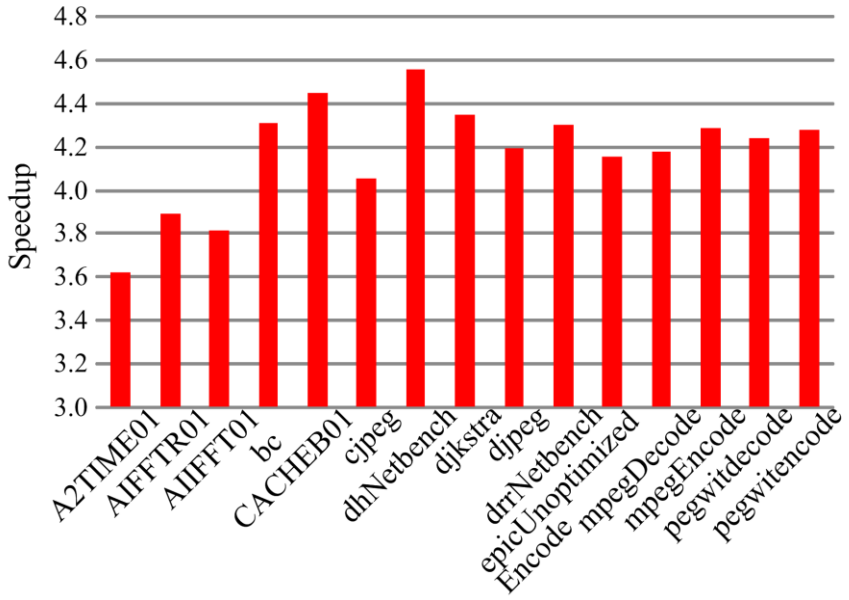


Fig. 2-13 Speedup on real applications

Fig. 2-14 shows the power consumption distribution of the proposed design using TSMC 40nm technology. It can be observed from this figure that the major power overhead is consumed by the extra link circuitry. This is because we have five more direct link connections between the virtual collision array and each of the five output channels. By contrast, power overhead

consumed in the virtual collision array control circuits and multiplexer is much less. On one hand, the extra hardware of the proposed virtual collision array based router architecture brings power overheads upon the conventional router. On the other hand, the router pipeline bypass of the proposed design saves partial power consumption over the conventional router. In summary, the average router power consumption of the new on-chip network is 45mW. Compare with the conventional virtual channel router structure (35mW power consumption), the proposed design has a power overhead of 28.6%.

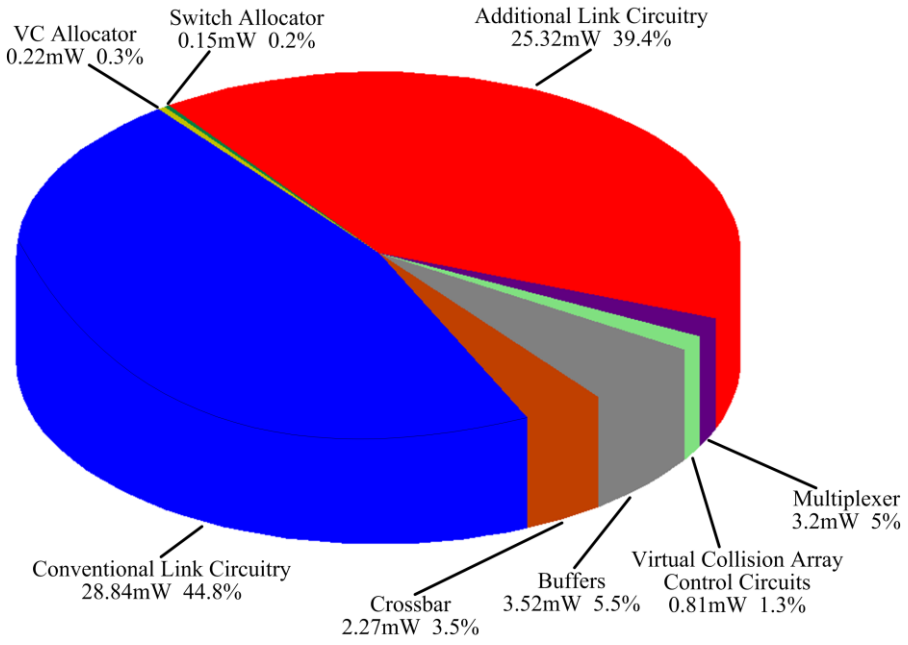


Fig. 2-14 Power distribution

However, due to the reduced number of data packets transmitted in the router pipeline and reduced router delay, the total energy consumption of the proposed design is saved. Fig. 2-15 shows the total energy comparison between the proposed structure and conventional structure on each of the fifteen real applications at 400MHz clock frequency. Energy per bit for the real applications is shown in Fig. 2-16. An average of 56% energy saving can be observed from these two figures. The proposed design increases router power consumption but decreases router delay. In order to gauge the merits, Table II-2 depicts the energy-delay product of the proposed design and compares it with the conventional router design.

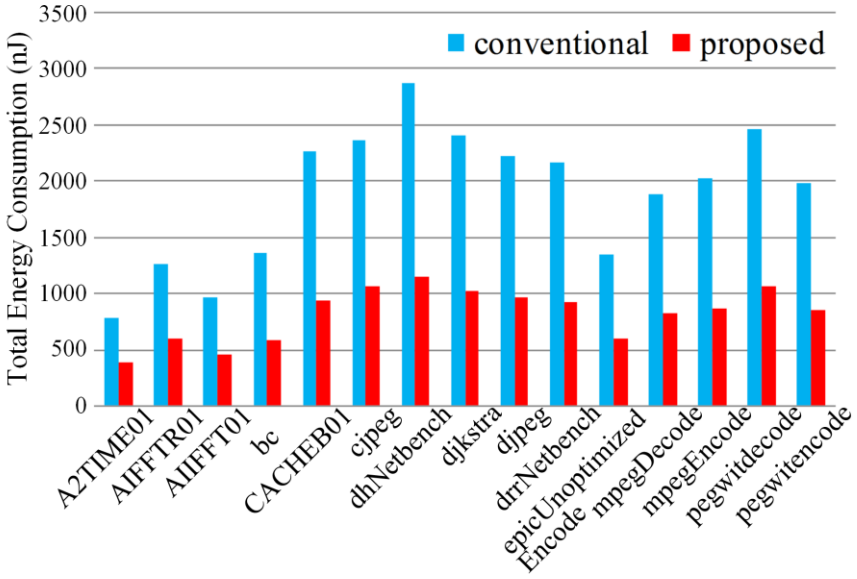


Fig. 2-15 Energy comparison on real applications

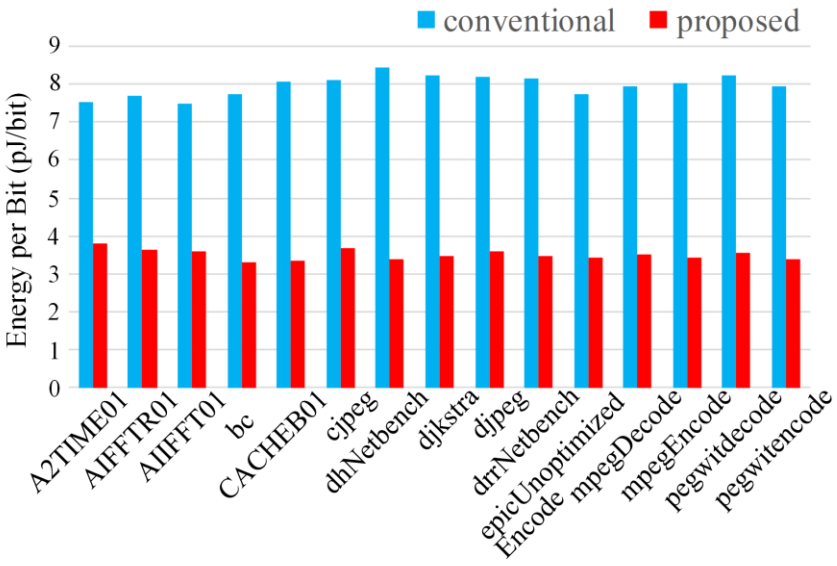


Fig. 2-16 Energy per bit

Table II-2 Energy-delay product

Application Name	**Energy-Delay Product (nJ·s)**	
	conventional	proposed
A2TIME01	1.7e-02	2.4e-03
AIFFTR01	4.6e-02	5.5e-03
AIIFFT01	2.6e-02	3.3e-03
bc	5.3e-02	5.3e-03
CACHEB01	1.5e-01	1.4e-02
cjpeg	1.6e-01	1.8e-02
dhNetbench	2.3e-01	2.1e-02
djestra	1.7e-01	1.6e-02
djpeg	1.4e-01	1.5e-02
drrNetbench	1.3e-01	1.3e-02
epicUnoptimized Encode	5.2e-02	5.5e-03
mpegDecode	1.0e-01	1.1e-02
mpegEncode	1.2e-01	1.2e-02
pegwitdecode	1.7e-01	1.8e-02
pegwitencode	1.1e-01	1.1e-02

CHAPTER 3 MILLIMETER-WAVE ON-CHIP ANTENNA BASED HYBRID NETWORK-ON-CHIP DESIGN

3.1 Current State of Art on NoC Topology

NoC topology is another important parameter to define an on-chip network architecture. It determines the number of routers (hop count) that an inter-core communication needs to be transmitted through the network. Therefore, the network liability, latency and power/energy consumption are also influenced by NoC topology.

There are three basic wired NoC topologies: ring, mesh, and torus. As shown in Fig. 3-1(a), each node in the ring topology has only two connected neighbors. Data packet can only be transmitted to the next node or previous node on ring topology. For mesh topology in Fig. 3-1(b), the inner router has four neighbor routers located at its east, west, north, and south directions, router on the edge has only three neighbors and router in the corner has only two neighbors. For torus topology in Fig. 3-1(c), each router has four neighbors. Compare with mesh and torus topology, ring topology has less links and lower port counts at each router. It is easy to be implemented. However, the simple structure also makes ring topology with fewer alternate paths and higher hop count than mesh and torus topology. Torus topology has the best performance. It has lower hop count, delay and power/energy consumption compared to ring and mesh topology. However, wire lengths in a folded torus topology are twice that in mesh topology of the same size. Therefore, the per-hop latency and power/energy of torus topology are actually higher. In

addition, torus topology requires twice the number of links which must be factored into the wiring budget.

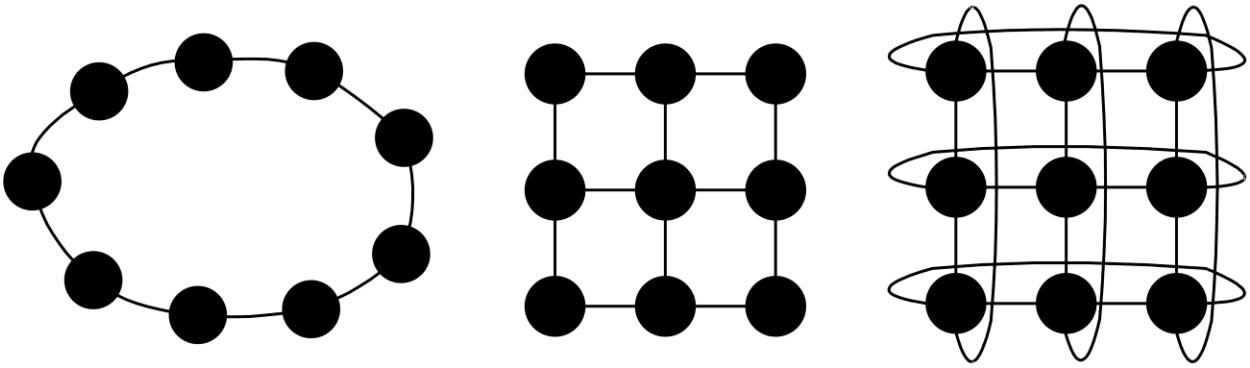


Fig. 3-1 Basic NoC topologies: (a) ring; (b) mesh; and (c) torus [9]

3.2 Hybrid NoC with both Wired and Wireless On-chip Communication Channels

Among diverse NoC topologies, 2D mesh topology is a commonly used one due to its high fault tolerance and short wire length. However, when the number of processors on a single chip increases, the performance of a 2D mesh topology is limited by routing latency and energy consumption [53]. This is because the network diameter of 2D mesh topology increases linearly with the number of nodes. In a 9x9 NoC platform, the longest routing path of 2D mesh topology can reach 16 hops which also introduce large energy consumption on the interconnection links among routers. Different approaches such as low latency/power wired channels [54] and 3D NoC topologies [55] have been introduced to address the long routing latency and high energy

consumption problems. However, all of these approaches still use metal wires. According to the International Technology Roadmap for Semiconductors (ITRS) [56], the performance requirements of multi-core system NoC architecture can no longer be satisfied by the improvements of metal wires. In order to reduce latency and energy consumption, wireless communication has been introduced to the NoC platform [57]-[61].

Recently, several research groups have published important papers in the NoC wireless architecture area. For example, [57] proposed a hybrid mechanism to transfer data among IP cores by taking advantages of both wired and wireless communications. It used data congestion as the decision parameter to choose among wired and wireless links. [58] proposed an adaptable wireless NoC architecture (A-WiNoC) that uses adaptable and energy efficient wireless transceivers to improve network power and throughput by adapting channels according to traffic patterns. Link utilization statistics are used to reallocate wireless channels. There are other papers using the similar strategies [59] [60]. However, none of these papers depicts a clear picture of the on-chip antenna, which is used in the NoC platform to provide wireless communication. How to design the on-chip antenna, how to package and implement it on the NoC platform, what are the performance metrics of the on-chip antenna, how does the antenna provide desired wireless communication on the NoC platform? All of these questions are still unsolved. In addition, due to the large amount of nodes in NoC platform, we have to consider network performance, for example traffic jam as well as energy consumption. This is especially true if we have large amount of data communication from one side of chip to the other sides. To

Summarize, adaptability, low energy consumption, and low latency are the preferable performance metrics for the wireless fabric design. We need reconfigurable antenna with more communication directions and adaptive routing scheme for the wireless NoC.

In this chapter, we review the background of a four-element-array on-chip antenna design as well as its package and implementation on NoC platform. Based on this high frequency high data rate on-chip antenna, we propose a new hybrid NoC architecture (HyNoC). We intend to create a reconfigurable on-chip network with both wired communication and wireless communication. Data packets can be transmitted from the source to the destination by either wired links or wireless links or a mixed of the two communication configurations.

3.3 Background of Millimeter-wave On-chip Antenna

3.3.1 Evaluation of Wireless Interconnects for NoC Platform

Before reviewing the design, implementation and fabrication of the NoC using four-element-array antenna, let's firstly reviewing the evaluation of wireless interconnect for NoC purpose. For NoC platform, power budget is a critical condition. The wireless interconnect on NoC platform must be power efficient to meet the critical requirement. In this section, performance metrics will be introduced to evaluate the NoC using wireless interconnect.

For recovering the free space loss during the wireless transmission, both transmitter (TX) and receiver (RX) are equipped with transmitting and receiving amplifiers. *Friis Transmission*

Equation calculates the ratio of power transmitted at the TX side over power received at the RX side [62]:

$$\frac{P_r}{P_t} = (1-|S_{11}|_t^2)(1-|S_{11}|_r^2)(\frac{\lambda}{4\pi d})^2 G_{t(TX)} G_{t(RX)} \tag{3.1}$$

where d is the transmission distance between the transmitter and receiver, λ is the operating wavelength. $\lambda/4\pi d$ denotes the spherical free space loss. $|S_{11}|_t^2$ and $|S_{11}|_r^2$ are the reflected power numbers for transmitter and receiver RF interconnects. $G_{t(TX)}$ and $G_{t(RX)}$ are the transmitter and receiver radiation gain numbers. The power ratio consists of three components: reflected power terms, the spherical free space loss, and antenna gain terms. On NoC platform, identical antennas are used for both the TX and RX sides. Therefore, (3.1) can be simplified into:

$$\frac{P_r}{P_t} = (1-|S_{11}|^2)^2(\frac{\lambda}{4\pi d})^2 G_T^2 \tag{3.2}$$

(3.2) indicates how much power need to be recovered by TX/RX amplifiers to compensate the signal loss in the wireless channel. (3.2) would be more suitable to be written into the dB form:

$$P_{RE} = -10log\left((1-|S_{11}|^2)^2(\frac{\lambda}{4\pi d})^2 G_T^2\right)(\text{dB}) \tag{3.3}$$

where P_{RE} denotes the power loss from the transmitter to the receiver during free space signal transmission. This P_{RE} must be recovered and amplified by the TX/RX circuits. Lower value of P_{RE} means higher efficiency of wireless system. The wireless link performance needs to be

calculated to quantify the efficiency of antennas. This chapter aims to develop NoC wireless interconnects with very low P_{RE}, to allow very low gain TX/RX amplifiers.

In high frequency wireless interconnect, transmission line is used to connect radio transmitter and receiver with their antennas [63]. However, the transmission line will transform the impedance of an antenna, making it very difficult to deliver power. Radiation power from the generator could be absorbed by the antenna or be reflected. Reflection is wasted power, it losses and reduces efficiency overall. To avoid reflection, the impedance of the antenna must be matched to the transmission line. Fig. 3-2 [64] illustrates a high frequency wireless interconnect example. A transmission line with length of L and characteristic impedance of Z_O is hooked to an antenna with impedance Z_A. The input impedance Z_{in} is given by [64]:

$$Z_{in} = Z_O \frac{Z_A + jZ_O \cdot tan(\frac{2\pi f}{c} L)}{Z_O + jZ_A \cdot tan(\frac{2\pi f}{c} L)} \tag{3.4}$$

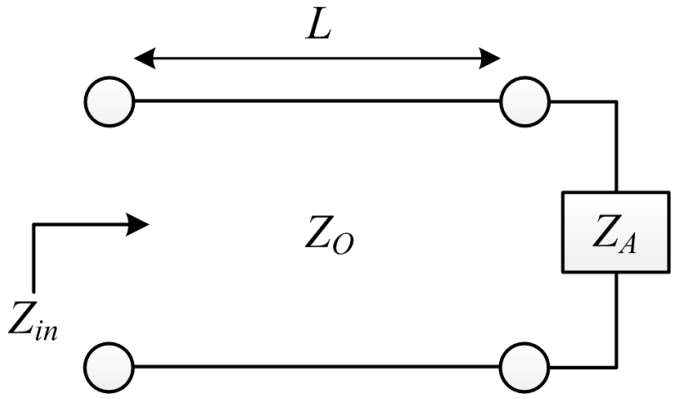


Fig. 3-2 High frequency wireless interconnect input impedance [64]

From (3.4), it can be told that if the antenna is matched to the transmission line ($Z_A = Z_O$), then the input impedance does not depend on the length of transmission line. And if the input impedance is not well matched to the source impedance, not very much power will be delivered to the antenna. The ideal Z_{in} is $50+j0\ \Omega$. j is the square root of -1. The imaginary part of the input impedance provides phase information. None imaginary part means the input voltage and current are exactly in time-phase.

The reflection coefficient is used to measure the power loss in reflection. Besides reflection losses, there are also free space signal losses and resistive losses. Transmission coefficient is used to measure the left over. It is a measurement of how much energy is detected by the receiver antenna and how much energy is lost over the air interface. The ideal transmission coefficient is equal to 1 or 0 dB ($10log1 = 0$). In the on-chip antenna design, the transmission coefficient is maximized and the reflection coefficient is minimized.

3.3.2 On-chip Antenna Package and Implementation

On-chip antennas with millimeter-wave frequency (mmW) must be chip/package compatible with the silicon integrated circuits. Several types of on-chip antennas with different frequencies have been investigated on silicon substrate with the standard CMOS process [65-68]. Unfortunately, these existing designs only achieved 10% to 15% radiation efficiencies. The bottleneck of efficiency lies in the low silicon conductivity. Due to the high dielectric constant ($\varepsilon r = 11.68$) of silicon, most of the antenna energy radiates and dissipates in the silicon. Simply,

to prevent the dissipation of radiation energy, it just need to directly place a radiation director, a thin metal layer, under the antenna. However, this would hardly work in the NoC platform. There are two major reasons forbidding placing a thin metal layer directly under the NoC on-chip antenna. First one is the dielectric layer of the on-chip antenna is very thin. In the current CMOS fabrication process, it is only 15μm or less. Secondly, the directly placed metal will also block the NoC on-chip routing.

With the above concerns, the millimeter-wave on-chip antenna is designed with the ground shielded metal inside the package to prevent the radiation energy being dissipated. One problem of this design is that the ground shielded metal has been proven to add difficulties to the impedance matching [62]. Therefore, instead of using perfect electric conductor (PEC), perfect magnetic ground conductor (PMC) is used to make the ground shielded structure of the millimeter-wave on-chip antenna. Low-profile antenna (on-chip antenna is low-profile antenna) in package mounted on the ground plane generates an image current source. The image current created by PEC deteriorates the original current source due to its opposite direction to the original antenna current source [69]. A good solution for the image current effect is using PMC as the ground plane that creates an identical directional image current source to the original antenna current direction. Fig. 3-3 [78] [92] compares the images created from horizontal electric dipole antennas with PEC and PMC as the ground shielded structure. Fig. 3-3(a) shows that if the antenna is placed very close to an electric ground plane, then the image set up within the ground cancels the radiation currents in the antenna. Fig. 3-(b), however, shows that if the antenna is

placed very close to a magnetic ground plane, the image does not cancel the radiation currents in the antenna. In the same figure, it also shows the phase of the reflection coefficient for PEC and PMC ground planes. For PEC, $\theta_r = 180°$, and for PMC, $\theta_r = 0°$.

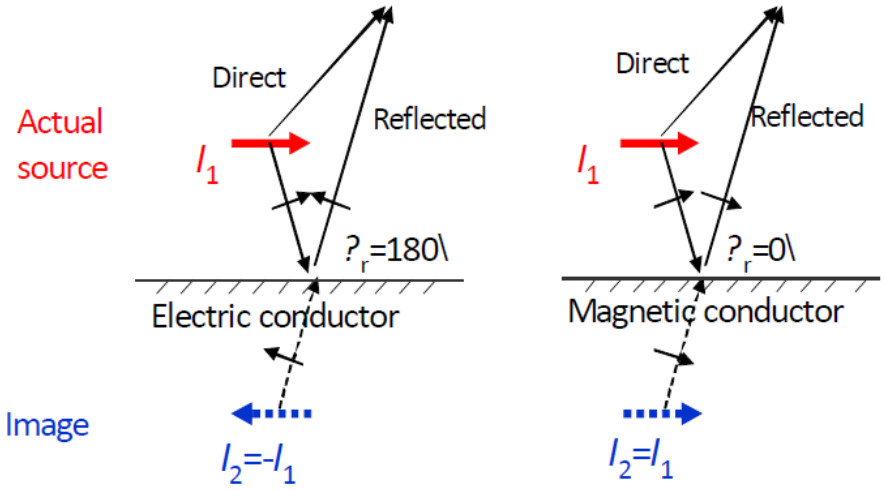


Fig. 3-3 Image created over PEC and PMC ground plane [78] [92]

It has been proven that artificial magnetic conductor (AMC) structure can be used to implement PMC [70-72]. AMC operating as PMC at specific frequencies has been introduced in [73]. The AMC structure creates a parallel resonant circuit with inductance and capacitance, which creates very high impedance near the resonant frequency [70]. The high impedance surface prevents the surface wave propagation at the resonant frequency.

HFSS [74] is used in simulation to determine the patch dimensions of the AMC layer. The unit cell of the AMC layer with the rectangular waveguide simulator is shown in Fig. 3-4 [92]. The initial setup value of AMC patch gap and height are g = 0.12 mm and h = 0.203 mm.

According to the reflection phase of HFSS simulation, the width of AMC patch can be determined as w = 0.56 mm. Fig. 3-5 [76] [92] and Fig. 3-6 [92] depict the top view and cross-sectioned view of the on-chip antenna and AMC layer design. A hydrocarbon ceramic type laminate, Rogers 4003, is used as the substrate material, which enables the copper-plated through hole generation [75]. The chemical wetting process is not required before producing the copper-plated through hole vias. Multi-layer Rogers 4003 laminates can be assembled with a standard printed circuit board (PCB) fabrication process. The dielectric constant and dielectric dissipation factor are ε_r = 3.55 and $tan\delta$ = 0.0027 s/m, respectively.

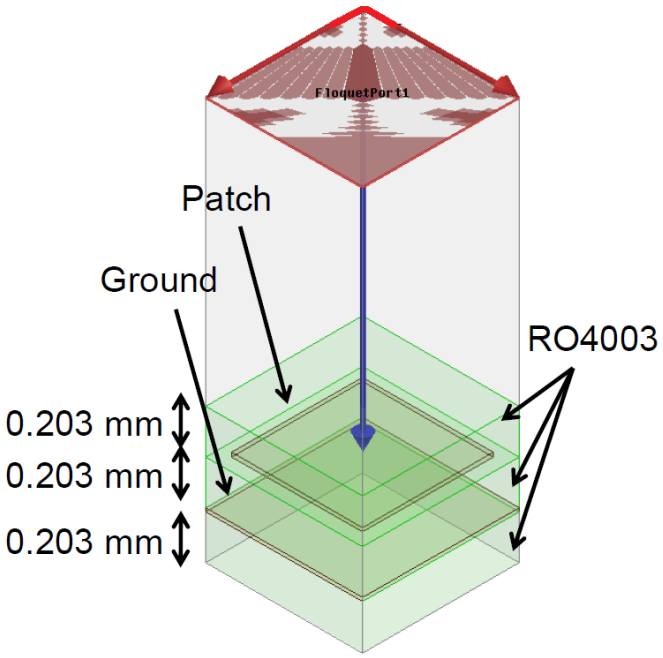


Fig. 3-4 Unit structure and dimensions of periodic AMC layer [92]

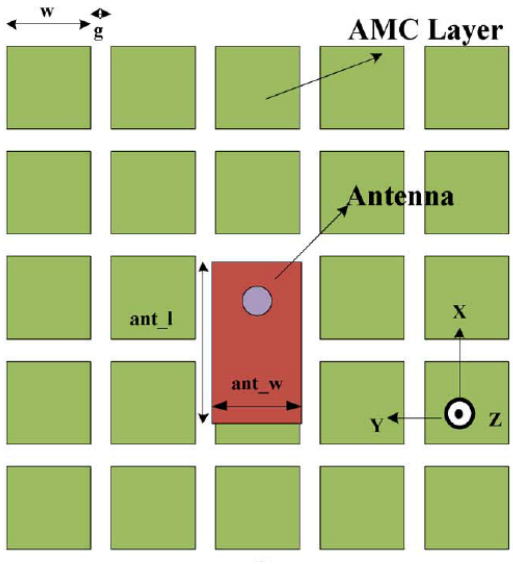


Fig. 3-5 On-chip antenna configuration (top view) [76] [92]

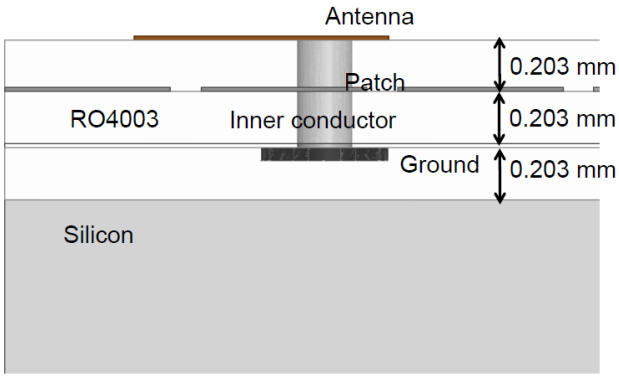


Fig. 3-6 On-chip antenna configuration (cross-sectioned view) [92]

Fig. 3-7 [78] [92] shows the simulated antenna reflection phase on the designed AMC layer. It can be observed that the reflection phase is about 0 degree around the 60 GHz frequency point,

which is the operating frequency of the millimeter-wave on-chip antenna. This means the current source and the image current are in phase around 60 GHz.

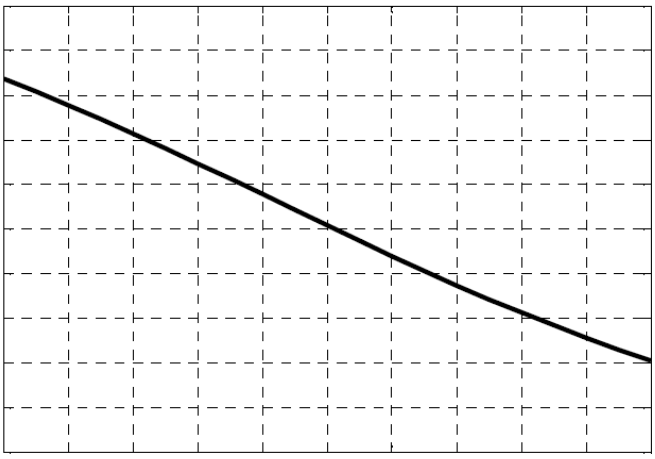


Fig. 3-7 Simulated phase of reflection coefficient [78] [92]

3.3.3 Folded Monopole Antenna and Two-element-array Antenna

As shown in Fig. 3-8, a folded monopole antenna (also called single-element-antenna) [76] [92] is able to communicate with another single-element one in orthogonal direction. However, it may not be sufficient in diagonal direction since the longer distance between the transmitter and receiver. A two-element-array antenna [77] [92] can communicate in diagonal directions due to its strong radiation. To illustrate, the two-element-array antenna is able to create stronger directivity [77] [92] compared with the folded monopole antenna. Fig. 3-9 [92] demonstrates the gain pattern from two identical folded monopole antennas. The gain of such two-element-array

antenna can be enhanced about 5 dBi in the horizontal direction which provides strong and clear communication signals.

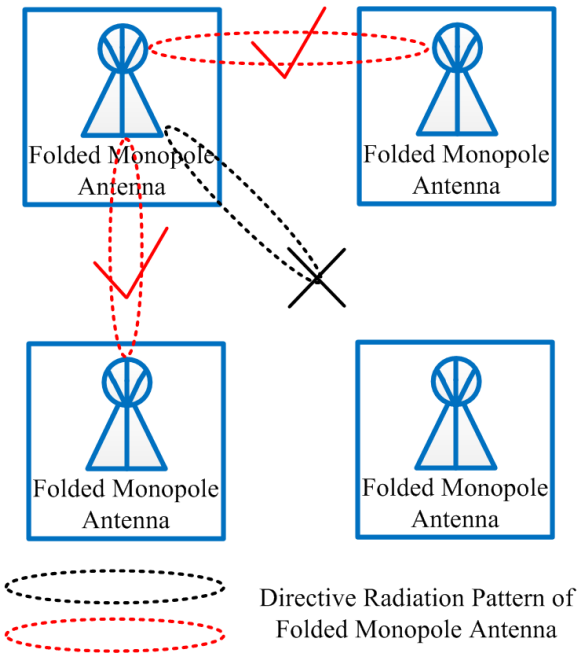


Fig. 3-8 Radiation pattern of folded monopole antenna

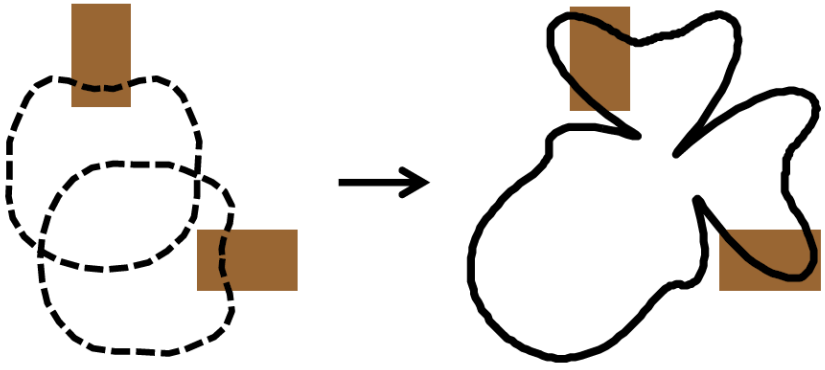


Fig. 3-9 Gain pattern created from two identical antennas [92]

3.3.4 Four-element-array Antenna

On-chip antenna for NoC platform is required to provide wireless communicate in both orthogonal and diagonal directions with strong and clear communication signals. As we discussed in the previous section, folded monopole antenna can provide wireless communication in the orthogonal direction, and two-element-array antenna can provide wireless communication in the diagonal direction. The direction restriction of folded monopole and two-element-array antenna requires multiple antennas for wireless communication on a NoC platform. Due to the high fabrication cost and power consumption, using four single-element-array antennas and four pairs of two-element-array antennas are not an ideal option for on-chip multicore systems.

To meet NoC requirements, one option is to provide an effective, reconfigurable array-antenna orientation to provide optimized gain. This is the reason why we investigate a reconfigurable four-element-array antenna for NoC platform. Based on two-element-array antenna, four-element-array antenna is designed by using four folded monopole antennas. The orientation and distance between these four antennas are carefully designed. By activating a combination of two from the four folded monopole antennas, a reasonable radiation pattern direction and radiation gain can be created. Fig. 3-10 [92] shows the placement of the proposed four-element-array antenna on AMC layer. The black dot on each folded monopole antenna identifies the antenna directivity. In this design, all of the four folded monopole antennas are orientated to the center. This four-element-array antenna is called in star shape orientation.

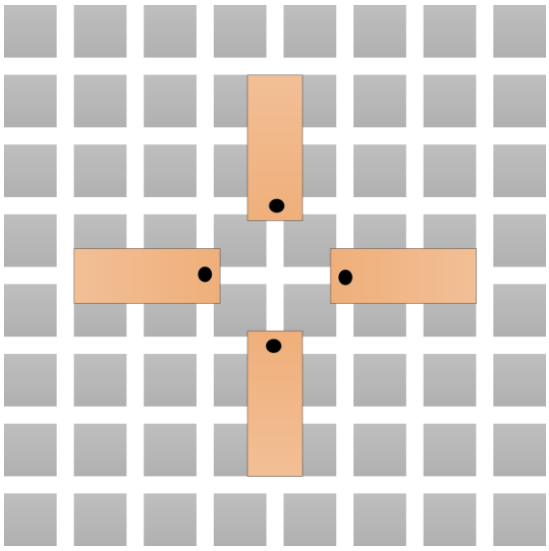


Fig. 3-10 Four element array antenna in star shape orientation [92]

The individual folded monopole antenna is expected to create the orthogonal direction of data communication while a pair of antennas is designed to produce the wave propagation in diagonal directions. Fig. 3-11 [92] shows four combinations cases of pair of antennas turned on. The black line represents the individual antenna directive gain pattern. The red line represents the two antennas directive gain pattern. The maximum gain of two antennas would be obtained at $\phi = 0^{\circ}$, $\phi = 270^{\circ}$, $\phi = 90$, and $\phi = 180^{\circ}$, respectively for case (1), (2), (3), and (4) in Fig. 3-11. At other angles, wireless communication can also be performed with a lower antenna radiation gain.

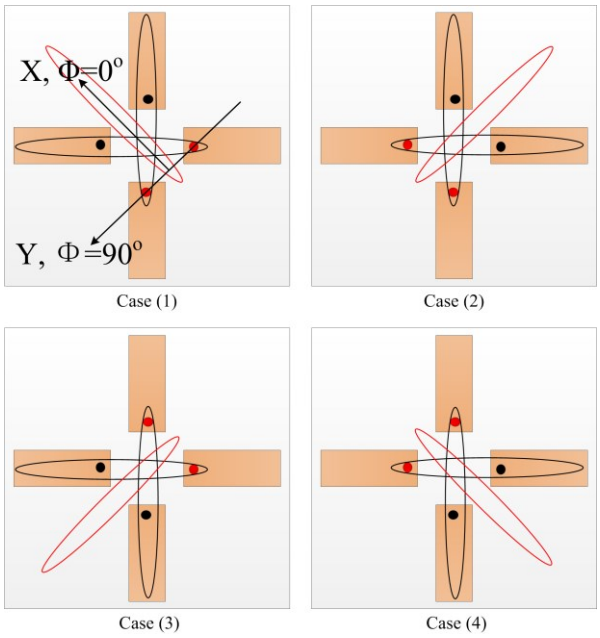


Fig. 3-11 Gain combinations of four-element-array antenna [92]

3.4 Reconfigurable Hybrid NoC Architecture

3.4.1 HyNoC Topology

The proposed HyNoC architecture is an evolution version of the conventional wired 2D mesh NoC architecture. As shown in Fig. 3-12, each processor uses a router to transmit data through the multi-core system. For mesh topology, each router uses short wired bidirectional interconnections to communicate with a central core and four neighbor routers located at its east, west, north, and south direction (routers on the edge have only three neighbors and routers in the corner have only two neighbors). The destination address is carried in the header flit of each data packet. The router follows a “push-model” style, where data packets are pushed from a source to a destination with one hop or multiple hops [34].

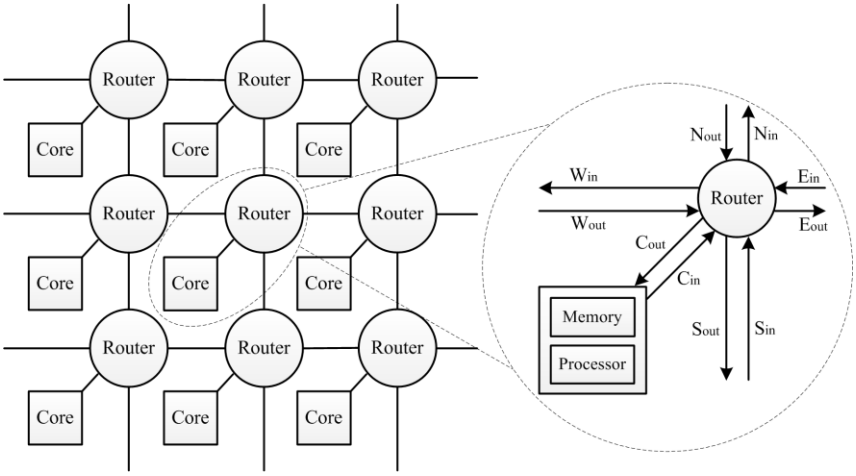


Fig. 3-12 Conventional 2D mesh NoC architecture

The proposed HyNoC uses four-element-array on-chip antennas introduced in the previous section to provide wireless interconnections. Instead of fixed communication direction, the reconfigurable combination of pairs of folded monopole antennas can communicate with each other in all directions. This feature allows us to extend the conventional mesh topology of traditional wired NoC architecture to achieve better performance with lower consumption. One significant advantage of wireless NoC over the wired NoC is hop count reduction. For example in Fig. 3-13, if a data packet has a source at *Node (0,0)* and destination at *Node (8,8)*, wired NoC needs 16 hops to transmit this data packet from the source to the destination. However if we have antennas on *Router (0,0)* and *Router (8,8)*, the wireless NoC only needs one hop to transmit this data packet. There is a requirement for on-chip antenna to provide one hop data transmission: the communication distance of on-chip antenna must be able to cover the distance between any transmitter and receiver on the NoC platform. For example, Tilra TILE-Gx36, TILE64/Pro64,

and TILE-Gx100 have 36, 64, and 100 processors, respectively. These processors are integrated on a single chip with a package size of 3.5cm x 3.5cm, 4cm x 4cm, and 4.5cm x 4.5cm, respectively [79]. When design the HyNoC topology, the on-chip antenna layout should be capable of providing decent wireless communication signal between the farthest diagonal transmitter and receiver.

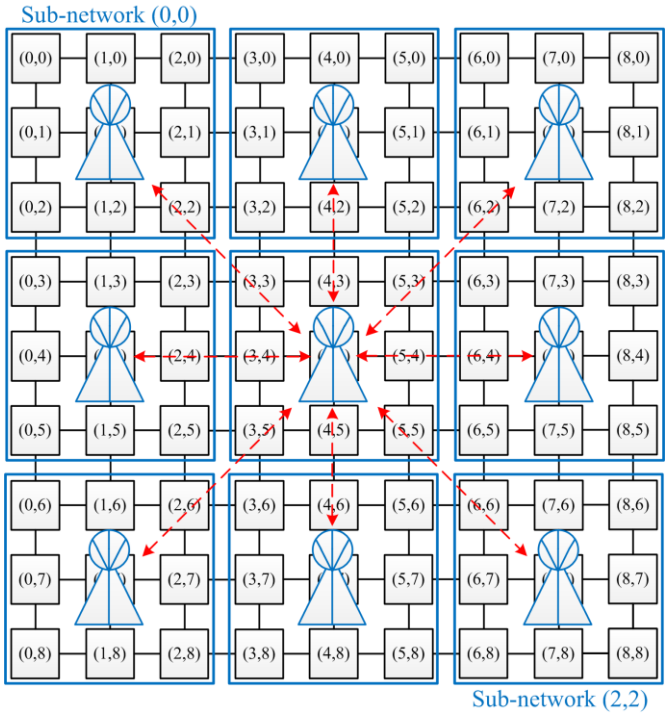


Fig. 3-13 9x9 HyNoC architecture

The topology of on-chip wireless network is determined by the number and locations of reconfigurable antennas implemented on the NoC platform. For a 9x9 multi-core system, we

cannot integrate each node with a four-element-array antenna. It would be an unaffordable budget in both money and area. In addition, it is also a waste of resources. The network can be divided into sub-networks; and one set of four-element-array antenna needs to be shared by several nodes in the same sub-network. Routers belong to the same sub-network can transmit data by wired links; Router belong to different sub-networks can preferentially choose the on-chip wireless network to transmit data packets. For more nodes in a sub-network sharing one set of four-element-array antenna, the cost and area overhead of the chip can be lower. However, this may introduce more congestion and the network performance can be lower. Fig. 3-13 illustrates an example of a 9x9 multi-core system divided into 9 sub-networks. Each sub-network contains 9 nodes. The four-element-array antenna in each sub-network can provide wireless communication with the other eight sub-networks. Each set of four-element-array antenna is integrated at the center of the sub-network. It is wired connected to and shared by 9 routers.

Wired links are used among the 9 routers in the same sub-network for data transmission. The longest inner sub-network communication has only 4 hops. For the inter sub-network communication, wireless links are used as the priority for the speed and energy concern. However, one antenna is shared by several routers, and some traffic patterns would cause hot spots on the network. The on-chip wireless network may suffer severe congestion which brings extremely high latency and low throughput. Therefore, we need an adaptive routing algorithm to switch between wireless link and wired link for the inter sub-network communication. We will discuss the adaptive routing algorithm in the following section.

3.4.2 HyNoC Routing Scheme

Routing scheme specifies the path for a data packet transmitting from the source to the destination. A good routing algorithm should distribute data traffic evenly on the network and avoid creating hot spots. A good routing algorithm should also provide deadlock freedom, low network latency, and high system throughput [80].

For the data communication with source and destination belonging to the same sub-network, it is not necessary to use wireless links. We use wired link to transmit data packet inside one sub-network. To simplify the design and implementation as well as guarantee the deadlock freedom, we choose dimension-ordered routing (DOR) scheme [9] for the conventional wired NoC. For the data communication with source and destination belonging to different sub-networks, data packets may be transmitted by wireless links, wired links, or both. The wireless interconnection of NoC has been proved with both low latency and high energy efficiency. Therefore, wireless link is the first choose. However, to avoid the wireless congestion, we may also choose wired links.

To adapt the real time network situation, the decision of routing path needs to be made at every sub-network. The routing scheme is a function of packet current location (c) and destination (d), wired transportation hop count (H), wireless buffer utilization (BUF_{AN}), and network energy consumption (E). Algorithm 3.1 illustrates the procedure of the routing scheme. This routing algorithm is executed when a data packet is injected into each sub-network. The 'RoutComp' operator takes input as a data packet. It evaluates the address information stored in

the header flit and computes the current location *c* and destination *d* of the data packet. Number of wired transportation hops from the current location to the destination is computed by the 'HopComp' operator using *c* and *d*. We define a wired communication hop threshold in line 2. This threshold is used to find out whether the current location and the destination are far enough to use wireless links for data transmission. If *H* is equal to or smaller than the threshold value, it indicates that the current location and the destination are in the same sub-network or at the edge of two adjacent sub-networks. It is not necessary to use wireless links for data transportation. If *H* is larger than the threshold value, the routing algorithm checks the current wireless buffer utilization in line 6. We use buffer utilization to measure congestion on the network. If the current wireless buffer still has space to store the data packet, wireless links would be chosen to transmit the data packet. However, if the current wireless buffer is full, which means it may create congestion on wireless links; the routing algorithm further calculates the network energy consumption for transmitting the data packet in line 10. If we use wired links to transmit the data packet, *threshold+1* wired hops would be added before the data packet being injected into the next sub-network. Therefore, the energy consumption is calculated as the summation of energy consumed in the previous hops, wired energy consumption for *threshold+1* hops, and wireless energy consumption for the rest transmission. If the energy consumption is lower than the upper limit, wired links are chosen to transmit the data packet to the next sub-network. Therefore the congestion on wireless links can be alleviated. If the energy consumption already violates the upper limit, wireless links are chosen anyway for the energy concern. The result of the adaptive

routing algorithm is stored in the header flit of each data packet and passed to the next sub-network.

Algorithm 3.1 Adaptive Routing Algorithm

---

input: a data packet D_{Pack}, current wireless buffer utilization BUF_{AN}, energy consumption of a data packet traversing a wired hop E_{hop}, energy consumption of a data packet traversing wireless link $E_{wireless}$, upper limit of network energy consumption for a data packet E_{up};

output: routing decision *RoutDec*;

1: $\{c, d\} \Leftarrow$ RoutComp(D_{pack}); $H = \Leftarrow$ HopComp(c, d)

2: **if** $H \leq threshold$ **do**

3: $RoutDec \Leftarrow$ WiredRout(c, d)

4: **end if**

5: **else do**

6: **if** BUF_{AN} is not full **do**

7: $RoutDec \Leftarrow$ WirelessRout(c, d)

8: **end if**

9: **else do**

10: $E \Leftarrow E + (threshold + 1) * E_{hop} + E_{wireless}$

11: **if** $E < E_{up}$ **do**

12: $RoutDec \Leftarrow$ WiredRout(c, d)

13: **end if**

14: **else do**

15: *RoutDec* $\Leftarrow$ WirelessRout(*c, d*)

16: **end else**

17: **end else**

18: **end else**

19: **return** *RoutDec*

3.5 HyNoC Workload Assignment and Task Scheduling

To facilitate the proposed hybrid architecture and routing algorithm, the workload assignment and task scheduling algorithm is presented in this section. The proposed scheduling algorithm aims to minimize total routing time as well as to minimize the total communication energy consumption. Capacity Rate is introduced as an indication of how much workload one core can sustain during workload assignment. Workloads are fundamentally composed of sets of tasks, named task flows. Tasks from different flows tend to have very few or no dependencies among them. We assign the process of one task flow to a group of cores. Such a group of cores is defined as a partition. The partitioning algorithm aims at mapping a task flow onto one particular partition. Task scheduling will accordingly be done within the partition that is designated to this task flow.

Generally a task flow is represented by a DAG (Direct Acyclic Graph) $G = (T, E)$ with T denoting a set of tasks to be executed and $E = \{(i, j)\}$ specifying precedence relationships among tasks. Each task T_i is associated with a task weight w_i representing its execution time. The workload of this task flow can be evaluated by the structure of the task graph and the task weights. The details about workload estimation can be found in [44]. When performing partitioning, the partition must be able to accept the workload evaluated based on the given flow. In addition, the total power consumption for all cores in use should be less than an upper bound limit.

After partitioning, the next step is task scheduling within the partition designated to a task flow. The goal is determine the task-core mapping results to achieve minimum routing time and minimum communication energy consumption. We formulate the task scheduling and mapping problem as a mixed-integer program (MIP), by introducing a set of binary variables $\{M_{ij}$'s$\}$ to represent all task-core mapping results. Each variable M_{ij} is a decision variable of binary type, which is set to "1" if task T_i is mapped onto the j-th core and "0" as the opposite situation. Suppose a flow of m tasks is to be executed within a partition of m cores. The total number of decision variables is thus (n x m). For each task T_i the definition of this binary variable imposes the following constraint:

$$\sum_{j=1}^{m} M_{i,j} = 1 \tag{3.5}$$

which indicates that one task can be assigned to only one core. Using these binary variables, task allocation can be conveniently determined.

In addition to the binary mapping variables, another set of decision variables are the starting times for all tasks, denoted by d^s. Task starting times are combined with the task-core mapping variables to determine the job sequences of all involved cores. An optimal schedule can be fully specified by determining the binary variables M_{ij}'s and task starting times d^s: the decision variables related to j-th core indicates a set of tasks to be executed on it, and the starting times help determine the execution sequence of all assigned tasks. This introduces the second set of constraints in task scheduling, which is for the purpose of keeping the precedence relationships among tasks. For each arc (i, j) on the task graph, the precedence relationship requires that task T_i must be finished before the execution of its successor T_j:

$$d_i^s + d_i^e + d_{i,j}^{comm} \leq d_j^s \tag{3.6}$$

where d_i^s and d_i^e represent the starting time and execution time for task T_i, while $d_{i,j}^{comm}$ represents the communication time between task T_i and T_j.

There are two other constraints for each core in the scheduling framework: workload constraint imposed by core capacity rate and upper bound limit of power consumption. Capacity rate is included as an upper bound limit for workload assigned to each core. A core with low capacity rate indicates that small or no workload should be assigned to this core. The workload on a core depends on the task allocation and the frequency scaling ratio. The estimation of core

workload is discussed in [44]. When performing task scheduling, the real workload assigned to a core cannot exceed a pre-evaluated value provided by the capacity rate; otherwise, it may exceed the performance limits and cause reliability issues.

The last concern in this scheduling framework is the optimization objective. One objective is to minimize the inter-core communication time to guarantee system performance when processing a task flow. The other objective is to minimize the total communication energy consumption, which is determined by energy consumption of a data packet traversing a wired hop and traversing a wireless link. In summary, task scheduling in this proposed router design can be generalized by the following mixed-integer program:

$$\begin{aligned}
&\textit{minimize} && \sum_k t_k \\
&\textit{minimize} && E_{total} = \sum_k E_k \\
&\textit{subject to} && d^s_i + d^e_i + d^{comm}_{i,j} \le d^s_j \\
& && CR_k \ge Workload_k \\
& && P_k \le P_{upk} \\
&\textit{variables} && M_k, d^s = \left\{ d^s_1, d^s_1, ..., d^s_n \right\}
\end{aligned}$$

where M_k denotes the set of mapping variables, which exert a constraint in this MIP to guarantee the uniqueness of the task-core mapping relationship. t_k represents the total routing time for a particular inter-core communication, and the objective function therefore is the summation of all

t_k's. This objective function is optimized simultaneously with the other objective of minimal routing energy consumption. d_i^s and d_j^s are the starting times of T_i and T_j, while d_i^e is the execution time of task T_i on a particular core, and $d_{i,j}^{comm}$ is the communication time between T_i and T_j, which is determined by the routing delay. The execution time of Task T_i must be completed before the execution of T_j, including the communication time. For each core processing a part of the task flow, its capacity rate CR_k must be larger than the workload of these tasks. Additionally, there is an upper boundary P_{upk} for each core's power consumption P_k. To solve this formulated MIP, the scheduling algorithm searches task-core mapping variables with simulated annealing to obtain a near optimal solution.

3.6 Result Analysis

In this section, we firstly analyze the performance of the proposed on-chip antenna. P_{RE}, input impedance, reflection coefficient, one antenna gain pattern, and two antennas gain pattern of the four-element-array antenna will be presented and analyzed. Then we compare the proposed reconfigurable HyNoC design with the conventional NoC architectures, to be specific, mesh, torus, and butterfly topologies. The HyNoC performance evaluation includes packet latency, system throughput, speedup, power, and energy consumption. A data packet size of four 64-bit flits is used in all experiments. We use Booksim2.0 [45], a cycle-accurate interconnection network simulator, to simulate a 9x9 on-chip network with 16 virtual channels per physical channel, 8 flits per virtual channel. We use virtual-channel flow control to improve link

utilization. To establish a fair comparison, we simulate different data traffics and generate different task graphs using real applications from three benchmark suites: MediaBench [46], MiBench [47], NetBench [48], and EEMBC [93]. The power and energy results for the on-chip network components are estimated by Xilinx ISE Design Suite 13.2 and Xilinx Power Analyzer with the Xilinx Virtex-6 library [49-51]. All the experiments are running on a computer with Intel Core i5-3230M CPU at 2.6GHz and Linux operating system.

Antenna dimensions are the main factors to determine the resonant frequency and input impedance. The dimension of the 60 GHz on-chip antenna needs to be turned to match the input impedance. After the antenna turning process, the optimized values of antenna width and antenna length are *ant_w* = 0.4 mm and *ant_l* = 1 mm. Fig. 3-14 [92] shows the four-element-array antenna input impedance Z_{in} as a function of frequency. The desired antenna Z_{in} is 50+j0 Ω. Real part = 50 Ω and imaginary part = 0. It can be observed from this figure that the on-chip antenna has nearly flat input impedance close to 50 Ω from 54 GHz to 65 GHz.

After matching the antenna input impedance to 50 Ω, Fig. 3-15 [92] presents the measured reflection coefficient for the 60 GHz four-element-array antenna on AMC layer implementation. This figure shows how much power can be transmitted through the wireless interconnect. The measured result shows good match performance around 60 GHz: a local minimal reflection coefficient region from 58 GHz to 62 GHz can be observed from Fig. 3-15. This result indicates that a high radiation efficiency can be achieved based on (3.3).

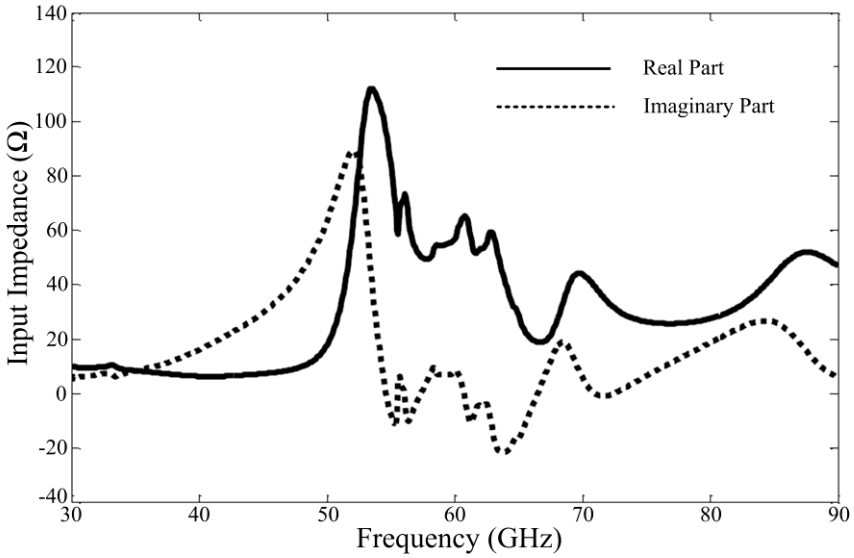


Fig. 3-14 Input impedance vs frequency [92]

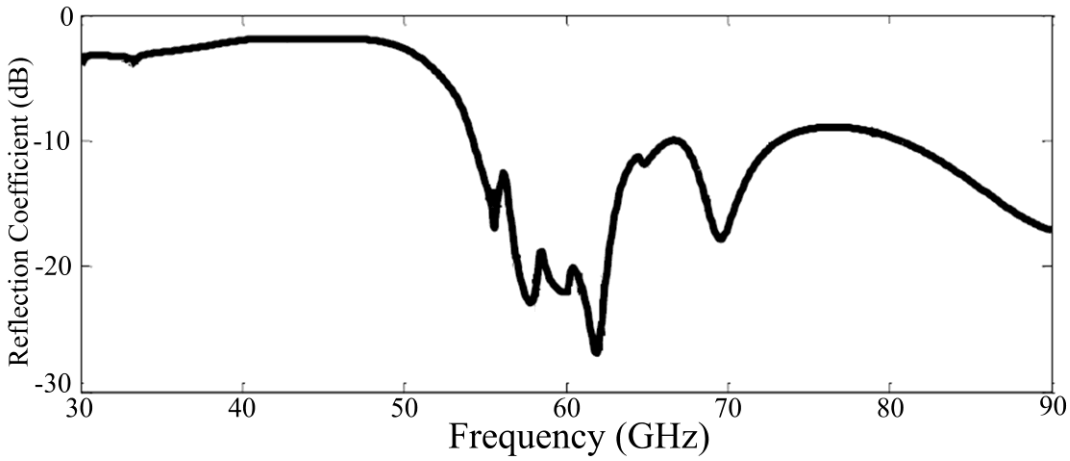


Fig. 3-15 Measured reflection coefficient vs frequency [92]

For the proposed four-element-array antenna, comparing with only one antenna on, the additional transmission gain of two antennas on can reduce the required P_{RE}. Fig. 3-16 [92]

depicts P_{RE} as a function of the antenna communication distance. With the increase of distance between the transmitter and receiver, radiation power required to be recovered and amplified by the TX/RX circuits also increases. It can be observed from Fig. 3-16 that when two antennas are turned on, the P_{RE} is at least 10 dB lower when compared to just one antenna is turned on.

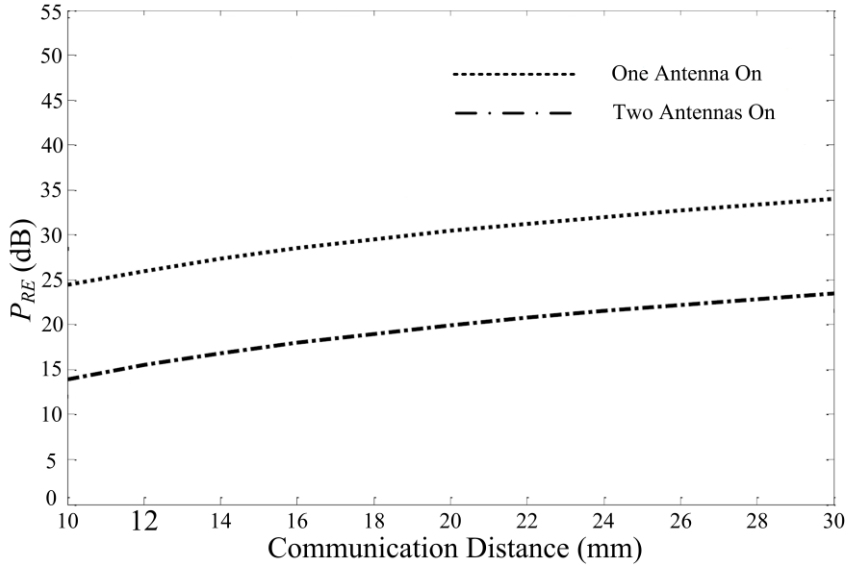


Fig. 3-16 P_{RE} vs communication distance [92]

Radiation Gains of the final four-element-array antenna are shown in Fig. 3-17 [92]. The symmetric structure of the star shape orientation antenna system results in a symmetric shape of gain. Fig. 3-17(a) shows the case of one antenna turned on. The maximum gain is 4.19 dBi at ϕ = 220˚ and ϕ = 140˚ for orthogonal direction communication. Fig. 3-17(b) shows the case of two

antennas turned on. The maximum gain is 7.23 dBi at ϕ = 225° for diagonal direction communication.

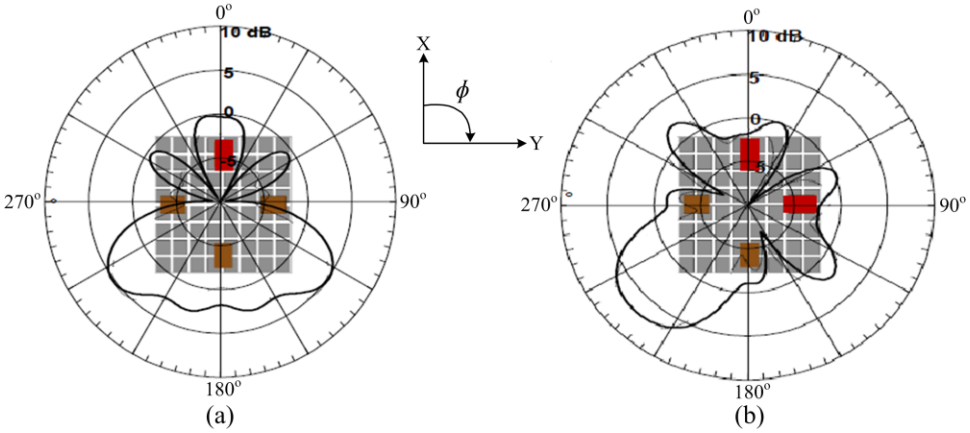


Fig. 3-17 Radiation gain pattern of the proposed on-chip antenna [92]

For proving the adaptivity of the proposed HyNoC architecture, we simulate three different data traffic patterns: Neighbor, Uniform, and Random Permutation [52]. Fig. 3-18 demonstrates the HyNoC's significant packet latency and system throughput improvement, compared with the conventional mesh, torus, and butterfly network topology on all of the three data traffics. Traffic 1 is Neighbor traffic. Network throughput of the conventional mesh, torus, and butterfly topology saturates at 0.3, 0.3, and 0.04 offered loads, respectively. Network throughput of the proposed HyNoC architecture saturates at 0.9 offered load. It shows an increase of 200% in throughput over the conventional mesh and torus network architectures. Traffic 2 is Uniform traffic. Network throughput of the conventional mesh, torus, and butterfly topology saturates at

0.13, 0.2, and 0.25 offered loads, respectively. Network throughput of the proposed HyNoC architecture saturates at 0.38 offered load. It shows an increase of 203%, 84%, and 50% in throughput over the conventional mesh, torus, and butterfly network architectures. Traffic 3 is Random Permutation traffic. Network throughput of the conventional mesh, torus, and butterfly topology saturates at 0.06, 0.09, and 0.11 offered loads, respectively. Network throughput of the proposed HyNoC architecture saturates at 0.19 offered load. It shows an increase of 200%, 113%, and 70% in throughput over the conventional mesh, torus, and butterfly network architectures. The latency plots show the network saturates at the same point as the throughput plots for all of the three traffic patterns. As long as the increase of offered load, the proposed HyNoC architecture has consistently lower packet latencies than the conventional mesh, torus, and butterfly network topologies. At the same time, packet latencies of the proposed HyNoC architecture saturate much slower than which of the conventional topologies.

To evaluate the routing delay and measure the speedup of the proposed technology, we generate different task graphs using real applications from three benchmark suites: MediaBench [46], MiBench [47], NetBench [48], and EEMBC [93]. Table III-1 lists fifteen real applications as well as their routing delay on mesh, torus, butterfly, and HyNoC topologies. The first column lists the name of each real application. The second column provides the total number of tasks in each application task flow. The third column shows the number of cores in use after applying task flow partitioning algorithm. For the task mapping on the multicore system, some benchmark applications have very simple task flow with few tasks. In this case, few cores are used for

application tasks mapping on. If the network is empty and only tasks from one simple application are mapping on the multi-core system, all of these tasks will be mapped onto one single sub-network. For our HyNoC architecture, no wireless link would be activated if all inter-core data communication is in the same sub-network. Each inter-core data communication uses wired links to transmit data packets. The performance of the proposed HyNoC architecture will not show any improvement compared with the conventional mesh network topology. Without utilizing the wireless network, even torus or butterfly topology works better than the hybrid NoC topology. To demonstrate the performance improvement of the proposed HyNoC architecture, it is more likely that the multi-core system is always working under a very busy situation. Most cores are almost fully occupied by existing workloads. There are only a few cores scattered on the system that can take the application tasks. Fig. 3-19 uses the *A2TIME01* applications as an example. After the partitioning algorithm, the task flow of the *A2TIME01* application is divided into 4 sub-flows. The blocks in yellow in Fig. 3-19 represent the cores having heavy workloads. The capacity rates of these cores are very low. The *A2TIME01* application tasks cannot be assigned to these cores. The blocks in blue scattered on the system represent the cores having high capacity rates. These cores are not working with heavy workload, and therefore can accept the *A2TIME01* application tasks. Our task mapping algorithm minimizes routing delay and communication energy consumption as the goal. Finally, as shown in Fig. 3-19, the *A2TIME01* application tasks are assigned to *Core (0, 1)*, *Core (6, 0)*, *Core (4, 3)*, and *Core (3, 8)*.

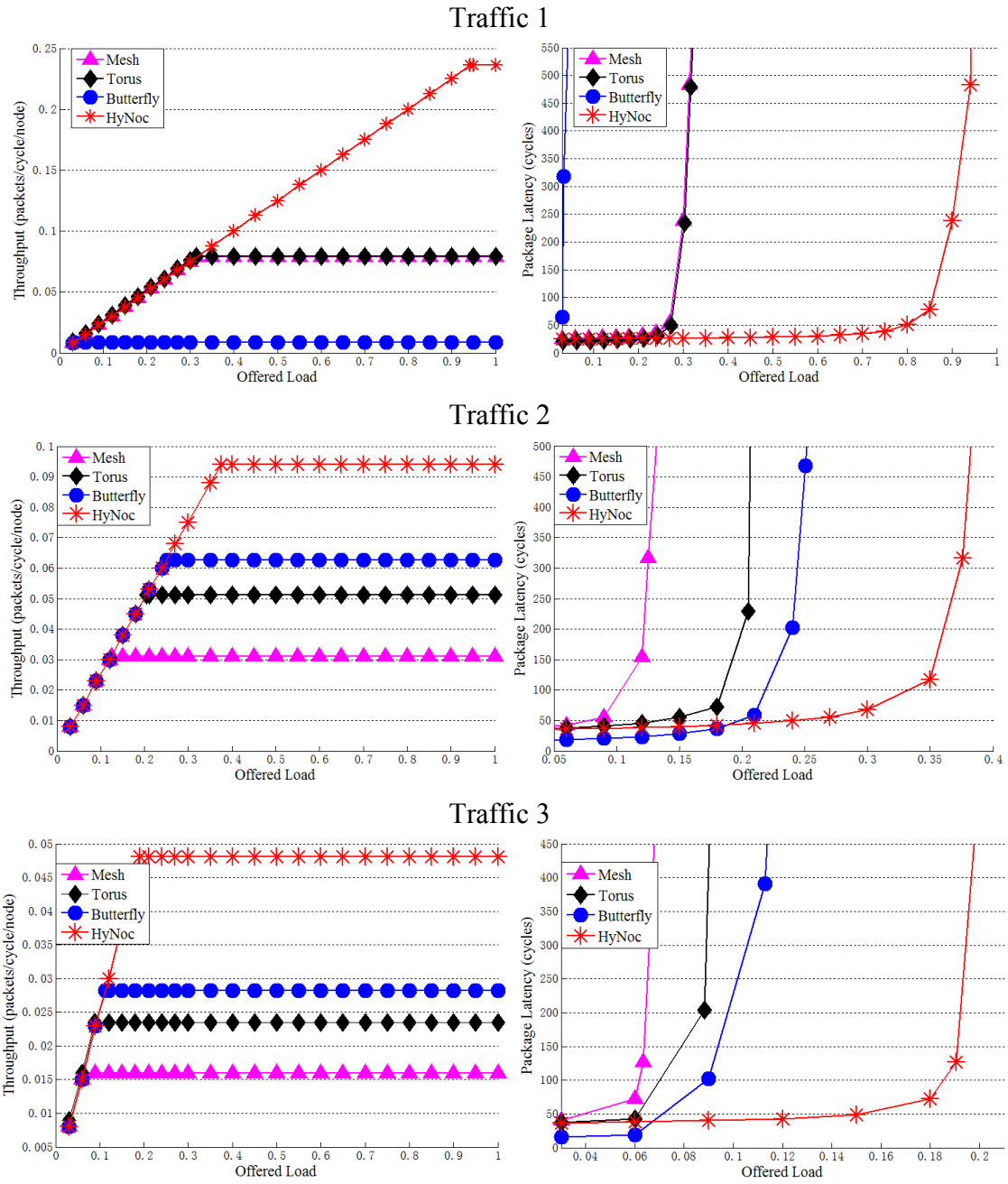


Fig. 3-18 Throughput and latency for different data traffics

Table III-1 Fifteen applications

Application Name	Number of Task Nodes	Number of Cores in Use	Number of Data Packets	Mesh Routing Delay (cycles)	Torus Routing Delay (cycles)	Butterfly Routing Delay (cycles)	HyNoC Routing Delay (cycles)
A2TIME01	7	4	404	14988	14059	6464	4835
AIFFTR01	14	7	640	24832	22912	10880	7680
AIIFFT01	11	5	502	18674	18223	8283	5890
bc	12	8	691	27225	25083	11885	8223
CACHEB01	24	5	1094	50433	42447	20786	13420
cjpeg	25	6	1140	53922	44688	21774	13870
dhNetbench	25	4	1322	71785	53409	26572	16393
djestra	25	4	1143	54635	44920	21946	13945
djpeg	21	7	1058	48245	40839	19785	13154
drrNetbench	21	7	1037	46769	39821	19392	12479
epicUnoptimized Encode	14	6	680	26656	24480	11628	8092
mpegDecode	18	3	926	39355	34540	16853	11112
mpegEncode	23	5	982	42619	37218	18069	11849
pegwitdecode	24	7	1163	56173	45939	22562	14189
pegwitencode	20	4	973	42423	36877	17903	11838

The fourth column of Table III-1 provides the total number of data packets needs to be transmitted among the network. The fifth, sixth, seventh, and eighth column show the routing delay on conventional mesh, torus, butterfly topologies, and the proposed HyNoC architecture, respectively. From Table III-1, it can be observed that the routing delay can be significant saved by using the proposed technology.

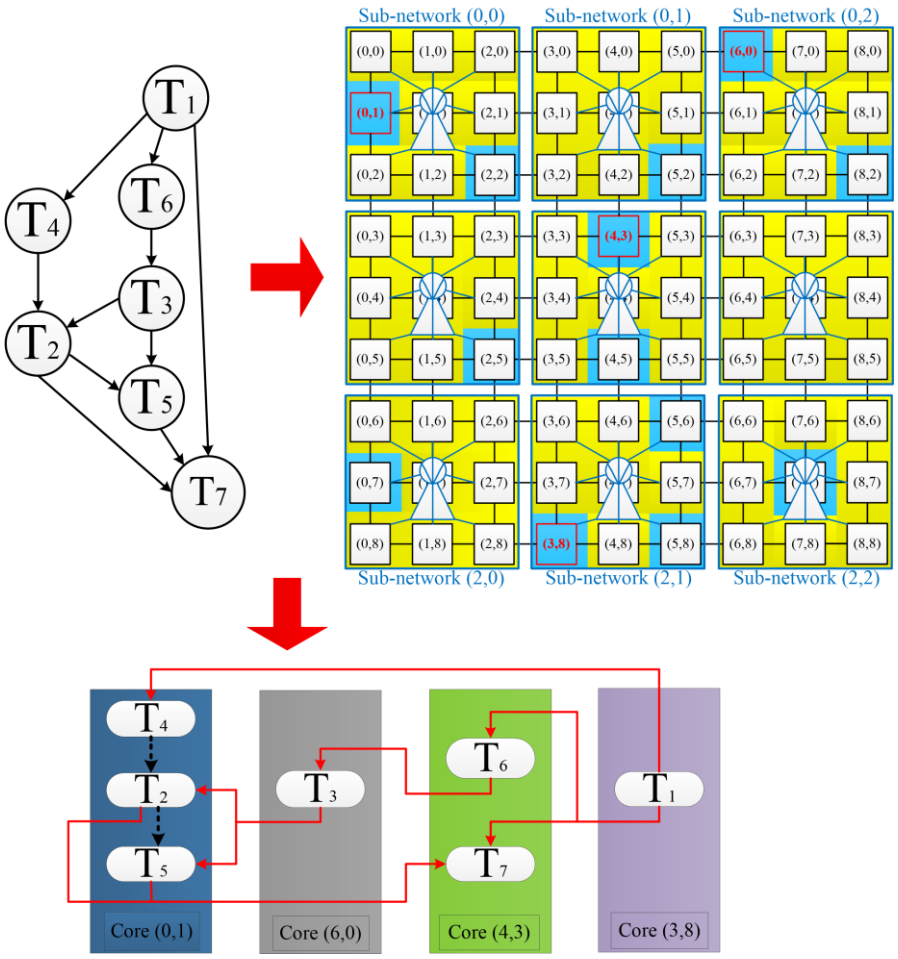


Fig. 3-19 Task flow partitioning and task mapping for the *A2TIME01* application

Fig. 3-20 shows speedups of different network topologies on the fifteen real applications. The speedups of the HyNoC design, torus topology, and butterfly topology are normalized by the speedup of mesh topology. The highest speedup of 4.5 times is achieved on the *dhNetbench* application, which has the most task nodes number of 25 and most inter-core communication data packets of 1322. The lowest speedup of 2.8 times is achieved on the *A2TIME01* application,

which has the least task nodes number of 7 and least inter-core communication data packets of 404. This is because the proposed HyNoC architecture uses four-element-array on-chip antennas to alleviate wired link congestion. The more complicated application with the more data packets transmitted among the network, the more decent performance the proposed technology can provide. For average, the HyNoC design achieves a 3.6 times of speedup over mesh topology as well as a 3.1 times of speedup and a 1.5 times of speedup over torus and butterfly topology. The speedup of the HyNoC design mainly owes to the one hop source-destination communication achieved by the four-element-array on-chip antenna. The task scheduling algorithm balances the utilization of wired and wireless links. The adaptive routing algorithm alleviates data congestion on the wireless network. Some applications have concentrated data patterns, creating hot spots on the network. To avoid traffic building up on wireless links, the adaptive routing algorithm assigns more data packets to the wired network. This lead to a relatively lower speedup compared with other applications with un-concentrated data patterns.

Table III-2 shows the power consumption distribution of the proposed design using TSMC 40nm technology. It can be observed from this table that the major power is consumed by the link circuitry. Wired links consume 59% more power than wireless links. By contrast, power consumed in input buffer, crossbar switch, virtual channel allocator, and switch allocator is much less. In summary, the average power consumption of the HyNoC network is 53.1mW.

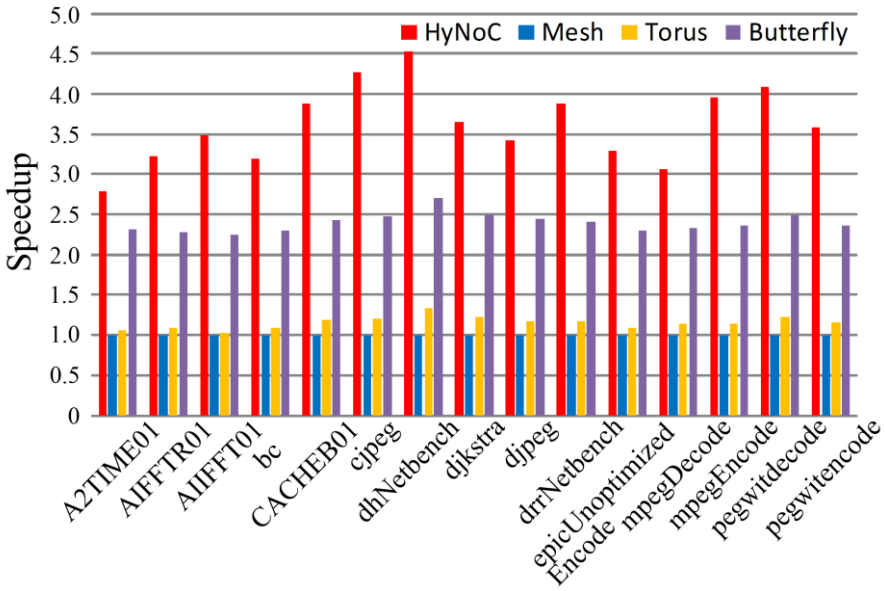


Fig. 3-20 Speedup on real applications

TABLE III-2 Power distribution

Component	Power (mW)
Wireless Link	18.1
Wired Link	28.84
Crossbar Switch	2.27
Input Buffer	3.52
VC Allocator	0.22
Switch Allocator	0.15

Fig. 3-21 shows the total energy comparison between different network architectures on each of the fifteen real applications at 400MHz clock frequency. For different applications with different data traffic patterns, the ratios of energy consumed on wired and wireless links are different for the HyNoC design. The HyNoC architecture has an average energy savings of 4.6X and 3.9X over the mesh and torus topology. These energy savings mainly owe to the low communication hop count achieved by the all-direction millimeter-wave on-chip antenna. With a less number of communication hops, energy consumed on both routers and wired links are greatly saved. The HyNoC architecture has an average energy saving of 48% over the butterfly topology. This saving is mainly due to the low energy wireless network. Furthermore, Fig. 3-22 shows the energy per bit comparison between different architectures on the real applications.

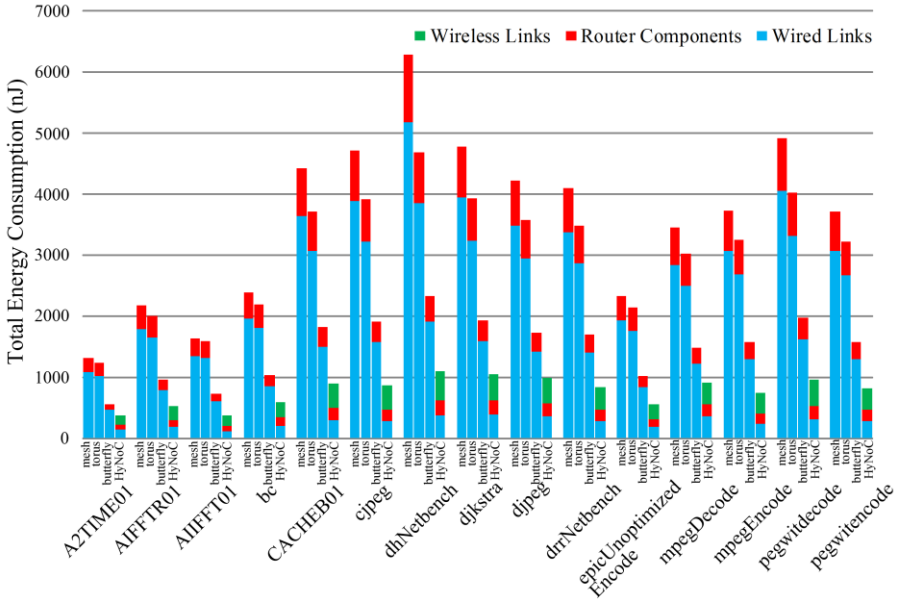


Fig. 3-21 Energy comparison on real applications

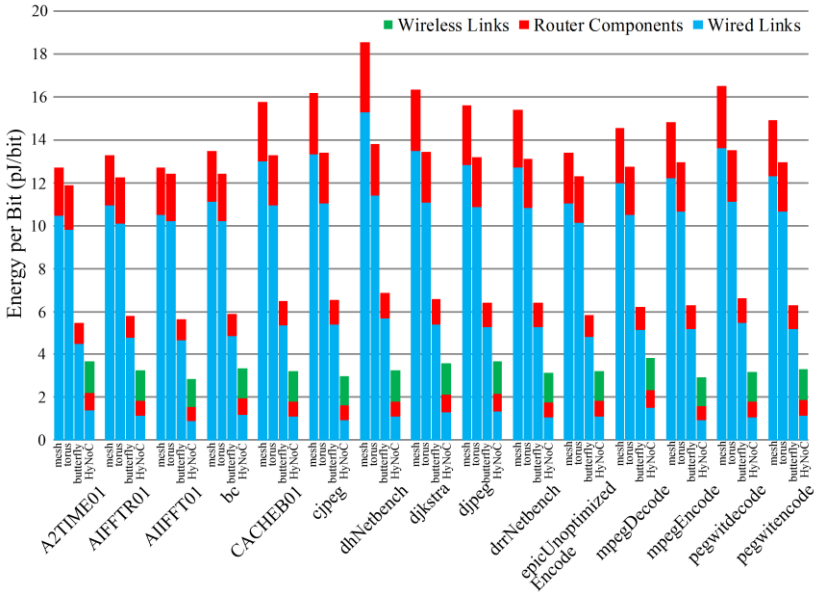


Fig. 3-22 Energy per bit

CHAPTER 4 PROTECTING EMBEDDED SYSTEM WITH OBFUSCATION TO PREVENT SIDE CHANNEL ATTACK

Due to the success in SoC architecture, PDAs, Smart Phones, Tablets, Smart Watches and a lot of other mobile platform based systems are getting more and more popular in all aspects of people's lives. A key component of these gadgets is the embedded system.

4.1 Current State of Art on Embedded System Encryption and Side Channel Attack

Embedded system uses cryptographic algorithms, such as AES, DES, ECC, RSA, SHA [21] [22] [81-83], to encrypt the private data. Among all of these cryptographic algorithms, Advanced Encryption Standard (AES) algorithm is one of the most widely used encryption programs in embedded systems [16] [24]. Fig. 4-1 [84] illustrates the encryption process of a 128-bit AES algorithm. As shown in Fig. 4-1, a 128-bit input is *xor*'ed (denoted as $\oplus$) with a 128-bit secret key to produce a 128-bit *Y*. This *Y* is divided into sixteen 8-bit blocks. Every 4 different 8-bit blocks are combined into a 32-bit block *(FT0*, *FT1*, *FT2* and *FT3*). Lines in different colors are used to indicate the combination of 8-bit blocks, for example *Y0[0]* is fed into *FT0*, *Y1[1]* is fed into *FT1*, *Y2[2]* is fed into *FT2*, and *Y3[3]* is fed into *FT3*. The 32-bit blocks are used as indices for the SBOX lookup. The SBOX lookup result is *xor*'ed with the secret key again to produce the 128-bit output. The whole process will continue for several iterations/rounds.

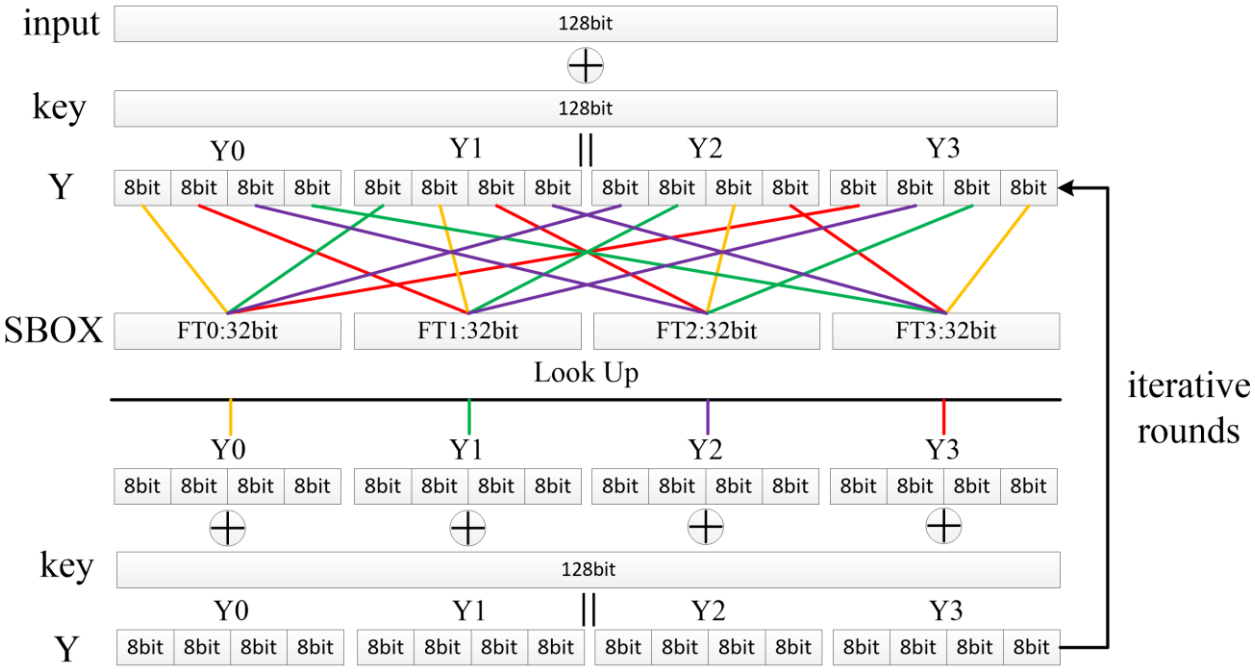


Fig. 4-1 128-bit AES example [84]

With the soaring prosperity of the embedded system, its security becomes a serious concern: attacks may penetrate the embedded system and may obtain the private information. One type of attacks is side channel attack — an attack based on information obtained from the measuring performance of the physical system, instead of identifying theoretical weaknesses in the software or algorithms (e.g. this is done through cryptanalysis). While cryptographic algorithms help embedded system develop secret keys to encode plaintext into ciphertext, they fail to effectively protect embedded system devices from SCAs. With simple measurement equipment or testing schemes, SCAs can monitor the system performance such as dynamic power dissipation,

electromagnetic emission, and processing time of the embedded system. Note that these measurements also include the performance metric when the embedded system is executing an encrypted program [16] [17] [19]. By correlating the performance measurement with the executed program, SCAs can easily predict the secret key and decipher the ciphertext. Among the SCA approaches, power analysis is an efficient and powerful attack. As shown in Fig. 4-2 [84], attacker can easily connect a resistor between the *Vcc* and *Gnd* segments of a smart card to measure the dissipate power. The attacker can deliberately control the clock (*CLK*) and data inputs (*I/O*) to perform power analysis on the encryption program [84].

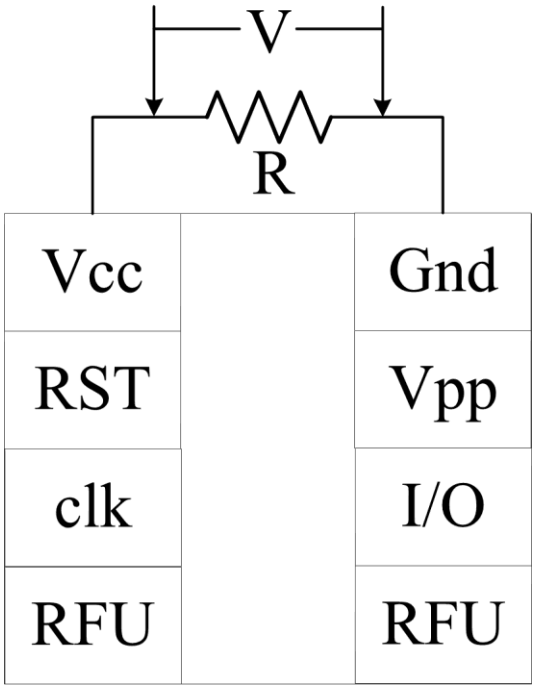


Fig. 4-2 Power analysis on a smart card [84]

There are two different types of power analysis: Simple Power Analysis (SPA) [16] [20] and Differential Power Analysis (DPA) [15] [21-24]. SPA analyzes power profile of the embedded system based on the magnitude of the power value. DPA, regarded as even more powerful than SPA, uses statistical method to predict the secret key of the cryptographic algorithm. Researchers have shown that the once the encryption location is identified from a power profile, SCAs can successfully obtain secret keys and thus break into the embedded system [84-87]. Therefore, even with cryptographic algorithms, embedded systems are still very vulnerable to SCAs.

To prevent power analysis attack, one potential approach is obfuscating power profile to hide the encryption in the program executed on embedded system. Several ideas and approaches were employed by a couple of research groups [11] [14] [85] [88]. Most of these ideas are applicable in some real applications. However, it is not clear how exactly the proposed efforts can decrease the risk from SCAs. Nor did these approaches achieve an optimized obfuscation. For example, *Table Masking* methodology proposed by Gebotys in [85] presents a countermeasure to resist second- and third-order DPA and differential electromagnetic analysis (DEMA). In this methodology, the regular SBOX is divided into several different tables. Each SBOX data uses a different random mask and be randomly masked in these tables. This algorithm shows success for preventing DPA and DEMA attacks; however it doesn't make any modification of the instruction sequence. Therefore, the arrestive repeated patterns in power profile are still distinct. Additionally, this algorithm lacks self-automation, and thus requires a lot

of human interference. Barbosa et al. [88] proposed an automatic process of construct indistinguishable operation sequence. By analyzing execution paths, the algorithm automatically inserts dummy tasks into instruction sequence, creating identical execution paths. This methodology has been proved to defend against SCAs without leaking information on program control flow and computationally identical execution paths. However, the dummy task insertion may be easily removed by using simple time shift [15]. In addition, the fixed insertion location is also unsecure. *Reconfigurable Architecture* introduced in [14] suggested an embedded processor with a reconfigurable functional unit (RFU). Proposed by Seyyedi et al., this algorithm used an automatically generated RFU to randomly execute custom instructions. The obfuscated power profile was compared with the one without obfuscation by a correlation coefficient. This coefficient reveals very little about how the added obfuscation can prevent SCAs. Nor did this methodology achieve an optimized result. Similar issue also occurred in the method introduced in [11]. Ambrose et al. [11] used a random code injection methodology to mask power analysis based SCAs. Random task was automatically generated and injected to a random location. A mathematic formulation called *RIJID* index was defined to evaluate the scrambling provided by the random instruction injection. However, it is not clear how much impact this random insertion would have on SCAs. And this methodology also failed to achieve an optimized result.

4.2 Cross-correlation based Encryption Obfuscation Model

Most popular cryptographic algorithms (such as AES, DES, TripleDES) utilize iterative computations to encrypt data with a secret key. This chapter focuses on protecting AES encryption algorithm on embedded systems against SCAs. As discussed in previous section, AES uses SBOX executing *xor* and *lookup* instructions successively for several iterations. Each iteration executes the same set of instructions. Thus, these iterations would consume similar amount of power. It is obvious that similar patterns of power consumption would repeat in the dynamic power profile [89]. These similar power patterns can be extracted as a power template and be cross-correlated to the power profile to locate the AES encryption. Once the location of encryption in the instruction sequence is identified, SCAs can apply power analysis (such as SPA [16] [20] and DPA [15] [21-24]) to intrude the device [84-87].

This section introduces an obfuscation model based on cross-correlation between power template and power profile. Inputs to this model include power template, power profile with obfuscation, and peak information of cross-correlation between power template and power profile without obfuscation. The output of this model is the *Dissimilarity Level* which indicates how little repeating patterns may be identified in the power profile with obfuscation. Before discussing the new cross-correlation based obfuscation model, let us first review cross-correlation concept, and how to use power template to identify AES location in power profile.

Definition 1 ***Cross-correlation f★g*** *is defined as an integral of the product of a complex conjugate function* $f^*(t)$ *and a function* $g(t+\tau)$*, where* $g(t+\tau)$ *is a time-lag function with time lag* τ [90].

This definition can be further written using the follow equation:

$$(f \star g)(\tau) = \int_{-\infty}^{+\infty} f^*(t) g(t+\tau) dt \tag{4.1}$$

Cross-correlation provides an accurate indication of whether *g* has presence in *f* or not. The amount of correlation between the two functions is indicated by peaks of cross-correlation result. Peak locations reveal the location of *g*'s presence in *f*, and higher peak values indicate a higher correlation between the two functions. Based on (4.1), *Power Profile Cross-correlation* can be defined:

Definition 2 ***Power Profile Cross-correlation:*** *Let P denote the power profile of a circuit or system and* P_T *be the power template of interest.* $P \star P_T$ *is defined as Power Profile Cross-correlation between the power profile and power template.*

$$(P \star P_T)(\tau) = \int_{-\infty}^{+\infty} P^*(t) P_T(t+\tau) dt \tag{4.2}$$

(4.2) is the direct result of using (4.1) on power profile. We can use the similar approach to identify whether power profile with obfuscation still has presence of P_T by replacing P with P_{obf}:

$$(P_{obf} \star P_T)(\tau) = \int_{-\infty}^{+\infty} P_{obf}{}^*(t) P_T(t+\tau) dt \tag{4.3}$$

Here P_{obf} is the power profile with obfuscation. Fig. 4-3 and Fig. 4-4 further explain the physical meaning of these definitions. Fig. 4-3(a) shows a power profile without obfuscation. Fig. 4-3(b) is the power template extracted from this power profile, e.g. the power profile for AES algorithm. Fig. 4-3(c) shows the cross-correlation result of Fig. 4-3(a) and Fig. 4-3(b). Let $X = P \bigstar P_T$ be the cross-correlation function for the power profile without obfuscation. Then X_{peak} represents the highest peak value of X, and $t_{X_{peak}}$ denotes its corresponding time. The time coordinate axis of cross-correlation has already been adjusted to the same coordinate axis as in Fig. 4-3(a). Significant cross-correlation peaks reveal the location of the power template at time $t = t_{X_{peak}}$ in the power profile without obfuscation in Fig. 4-3 (a).

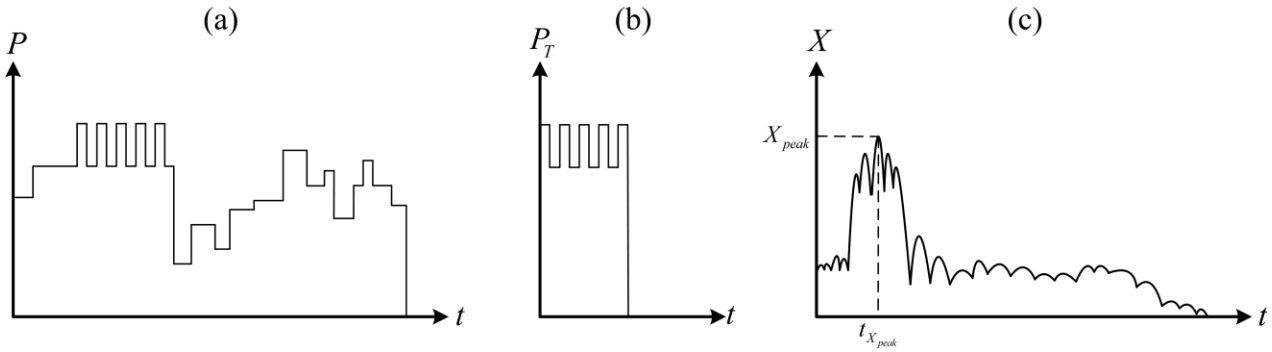


Fig. 4-3 (a) Power profile without obfuscation; (b) Power template; (c) Cross-correlation of (a) and (b)

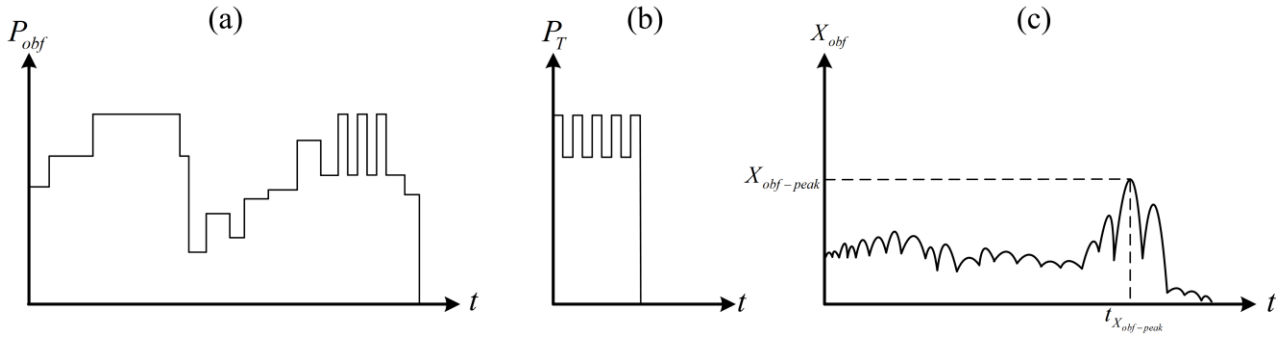


Fig. 4-4 (a) Power profile with obfuscation; (b) Power template; (c) Cross-correlation of (a) and (b)

Fig 4-4(a) shows the power profile with obfuscation (denoted as P_{obf}). Section 4.4 will further discuss two new obfuscation methodologies: *Real Instruction Insertion* and *AES Mimic.* Fig 4-4(b) demonstrates the same power template as Fig 4-3(b). The cross-correlation between power template and new power profile with obfuscation (represented as $X_{obf} = P_{obf} \bigstar P_T$) is shown in Fig 4-4(c). Let $X_{obf\text{-}peak}$ represent the highest peak value of X_{obf}, and $t_{X_{obf-peak}}$ denotes its corresponding time. Apparently, the cross-correlation peaks are shifted to new locations around time $t = t_{X_{obf-peak}}$ when obfuscations are applied. A shift of peak locations indicates the impact of obfuscation. Thus, when the attack applies cross-correlation method, it cannot correctly identify the encryption location and thus cannot discover the secret key. Henceforth, the shifted cross-correlation peaks mislead attacks to some locations other than the real encryption one.

To measure how much dissimilarity between the power profiles with and without obfuscation, a new cross-correlation based obfuscation model is introduced in this chapter. A

concept called *Dissimilarity Level* (denoted as parameter D_L) is introduced to measure and control the optimization process. Fig. 4-5 shows the comparison of *Power Profile Cross-correlation* with and without obfuscation. The dash curves represent X and the solid curves represent X_{obf}. The original peaks in X are reduced in X_{obf}. This indicates that the power template is at low correlation with the obfuscated power profile at this location. New peaks show up at time $t = t_{X_{obf-peak}}$ in X_{obf}. A larger distance *dis* between $t_{X_{obf-peak}}$ and $t_{X_{peak}}$ promises a larger deviation of encryption location identification. In addition, peak value is another critical factor to measure *Dissimilarity Level*. This paper proposes two methodologies to obfuscate the power profile: *Real Instruction Insertion* and *AES Mimic*. For the *Real Instruction Insertion*, random instructions are inserted into the AES *xor&lookup* iteration sequence to prevent SCAs from identifying the AES encryption location. Smaller cross-correlation peak value would help for hiding the encryption location. In this case, the *Dissimilarity Level* is defined as:

$$D_L = \frac{1}{N} \cdot \sum \left[\left| t_{X_{obf-peak}} - t_{X_{peak}} \right| \Big/ \frac{X_{obf-peak}}{X_{peak}} \right] \tag{4.4}$$

where N is total number of peaks of cross-correlation between power template and power profile with obfuscation. The value of each peak is normalized by X_{peak}. D_L is calculated by the average of all peaks. For the *AES Mimic*, imitative *xor&lookup* tasks are created on the parallel hardware. By this methodology, fake "AES encryption" is created to delude SCAs. In this case, higher

cross-correlation peak values are expected for the fake encryption identification. The *Dissimilarity Level* is defined as:

$$D_L = \frac{1}{N} \cdot \sum \left[\left| t_{X_{obf-peak}} - t_{X_{peak}} \right| \cdot \frac{X_{obf-peak}}{X_{peak}} \right] \tag{4.5}$$

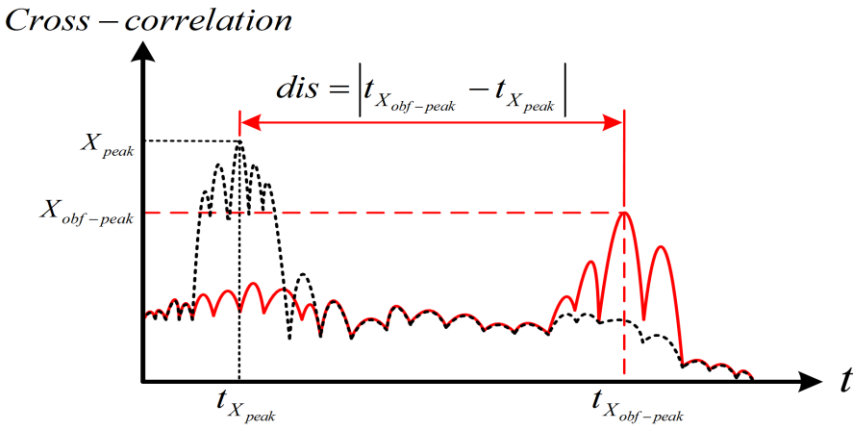


Fig. 4-5 Comparison between X and X_{obf}

4.3 Evaluation Dissimilarity Level Impact on DPA

All SCAs utilize a common hypothesis test procedure to predict the secret key of encryption algorithm [13]. Fig. 4-6 [13] presents this common process flow of DPA attack. Power profiles are measured from embedded systems by SCAs. We can denote the power profiles as $P(S, k_c) = \{P_1, P_2, \ldots, P_n\}$. Power profiles can be represented as deterministic functions of input data set $S = \{s_1, s_2, \ldots, s_n\}$ and correct secret key k_c, where P_i is the power profile with a specific input test data s_i, and n is the total number of input data. Key hypotheses $K = \{k_1, k_2, \ldots, k_m\}$ contains all

possible subkey candidates for DPA. Select function, denoted as $V = \Psi(S, K)$, is a function calculating the intermediate value for each pair of input test data and key candidate.

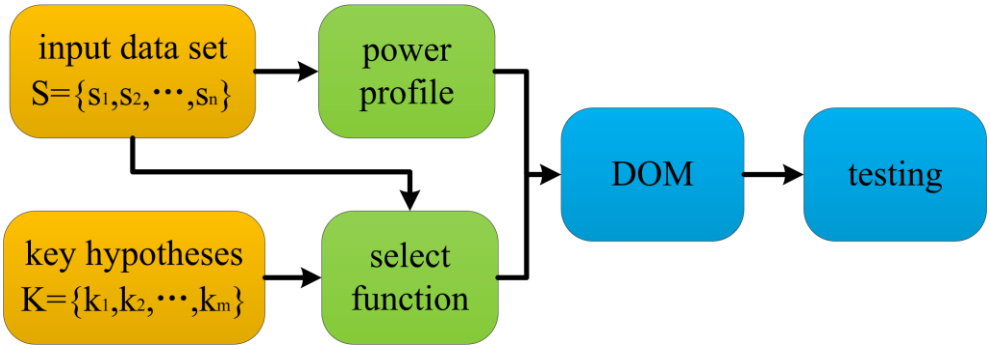


Fig. 4-6 Common procedure of DPA through test [13]

In DPA attack, V is a single bit. It can only be either 1 or 0. For all the input tests, we can model the power distribution using Gaussian model. Thus, power profile can be represented as [13]:

$$P_i = \varepsilon V_{i,c} + c + r_i \tag{4.6}$$

where ε is the unit power consumption, c is an unknown constant, r_i is a random noise from circuitry and measurement following a Gaussian distribution N(0, σ^2), and $V_{i,c}$ is the select function for input data s_i and correct key k_c: $V_{i,c} = \Psi(s_i, k_c)$. Different of means (DOM) between power profile and select function is calculated for each key candidate in DPA attack. For each key candidate k_j, select function with different input data are divided into two groups: $V/k_j = \Psi(S,$

k_j) = 1 and $V/k_j = \Psi(S, k_j) = 0$. DOM is defined as the difference between the average power consumption of these two groups [13]:

$$\delta_j = avg(P_{V=1}) - avg(P_{V=0}) = \frac{\sum P_{V=1}}{N_{V=1}} - \frac{\sum P_{V=0}}{N_{V=0}} \tag{4.7}$$

where $N_{V=1}$ is the total number of power profiles with $V = 1$, $N_{V=0}$ is the total number of power profiles with $V = 0$, and $N_{V=1} + N_{V=0} = n$. n is the total number of input data, which also is the total number of power profiles. Finally, the secret key is predicted in the testing process. With a sufficient number of power profiles, the DOM value (δ_c) of correct key (k_c) would converge to the unit power consumption (ε), while DOM values for other incorrect keys are much smaller [13]:

$$\lim_{n\to\infty} \delta_c = \varepsilon, \quad \lim_{n\to\infty} \delta_j << \varepsilon \quad j \neq c \tag{4.8}$$

In (4.5), D_L is defined as proportional to the distance (*dis*) between real and fake AES encryption. A large D_L value indicates the fake "AES encryption" is far away from the real one. In this case, power profiles are presented as $P_{obf}(S, k_f) = \{P_{obf1}, P_{obf2}, \ldots, P_{obfn}\}$, with

$$P_{obfi} = \varepsilon V_{i,f} + c + r_i \tag{4.9}$$

$$V_{i,f} = \Psi(s_i, k_f) \tag{4.10}$$

where $S = \{s_1, s_2, \ldots, s_n\}$ is the input date set, k_f is the fake secret key of imitative *xor&lookup* tasks in *AES Mimic*. Therefore, a larger D_L promises a larger chance that DPA predicts a fake secret key.

Another efficient approach to prevent DPA attack is increasing the number of power profiles needed to predict the correct key. More number of power profiles needed means the attack needs to perform more number of measurement trials. If n is the number of power profiles needed before obfuscation and n' is the number of power profiles needed after obfuscation. The attack needs to pay n'/n times of additional effort to predict the correct key. In (4.4), D_L is defined inversely proportional to the peak values of cross-correlation between power template and power profile after *Real Instruction Insertion*. Therefore, a higher D_L value appears with more inserted instructions, providing a lower unit power consumption. The *DOM* value of correct key would converge to the unit power consumption with a sufficient number of power profiles. Although an individual power profile is a deterministic waveform, many power profiles present a Gaussian distribution. The relationship between the number of power profiles needed to predict the correct key and unit power consumption also follows Gaussian distribution. If the unit power consumption decreases, the number of profiles needed increases. Therefore, a larger D_L promises more power profiles needed which leads to more effort needed for SCAs to predict the correct secret key in DPA attack.

4.4 Optimization for Power Obfuscation using Dissimilarity Level

This section proposes two optimization methodologies to obfuscate the power profile. Both of these two methodologies utilize the cross-correlation based obfuscation model to promise an optimal result.

4.4.1 Real Instruction Inserting

As AES algorithm uses SBOX to repeatedly execute *xor&lookup* instructions, many encryption algorithms run iterative instructions to cipher a secret data. This leaves a distinct pattern in the power profile. By randomly (random instructions at random locations with random input data) insert real instructions into the encryption instruction sequence, the power profile can be scrambled and the encryption location can be disguised. Shown in Fig. 4-7, two flags, *AES_start* and *AES_end*, are defined to monitor the starting and ending of AES encryption. The *Real Instruction Insertion* algorithm is only applied during the AES encryption process. The algorithm picks a random location at each time, inserts a real instruction into the encryption instruction sequence. The instruction type is randomly selected. Its input data are also randomly generated. Although randomness is employed, the *Real Instruction Insertion* algorithm can reach a locally optimized result by applying the cross-correlation based obfuscation model. The algorithm only inserts one instruction into the encryption instruction sequence at a time. After insertion, the algorithm uses the obfuscation model to calculate D_L value for the new power profile. If the new D_L value is larger than the previous one, the inserted instruction is taken. Otherwise, the inserted instruction is discarded. The algorithm continues generating and inserting

instructions one by one into the encryption instruction sequence, until constraints are violated. The *Real Instruction Insertion* methodology can scramble the iterative power patterns of the encryption algorithm; however these inserted real instructions also consume energy and take up execution time. With more instructions inserted into the encryption instruction sequence, larger timing and energy overheads appear.

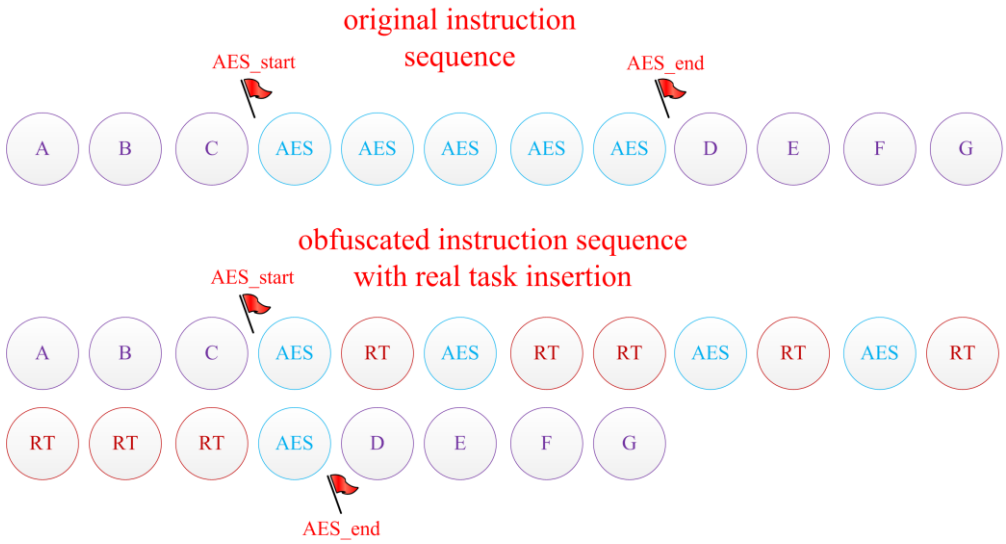


Fig. 4-7 A-G: Non-AES instructions; AES: AES encryption instructions; RT: Inserted real instructions

4.4.2 AES Mimic

The *Real Instruction Insertion* methodology is used to scramble the iterative power patterns of the AES encryption. This methodology can hide the AES encryption location, creating difficulty for SCAs to identify the AES encryption. Besides *Real Instruction Insertion*, another

methodology called *AES Mimic* further obfuscates the power profile. When the instruction sequence is executed with its non-AES instructions, imitative *xor&lookup* tasks are created and executed on parallel hardware. A fake secret key is used in these tasks. In [91], the *Confusion Coefficient* k is defined to indicate the distinguishable level between two different keys under DPA attack:

$$k(k_i, k_j) = P_r[(V / k_i) \neq (V / k_i)] = \frac{N_{(V/k_i)\neq(V/k_i)}}{n} \tag{4.11}$$

where $N_{(V/k_i)\neq(V/k_i)}$ is the number of power profiles for which key candidate k_i and k_j result in different selection function values. n is the total number of power profiles measured. Lager $k(k_i,k_j)$ indicates that it is easy to distinguish keys k_i and k_j from DPA attack. Although the fake secret key is randomly generated in each trial, it must have a large *Confusion Coefficient* with the real AES secret key.

The imitative *xor&lookup* tasks would provide iterative patterns in power profile similar as the AES encryption, misleading SCAs to a wrong location. Because of the instruction sequence and imitative *xor&lookup* tasks running on different parts of hardware parallel and simultaneously, this *AES Mimic* methodology only creates energy overheads, but has no timing overhead.

Similar with *Real Instruction Insertion*, at each time, one *xor&lookup* task is created and a new D_L value is calculated to decide whether this *xor&lookup* task is taken or not. This process stops when a maximum mimic task number is reached or the energy overhead constraint is

violated. Fig. 4-8 shows an example of *AES Mimic* methodology. Notice that the execution time of the mimic tasks is not constant. These tasks can be created and executed at any random time outside of the [*AES_start*, *AES_end*] interval.

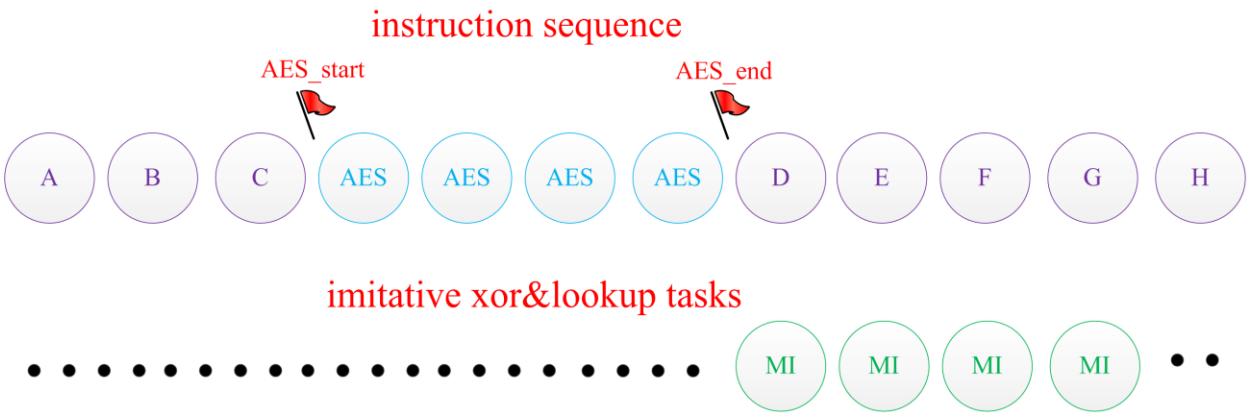


Fig. 4-8 A-H: Non-AES instructions; AES: AES encryption instructions; MI: Imitative *xor&lookup* tasks

4.4.3 Obfuscation Optimization

For the Real *Instruction Insertion* methodology, the instruction type, input data, and inserted location are randomly selected each time. For the *AES Mimic* methodology, the execution time of imitative *xor&lookup* tasks is also random (as long as it's not the same time with real AES encryption execution). Therefore, it can be ensured that the power profiles with obfuscation look different every time, even the same input data set is provided in each measurement trial.

With the cross-correlation based model, power profile obfuscation can be mathematically included into optimization. Along with the timing overhead constraint T_{max} and energy overhead constraint E_{max}, which limiting the number of inserted instructions and imitative *xor&lookup* tasks, the problem of obfuscation optimization can be described as to find P_{obf} while

$$\textit{maximize}: \quad D_L(P_{obf})$$

$$\textit{subject to}: \quad T_{overhead} \le T_{max} \ ; \ E_{overhead} \le E_{max}$$

where P_{obf} is power profile with obfuscation. D_L can be calculated by (4.2) to (4.5). Because P_{obf} is a nonlinear function with regard to time. $D_L(P_{obf})$ is also nonlinear. The energy and timing constraints may be modeled as linear constraints. However, this is a non-linear optimization problem with computation complexity as NP-hard. To resolve this high complexity issue, this chapter proposes an algorithm for obfuscation optimization that is based on simulation with heuristic.

Algorithm 4.1 describes the pseudocode for the obfuscation optimization. When a power template and power profile without obfuscation is available, the cross-correlation between them is calculated first. 'Peak_Extraction' operator extracts peak information from the cross-correlation result. Random instructions are generated by 'Instruction_Generation' operator and randomly inserted into the encryption instruction sequence by 'Instruction_Insertion' operator. Imitative *xor&lookup* tasks are created by "Mimic_Generation" operator. Timing/energy overheads are computed by 'Compute_Time/EnergyOverhead' operator, and used to control the algorithm working under boundaries. 'Compute_D_L' operator uses the cross-correlation based

obfuscation model to calculate D_L value and decides whether the inserted instruction or mimic task is taken or discarded. The computation complexity depends on the number of sampling points on power profiles. Give *Ns* as the maximum number of sampling points on power profile. Peak_Extraction( ) and Compute_D_L( ) are the dominant procedures. Their computation complexity is O(Ns^2). Thus the total computation complexity for Algorithm 4.1 is O(Ns^2).

Algorithm 4.1 Obfuscation Optimization

Input: power profile without obfuscation (P); power template (P_T)*;*

timing overhead constraint (T_{max}); energy overhead constraint (E_{max}); maximum mimic task number (C_{max})

Output: power profile with obfuscation (P_{obf}); D_L value

1: $[X_{peak},\ t_{X_{peak}}] \Leftarrow$ Peak_Extraction($P \bigstar P_T$)

2: $D_L = 0$

3: $C_{mimic} = 0$

4: **while**(1) **do**

5: $Instru_i \Leftarrow$ Instruction_Generation( )

6: $P_{temp} \Leftarrow$ Instruction_Insertion($Instru_i$, P)

7: $T_{overhead} \Leftarrow$ Compute_TimeOverhead(P_{temp})

8: $E_{overhead} \Leftarrow$ Compute_EnergyOverhead(P_{temp})

9: **if** $T_{overhead} > T_{max}$ *or* $E_{overhead} > E_{max}$ **do**

10: **break**

11: **end if**

12: $D_{Ltemp} \Leftarrow$ Compute_$D_L(P_{temp}, X_{peak}, t_{X_{peak}})$

13: **if** $D_{Ltemp} > D_L$ **do**

14: $D_L \Leftarrow D_{Ltemp}$

15: $P \Leftarrow P_{temp}$

16: **end if**

17: **if** $C_{mimic} > C_{max}$ **do**

18: **continue**

19: **end if**

20: $P_{temp} \Leftarrow$ Mimic_Generation(P)

21: $T_{overhead} \Leftarrow$ Compute_TimeOverhead(P_{temp})

22: $E_{overhead} \Leftarrow$ Compute_EnergyOverhead(P_{temp})

23: **if** $T_{overhead} > T_{max}$ *or* $E_{overhead} > E_{max}$ **do**

24: **break**

25: **end if**

26: $D_{Ltemp} \Leftarrow$ Compute_$D_L(P_{temp}, X_{peak}, t_{X_{peak}})$

27: **if** $D_{Ltemp} > D_L$ **do**

28: $D_L \Leftarrow D_{Ltemp}$

29: $P \Leftarrow P_{temp}$

30: C_{mimic} ++

31: **end if**

32: **end while**

33: $P_{obf} \Leftarrow P$

34: **return** P_{obf}, D_L

4.5 Result Analysis

To evaluate the proposed encryption obfuscation model and optimization methodologies, experimental results presented in this section are based on simulations.

4.5.1 Experimental Setup

As a test example, a simple instruction sequence shown in Fig. 4-9 is generated. All of the instructions are fine grain and their input and output data are 128-bit. A 128-bit AES encryption is applied in this experiment. Fig. 4-9 also presents the DAG of this instruction sequence.

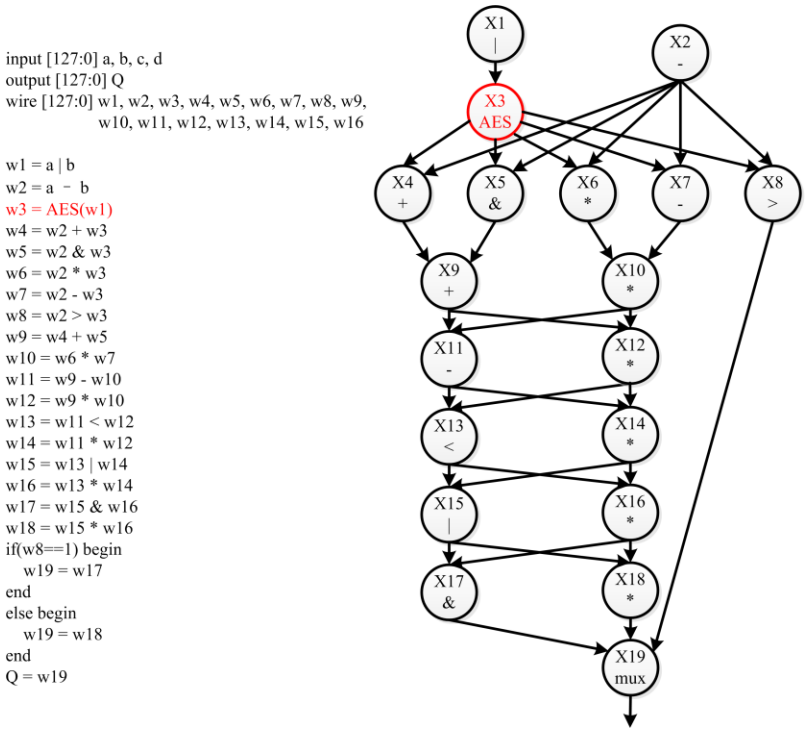


Fig. 4-9 Instruction sequence example

Fig. 4-10 shows the experiment process of task generation, power profile obfuscation, and D_L calculation. The Task Generator is implemented in C and compiled using Microsoft Visual Studio®. The test instruction sequence and inserted real instructions/mimic tasks are implemented in Verilog. All Verilog modules are simulated and synthesized on Xilinx ISE Design Suite 13.2®. The embedded device used in the experiment is the *Virtex-6*® family of Xilinx FPGA [49-51]. Device details are described in TABLE IV-1. The runtime of each execution is also measured using Xilinx simulation tool. The power profile is measured by

Xilinx Power Analyzer®. The encryption obfuscation model is implemented in Matlab®. D_L value and timing/energy overheads are also calculated by using Matlab®.

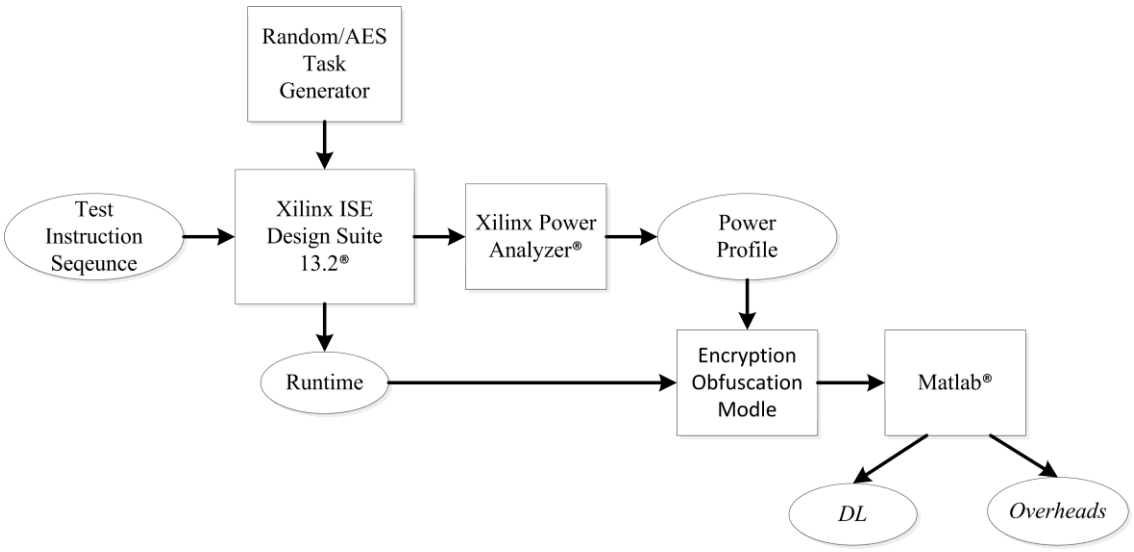


Fig. 4-10 Experiment process of the proposed experiments

TABLE IV-1 Device details

Family	Virtex6
Part	xc6vlx760
Package	ff1760
Grade	Commercial
Process	Typical
Speed Grade	-2
Characterization	Production,v1.3,2011-05-04
Frequency	180MHz

4.5.2 Obfuscation and DPA

Fig. 4-11 shows the power profile without obfuscation and the cross-correlation between this power profile and power template for the target instruction sequence. Peaks of the cross-correlation are located around time t = 22 ns. Which indicates the AES encryption with secret key occurs at time t = 22 ns in the power profile. Fig. 4-12(a) shows the power profile of target instruction sequence after applying the proposed obfuscation methodologies. The energy and timing overhead boundary for *Real Instruction Insertion* is set as 15% and 25% respectively. The energy overhead boundary for *AES Mimic* is set as 30%. Fig. 4-12(b) is the cross-correlation between this obfuscated power profile and power template. Around time t = 22ns, the peak values are decreased to 9 units, which are around 15 units in Fig. 4-11(b). The decreasing of peak values at the AES encryption location proves the efficiency and effectiveness of the proposed methodologies for hiding the AES encryption in an instruction sequence. In addition, the new significant peaks in Fig. 4-12(b) appear around time t = 127 ns, with values around 21 units. This would mislead SCAs to a wrong location with wrong power profiles [13]. As a result, a fake secret key, which has a large confusion coefficient with the real secret key [91], would be predicted by DPA attack.

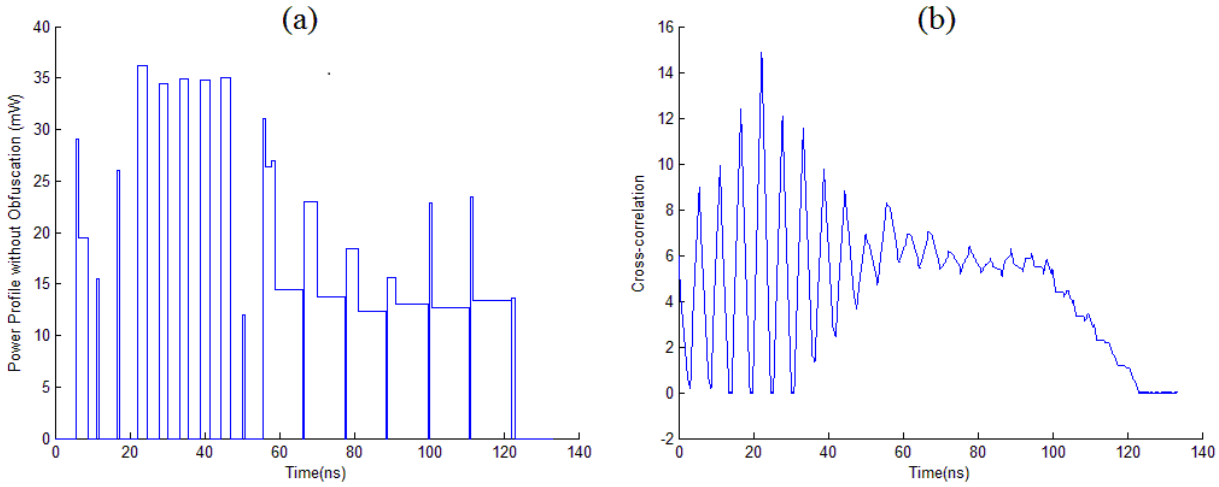


Fig. 4-11 Power profile without obfuscation and cross-correlation

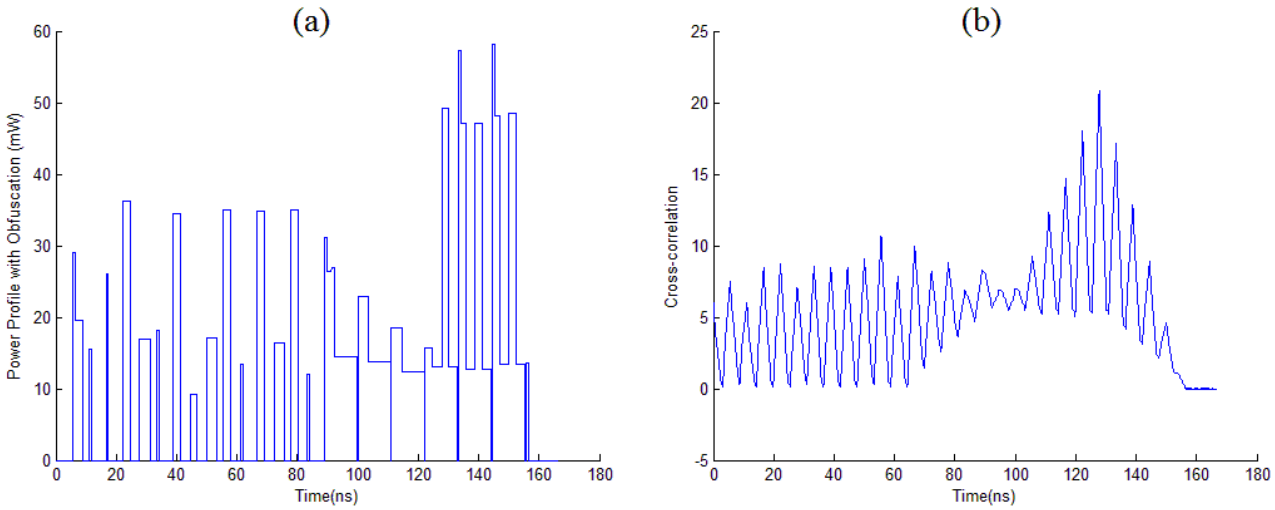


Fig. 4-12 Power profile with obfuscation and cross-correlation

The proposed methodologies also decrease the unit power consumption of the real AES encryption. Unit power consumption ε of AES encryption is 17.17 mW for the power profile without obfuscation. After applying the proposed power obfuscation methodologies, $\varepsilon' = 11.06$ mW. The relationship of unit power consumption and power profiles needed in DPA to predict the secret key is a Gaussian distribution. According to the Gaussian distribution table, it can be calculated that after applying the proposed power obfuscation methodologies, the DPA attack needs 10 times of additional number of power profiles to predict the secret key. This analysis demonstrates that even if SCAs somehow correctly identify the real AES encryption location in the instruction sequence, it becomes 10 times harder for DPA to predict the correct secret key after applying the proposed methodologies.

To demonstrate the effectiveness of the proposed obfuscation model and optimization methodologies, we present five test cases in this paper. TABLE IV-2 depicts the detail data information of the five test cases before and after applying the proposed power obfuscation technology. The second and third columns of TABLE IV-2 list the average power consumptions and execution time of the five test cases before the obfuscation process. The fourth and fifth column list the average power consumptions and execution time of these test cases after applying the proposed *Real Instruction Insertion* and *AES Mimic* methodologies. The sixth column shows the final D_L value for the test cases. The last column indicates whether DPA attack is successful or failed.

TABLE IV-2 Power obfuscation with proposed technology

Test Case	Before Obfuscation		After Obfuscation			
	Avg. Power Consumption (mW)	Execution Time (ns)	Avg. Power Consumption (mW)	Execution Time (ns)	D_L Value	DPA Success Or Fail
A	10.43	88.90	12.20	105.56	46.15	Fail
B	12.56	122.23	14.30	150.01	100.70	Fail
C	12.44	133.34	13.68	166.68	103.51	Fail
D	11.53	133.34	12.54	166.68	114.23	Fail
E	11.38	150.01	12.89	183.35	133.42	Fail

For the five test cases, A has the least number of instructions. The average power consumption and execution time for A is much lower than which for B, C, D and E. Therefore, under a same percentage of energy and timing overhead constraint, A has much less space for real instruction insertion and imitative *xor&lookup* task creating. As a result, the final D_L value for test case A is only 46.15, which is much lower than the D_L values of test cases B to E. However, the DPA attack on test case A is still failed due to the wrong AES encryption location identification.

To evaluate the individual contribution of the proposed *Real Instruction Insertion* methodology and *AES Mimic* methodology, two separate experiments are fulfilled on the test cases. TABLE IV-3 depicts the energy overheads, D_L value, and fake secret key prediction for the test cases after applying *AES Mimic* methodology. For *AES Mimic* methodology only, there is no timing overhead. The energy overhead constraint is set as 30% for the optimization process.

The fourth column of TABLE IV-3 indicates whether the fake encryption key is predicted by DPA attack. For test case B to E, large D_L values are achieved. There are enough imitative *xor&lookup* tasks to mislead the attack to a fake encryption location with a fake secret key prediction. However for test case A, the D_L value is much smaller. There are not enough imitative *xor&lookup* tasks to mislead the attack. The real AES encryption location can still be identified and the real secret key can be predicted by DPA.

TABLE IV-3 Power obfuscation with *AES Mimic*

Test Case	Energy Overheads (%)	D_L Value	Fake Key Prediction
A	25.46	21.03	No
B	27.92	64.10	Yes
C	23.10	62.12	Yes
D	27.78	91.89	Yes
E	24.52	93.41	Yes

TABLE IV-4 depicts the energy and timing overheads, D_L value, and additional effort of DPA attack for the test cases after applying *Real Instruction Insertion* methodology. The constraints of energy and timing overheads for the optimization process are set as 15% and 25% respectively. The *Real Instruction Insertion* methodology can create more difficulties for DPA attack by decreasing the unit power consumption. The fifth column in TABLE IV-4 illustrates the number of times of additional efforts for DPA attack on the test cases.

TABLE IV-4 Power obfuscation with *Real Instruction Insertion*

Test Case	Energy Overheads (%)	Timing Overheads (%)	D_L Value	Additional Effort (# of times)
A	14.93	25	26.35	6.67
B	11.84	22.73	13.01	7.34
C	11.61	25	11.53	9.50
D	14.70	25	57.34	8.42
E	13.89	22.22	50.73	7.78

By averaging the test cases, Fig. 4-13 shows the percentage of D_L value and energy overheads taking up by the two power obfuscation methodologies. Apparently, comparing with *Real Instruction Insertion*, *AES Mimic* consumes the major part of energy (65.79%) but also provides a major of D_L value (67.66%).

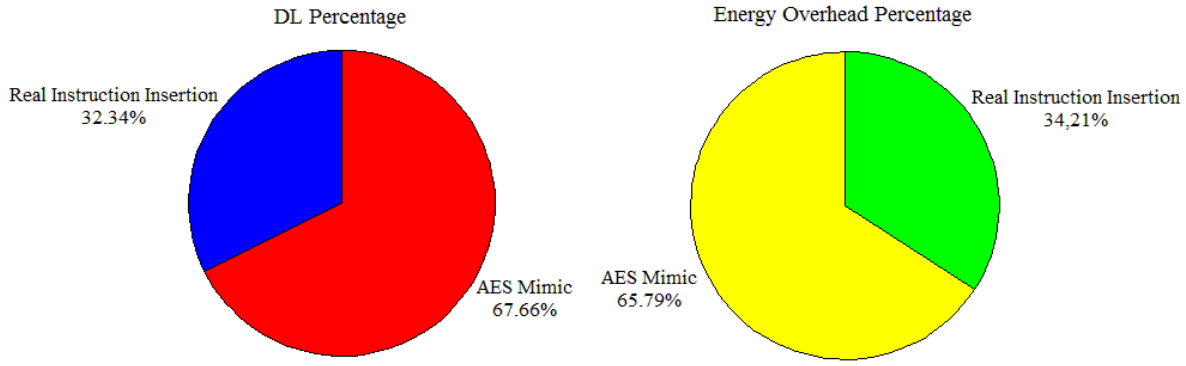


Fig. 4-13 D_L and energy overhead percentage for *Real Instruction Insertion* and *AES Mimic*

CHAPTER 5 CONCLUDING REMARKS

High performance computing architecture needs adoptable network-on-chip to provide efficient and reliable on-chip communication with a large number of communication channels. It also requires sophisticated security mechanism to protect the system from side channel attacks. This dissertation proposes efficient and reliable NoCs for high performance computing architectures by targeting at novel NoC router microarchitecture and hybrid NoC topology. A power obfuscation model to prevent SCA-DPA is also described in this dissertation.

In Chapter 2, by applying the virtual collision array concept to multi-core system NoC architecture, we propose a new NoC router architecture to reduce the number of sequential data access to router pipeline. To facilitate the new architecture, we provide a new workload assignment and task scheduling. Through the new algorithms, we achieve minimal router delay while considering performance constraints and system constraints, e.g. power constraints and timing constrains. We demonstrate the potentials of using the new design to ease the exiting data I/O limitations, routing resource contentions, as well as communication bottlenecks in the current multi-core systems.

In Chapter 3, the proposed a 9x9 HyNoC architecture uses only nine sets of four-element-array antennas to provide wireless communication. The number of antennas used on the NoC platform is reduced. Thus budgets in both cost and area are saved. The new proposed adaptive routing algorithm takes data packet current location and destination, wired transmission hop

count, buffer utilization including traffic jams, and network energy consumption into consideration. It outputs routing decision of wireless or wired fabrics. The adaptive routing algorithm can dynamically choose a proper transmission path, wired or wireless, for each data packet during the whole routing process.

In Chapter 4, two obfuscation ideas *Real Instruction Insertion* and *AES Mimic* are introduced. The first one reshapes the power profile of the embedded system. The later one intends to mislead SCAs by using a fake secret key. It creates power profile that resembles the impact of this wrong key. A new concept referred to as *Dissimilarity Level* is introduced to provide an explicit assessment of power profile difference between the one without obfuscation and the one with obfuscation. For the first time, we show the impact of the proposed obfuscation methods on SCAs in terms of the amount of effort in identifying the correct secret keys. Our methods consider the additional overheads of the obfuscation method through optimization. We maximize *Dissimilarity Level* using *Real Instruction Insertion* and *AES Mimic* under performance constraints (e.g. energy constrains and timing constraints).

REFERENCES

[1] G. Moore, "Excerpts from a conversation with Gordon Moore: Moore's law," *Video Transcript, Intel, 54*, 2005.

[2] P. Gelsinger, "Moore's Law–The Genius Lives On," *Solid-State Circuits Society Newsletter, IEEE, 11*(5), 18-20, 2006.

[3] Hitex, "Using Multicore Processors in Embedded Systems," [Online] http://www.hitex.com/fileadmin/pdf/news/articles/multicore-processors-in-embedded-systems.pdf

[4] M. Levy and T. M. Conte, "Embedded multicore processors and systems," *IEEE micro 3: 7-9*, 2009.

[5] "Multicore processors: Providing opportunities for embedded systems designers", *Embedded Computing Design*, Oct. 2008, [Online] http://embedded-computing.com/article-id/?3625=

[6] A. Melnichuk, "Multi-processor Embedded Systems," [Online] http://www.ann.ece.ufl.edu/courses/eel6935_11fal/slides/Multi-Processor%20Embedded%20Systems.pdf

[7] C. Demerjian, "A look at a 100-core Tilera Core Gx. Efficiency," *Microprocessor and Servers*, Oct. 2009, [Online] http://semiaccurate.com/2009/10/29/look-100-core-tilera-gx/

[8] R. Schooler, "Tile processors: Many-core for embedded and cloud computing," *Workshop on High Performance Embedded Computing*, 2010.

[9] L.-S Peh, S. W. Keckler, and S. Vangal, "On-chip networks for multicore systems," *Multicore Processors and Systems, pp. 35-71, Springer US*, 2009.

[10] J. D. Owens, W. J. Dally, R. Ho, D. N. J. Jayasimha, S. W. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *IEEE micro*, (5), 96-108, 2007.

[11] J. A. Ambrose, R. G. Ragel and S. Parameswaran, "A smart random code injection to mask power analysis based side channel attacks," *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 5th IEEE/ACM/IFIP International Conference on,* pp. 51-56, IEEE, 2007.

[12] J. A. Ambrose, R. G. Ragel and S. Parameswaran, "Randomized Instruction Injection to Counter Power Analysis Attacks," *ACM Transactions on Embedded Computing Systems (TECS)*, *11*(3), 69, 2012.

[13] Y. Fei, A. A. Ding, J. Lao, and L. Zhang, "A Statistics-based Fundamental Model for Side-channel Attack Analysis," *IACR Cryptology ePrint Archive*, 152, 2014.

[14] S. A. Seyyedi, M. Kamal, H. Noori, and S. Safari, "Securing Embedded Processors against Power Analysis based Side Channel Attacks using Reconfigurable Architecture," *Embedded and Ubiquitous Computing (EUC), IFIP 9th International Conference on,* pp. 255-260, IEEE, 2011.

[15] C. Clavier, J.-S. Coron, and N. Dabbous, "Differential Power Analysis in the Presence of Hardware Countermeasures," *Cryptographic Hardware and Embedded Systems—CHES,* pp. 252-263, Springer Berlin Heidelberg, 2000.

[16] S. Mangard, "A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion," *Information Security and Cryptology—ICISC,* pp. 343-358, Springer Berlin Heidelberg, 2002.

[17] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," *Smart Card Programming and Security*, pp. 200-210, Springer Berlin Heidelberg, 2001.

[18] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," *Cryptographic Hardware and Embedded Systems—CHES,* pp. 251-261, Springer Berlin Heidelberg, 2001.

[19] D. Brumley and D. Boneh. "Remote timing attacks are practical," *Computer Networks*, *48*(5), 701-716, 2005.

[20] R. Mayer-Sommer. "Smartly analyzing the simplicity and the power of simple power analysis on smartcards," *Ches '00*, pp. 78-92, London, UK, 2000.

[21] J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," *CHES*, pp. 292-302, 1999.

[22] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *DPA First Article*, 1998.

[23] G. Boracchi and L. Breveglieri, "A Study on the Efficiency of Differential Power Analysis on aes S-Box," *Technical Report*, 2007.

[24] Y. Han, X. Zou, Z. Liu, and Y. Chen, "Improved differential power analysis attacks on aes hardware implementations," *WiCom '07*, pp. 2230-2233, 2007.

[25] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *Int. Workshop on Cryptographic Hardware & Embedded Systems*, pp. 135-152, 2004.

[26] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," *Int. Workshop Cryptographic Hardware & Embedded System*, pp. 426-442, 2008.

[27] T. H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J.L. Lacoume, "A proposition for correlation power analysis enhancement," *Int. Workshop on Cryptographic Hardware & Embedded Systems*, pp. 174-186, 2006.

[28] K. Chang, S. Deb, A. Ganguly, X. Yu, S. P. Sah, P. P. Pande, B. Belzer, and D. Heo, "Performance evaluation and design trade-offs for wireless network-on-chip architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 8(3), 23, 2012.

[29] P. Wettin, A. Vidapalapati, A. Gangul, and P. Pratim Pande, "Complex network-enabled robust wireless network-on-chip architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 9(3), 24, 2013.

[30] Y. Ye, J. Xu, X. Wu, W. Zhang, W. Liu, and M. Nikdast, "A Torus-Based Hierarchical Optical-Electronic Network-on-Chip for Multiprocessor System-on-Chip," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 8(1), 5, 2012.

[31] M. J. Cianchetti and D. H. Albonesi, "A low-latency, high-throughput on-chip optical router architecture for future chip multiprocessors," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 7(2), 9, 2011.

[32] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *Proceedings of the spring joint computer conference*, pp. 483-485, ACM, 1967.

[33] D. P. Rodgers, "Improvements in multiprocessor system design," *ACM SIGARCH Computer Architecture News*, vol. 13, no. 3, pp. 225-231, IEEE Computer Society Press, 1985.

[34] W. J. Dally, and B. P. Towles, "Principles and practices of interconnection networks," Elsevier, 2004.

[35] V. Thiyagarajan, "Multicore CPUs and the Concurrency changes they bring," *IBM Developerworks*, Jul. 2012.

[36] M. Kanellos, "New life for Moore's Law," *Industry experts predict the near and distant future of chip technology, CNET*, Apr. 2005.

[37] B. Schauer, "Multicore Processors - A Necessity," Sep. 2008.

[38] P. E. McKenney, "Is Parallel Programming Hard, And, If So, What Can You Do About It?" *IBM Beaverton*, Jan. 2011.

[39] S. H. Fuller and L. I. Millett, "The Future of Computing Performance: Game Over or Next Level?" *The National Academies Press*, 2011.

[40] D. Hendler, N. Shavit, and L. Yerushalmi, "A Scalable Lock-free Stack Algorithm," *Proc. Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 806-215, 2004.

[41] N. Shavit, "Data Structures in the Multicore Age," *Communications of the ACM*, vol 54, issue 3, pp 76-84, 2011.

[42] H. Rewini, T. G. Lewis, and H. H. Ali, "Task Scheduling in Parallel and Distributed Systems," *Prentice Hall*, 1994.

[43] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235-1245, 1987.

[44] J. Sun, R. Lysecky, K. Shankar, A. Kodi, A. Louri, and Jm Roveda, "Workload assignment considering NBTI degradation in multicore systems," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 10(1), 4, 2014.

[45] William J. Dally, "BookSim Interconnection Network Simulator," 2013, [Online] http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki

[46] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems," *Proceedings of MICRO*, pp. 330-335, 2008.

[47] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," *Proceedings of IEEE International Workshop on Workload Characterization*, pp. 3–14, 2001.

[48] G. Memik, W. H. Mangione-Smith, and W. Hu, "NetBench: A benchmarking suite for network processors," *Proceedings of ICCAD*, pp. 39–42, 2001.

[49] Xilinx, 2011, [Online] http://www.xilinx.com/products/design-tools/ise-design-suite.html

[50] Xilinx, 2011, [Online] http://www.xilinx.com/products/design_tools/logic_design/verification/xpower_an.htm

[51] Xilinx, DS150, X. D. S., Virtex-6 Family Overview, 2011.

[52] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles and W. J. Dally, "BookSim 2.0 User's Guide," 2013, [Online] http://nocs.stanford.edu/cgi-bin/trac.cgi/raw-attachment/wiki/Resources/BookSim/manual.pdf

[53] G. Reehal and M. Ismail, "A Systematic Design Methodology for Low-Power NoCs," *IEEE Transactions on Very Large Scale Integration (VLSI)*, vol. 22, no. 12, pp. 2585-2595, Dec. 2014.

[54] A. Kumar, L. S. Peh, P. Kundu, and N. K. Jha, "Toward ideal on-chip communication using express virtual channels," *IEEE micro*, vol. 28, no. 1, pp. 80-90, Feb. 2008.

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

[56] ITRS 2007, [Online] http://www.itrs.net/Links/2007ITRS/Home2007.htm

[57] C. Wang, W. Hu, and N. Bagherzadeh, "A wireless network-on-chip design for multicore platforms," *19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pp. 409–416, 2011.

[58] D. DiTomaso, A. Kodi, D. Matolak, S. Kaya, S. Laha, and W. Rayess, "Energy-efficient adaptive wireless NoCs architecture," *7th IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pp. 1-8, 2013.

[59] P. Chiang, S. Woracheewan, C. Hu, L. Guo, R, Khanna, J. Nejedlo, and H. Lui, "Short-range, wireless interconnect within a computing chassis: Design challenges," *IEEE Design and Test of Computers*, vol. 27, no. 4, pp. 32–43, 2010.

[60] S. Lee, S. Tam, I. Pefkianakis, S. Lu, M. Chang, C. Guo, G. Reinman, C. Peng, M. Naik, L. Zhang, and J. Cong, "A scalable micro wireless interconnect structure for CMPs," *Mobicom*, pp. 217–228, 2009.

[61] W. Green, M. Rooks, L. Sekaric, and Y. Vlasov, "Ultracompact, low RF power, 10 Gb/s silicon Mach-Zehnder modulator," *Optics Express*, vol. 15, no. 25, pp. 17106-17113, 2007.

[62] C. A. Balanis, "Antenna Theory Analysis and Design," 3rd edition, 2005.

[63] S. M. Jackman, M. Swartz, M. Burton, and T. W. Head, "CWDP Certified Wireless Design Professional Official Study Guide: Exam," PW0-250, 2011.

[64] P. J. Bevelacqua, "Antenna Impedance," [Online] http://www.antenna-theory.com

[65] K. Kim, B. Floyd, J. Mehta, H. Yoon, C. Hung, D. Bravo, T. Dickson, X. Guo, R. Li, N. Trichy, J. Caserta, W. Bomstad, J. Branch, D. Yang, J. Bohorquez, E. Seok, L. Gao, A. Sugavanam, J. Lin, J. Chen, and J. Brewer, "On-chip antennas in silicon ICs and their application," *IEEE Transactions on Electron Devices*, vol. 52, no. 7, pp.1312-1323, Jul. 2005.

[66] P. Guo, and H. Chuang, "A 60-GHz millimeter-wave CMOS RFIC-on-chip meander-line planar inverted-F antenna for WPAN applications," *IEEE Antennas Propagat. Soc. Int. Symp*, 2008.

[67] S. S. Hsu, K. C. Wei, C. Y. Hsu, and H. Ru-Chuang, "A 60-GHz millimeter-wave CPW-fed Yagi antenna fabricated by using 0.18-μm CMOS technology," *IEEE Electron Device Letters*, vol. 29, no. 6, pp.625-627, Jun 2008.

[68] C. C. Lin, S. S. Hsu, C. Y. Hsu, and H. R. Chuang, "A 60-GHz millimeter-wave CMOS RFIC-on-chip triangular monopole antenna for WPAN applications," *IEEE Antennas Propagat. Soc. Int. Symp.*, pp. 2522-2525, Jul. 2007.

[69] C. A. Balanis, "Advanced engineering electromagnetism", 1989.

[70] D. Sievenpiper, L. Zhang, R. Broas, N. Alexopolous, and E. Yablonovitch, "High-impedance electromagnetic surfaces with a forbidden frequency band," *IEEE Transactions on Microwave Theory Tech*, vol. 47, no. 11, pp. 2059–2074, Nov. 1999.

[71] M. Abedin and M. Ali, "Effects of EBG reflection phase profiles on the input impedance and bandwidth of ultrathin directional dipoles," *IEEE Transactions on Antenna and Propagation*, vol. 53, no. 11, pp. 3664-3672, Nov. 2005.

[72] F. Yang and Y. Rahmat-Samii, "Reflection Phase Characterizations of the EBG Ground Plane for Low Profile Wire, Antenna Applications," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 10, pp. 2691-2703, Oct. 2003.

[73] F. Yang, A. Aminian, and Y. Rahmat-Samii, "A novel surface-wave antenna design using a thin-periodically loaded ground plane," *Microwave and Optical Technology Letters*, vol. 47, no. 3, pp. 240-245, Nov. 2005.

[74] HFSS ANSYS, [Online] http://www.ansys.com/Products/Simulation+Technology/Electronics/Signal+Integrity/ch.ANSYS+HFSS.cz

[75] Rogers 4003C Laminates Manual, Rogers Corporation, [Online] http://www.rogerscorp.com

[76] H. Yeh, N. Hiramatsu, and K. Melde, "The Design of Broadband 60 GHz AMC Antenna in Multi-Chip RF Data Transmission," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 4, pp. 1623-1630, Dec. 2012.

[77] H. Yeh and K. Melde, "Development of 60-GHz wireless interconnects for interchip data transmission." *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 3, no. 11, pp. 1946-1952, Jul. 2013.

[78] H. Yeh, N. Hiramatsu, and K. Melde, "Wireless RF data communications using 60 GHz antennas in multi-core systems," *Electrical Performance of Electronic Packaging and Systems (EPEPS), IEEE 20th Conference on*, 2011.

[79] J. Mars and R. Hott, Tilera (RAW) Processor, 2011, [Online] http://www.cs.virginia.edu/~skadron/cs8535_s11/Tilera.pdf

[80] A. Mejia, M. Palesi, J. Flich, S. Kumar, P. López, R. Holsmark, and J. Duato, "Region-based routing: a mechanism to support efficient routing algorithms in NoCs," *IEEE Transactions on Very Large Scale Integration (VLSI)*, vol. 17, no. 3, pp. 356-369, Mar. 2009.

[81] S. Chari, C. Jutla, J. R. Rao, and P. Rohatgi, "A cautionary note regarding evaluation of AES candidates on smart-cards," *Second AES Candidate Conference*, Rome, Italy, 1999.

[82] J. Daemen and V. Rijmen, "Resistance against implementation attacks: a comparative study of the AES proposals," 1999.

[83] J. Daemen and V. Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard," *Springer-Verlag*, 2002.

[84] J. A. Ambrose, N. Aldon, A. Ignjatovic, and S. Parameswaran, "Anatomy of differential power analysis for AES," *Symbolic and Numeric Algorithms for Scientific Computing, SYNASC'08, 10th International Symposium on*, pp. 459-466, IEEE, 2008.

[85] C. Gebotys, "A Table Masking Countermeasure for Low-Energy Secure Embedded Systems," *IEEE Trans. on VLSI*, 14(7):740–753, 2006.

[86] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Computers*, 51(5):541–552, 2002.

[87] E. Oswald and M. Aigner, "Randomized addition-subtraction chains as a countermeasure against power attacks," *CHES '01*, pp. 39-50, London, UK, 2001.

[88] M. Barbosa and D. Page, "On the automatic construction of indistinguishable operations," *Cryptography And Coding*, pp. 233-247. Springer-Verlag LNCS 3796, Nov. 2005.

[89] M.-L. Akkar, R. Bevan, P. Dischamp, and D. Moyart, "Power analysis, what is now possible...," *Asiacrypt '00*, pp. 489-502, London, UK, Springer-Verlag, 2000.

[90] P. Tahmasebi, A. Hezarkhani, and M. Sahimi, "Multiple-point geostatistical modeling based on the cross-correlation functions," *Computational Geosciences*, 16(3), 779-797.

[91] Q. Luo, Y. Fei, "Algorithmic collision analysis for evaluating cryptographic systems and side-channel attacks," *Hardware-Oriented Security and Trust (HOST), IEEE International Symposium on*, 75-80, IEEE, 2011.

[92] S. Yoo, “Electromagnetic Modeling of Multi-Dimensional Scale Problems: Nanoscale Solar Materials, RF Electronics, Wearable Antennas,” 2014.

[93] J. Poovey, M. Levy, S. Gal-On, and T. M. Conte, “A benchmark characterization of the EEMBC benchmark suite,” 2009.