Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 509465, 15 pages
doi:10.1155/2012/509465

Review Article

Networks on Chips: Structure and Design Methodologies

Wen-Chung Tsai,[1] Ying-Cherng Lan,[1] Yu-Hen Hu,[2] and Sao-Jie Chen[3]

[1] *Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan*
[2] *Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706-1691, USA*
[3] *Department of Electrical Engineering and Graduate Institute of Electronics Enginering, National Taiwan University, Taipei 106, Taiwan*

Correspondence should be addressed to Sao-Jie Chen, csj@cc.ee.ntu.edu.tw

Received 18 September 2011; Accepted 1 October 2011

Academic Editor: Jiang Xu

The next generation of multiprocessor system on chip (MPSoC) and chip multiprocessors (CMPs) will contain hundreds or thousands of cores. Such a many-core system requires high-performance interconnections to transfer data among the cores on the chip. Traditional system components interface with the interconnection backbone via a bus interface. This interconnection backbone can be an on-chip bus or multilayer bus architecture. With the advent of many-core architectures, the bus architecture becomes the performance bottleneck of the on-chip interconnection framework. In contrast, network on chip (NoC) becomes a promising on-chip communication infrastructure, which is commonly considered as an aggressive long-term approach for on-chip communications. Accordingly, this paper first discusses several common architectures and prevalent techniques that can deal well with the design issues of communication performance, power consumption, signal integrity, and system scalability in an NoC. Finally, a novel bidirectional NoC (BiNoC) architecture with a dynamically self-reconfigurable bidirectional channel is proposed to break the conventional performance bottleneck caused by bandwidth restriction in conventional NoCs.

1. Introduction

As the density of VLSI design increases, the complexity of each component in a system raises rapidly. To accommodate the increasing transistor density, higher operating frequencies, and shorter time-to-market pressure, multiprocessor system on chip (MPSoC) and chip multiprocessor (CMP) architectures, which use bus structures for on-chip communication and integrate complex heterogeneous functional elements on a single die, are more and more required in today's semiconductor industry. However, today's SoC designers face a new challenge in the design of the on-chip interconnects beyond the evolution of an increasing number of processing elements. Traditional bus-based communication schemes, which lack for scalability and predictability, are not capable to keep up with the increasing requirements of future SoCs in terms of performance, power, timing closure, scalability, and so on. To meet the design productivity and signal integrity challenges of next-generation system designs, a structured and scalable interconnection architecture, network on chip (NoC), has been proposed recently to mitigate the complex on-chip communication problem.

An application can be represented as a set of computational units that require a set of communication blocks to pass information between the units. To distinguish the performance impact of these two major components, computation time is dominated by gate delay whereas communication time is dominated by wire delay. When the amount of computational units is low, the communication blocks can be done on an ad-hoc basis. However, with the shrinking size of transistors in recent years, gate delay is ever decreasing with respect to wire delay. Thus, we need a structured and scalable on-chip communication architecture to fit the increasingly complex applications on a single chip. This translates to the design of on-chip communications architecture as being more and more important and promotes the design concept from computation-centric design to communication-centric design.

System on chip (SoC) is an architectural concept developed in the last few decades, in which a processor or few

processors along with memory and an associated set of peripherals connected by busses are all implemented on a single chip. According to the Moore's law, the trend toward many-core processing chips is now a well established one. Power-efficient processors combined with hardware accelerators are the preferred choice for most designers to deliver the best tradeoff between performance and power consumption, since computational power increases exponentially according to the calculation of dynamic power dissipation [1]. Therefore, this trend dictates spreading the application tasks into multiple processing elements where (1) each processing element can be individually turned on or off, thereby saving power, (2) each processing element can run at its own optimized supply voltage and frequency, (3) it is easier to achieve load balance among processor cores and to distribute heat across the die, and (4) it can potentially produce lower die temperatures and improve reliability and leakage. However, while ad-hoc methods of selecting few blocks may work based on a designer's experience, this may not work as today's MPSoC and CMP designs which becomes more and more complex. Consequently, SoC design nowadays needs techniques which can provide an efficient method of enabling a chip to compute complex applications and to fit area-wise on a single chip according to today's technology trends.

A communication scheme is composed of an interconnection backbone, physical interfaces, and layered protocols which make the on-chip communication take place among components on a MP-SoC or CMP. As the design complexity scales up, intrachip communication requirements are becoming crucial. Data-intensive systems such as multimedia devices, mobile installations, and multiprocessor platforms need a flexible and scalable interconnection scheme to handle a huge amount of data transactions on chip. Customarily, dedicated point-to-point wires are adopted as sets of application-specific global on-chip links that connect the top-level modules. However, as wire density and length grow with the system complexity, the communication architecture based on point-to-point wires becomes no more feasible due to its poor scalability and reusability. Specifically, as signals are carried by the global wires across a chip, these metal wires typically do not scale in length with technology. Propagation delay, power dissipation, and reliability will be the serious issues of global wires in deep submicron VLSI technology. According to [2], as silicon technologies advance to 50 nm and beyond, global wires will take 6 to 10 cycles to propagate, which will then far outweigh gate delays and make cross-chip long wire timing difficult to meet. Keeping track of the status in all elements and managing the global communication among top-level modules by a centralized way are no longer feasible. Therefore, reusable on-chip bus interconnect templates such as ARM's AMBA [3] and IBM's CoreConnect [4] are commonly used in current MP-SoC amd CMP designs, such that the modules can share the same group of interconnection wires in a bus-based communication architecture.

However, on-chip bus allows only one communication transaction at a time according to the arbitration result; thus, the average communication bandwidth of each processing element is in inverse proportion to the total number of IP cores in a system. This character makes a bus-based architecture inherently not scalable for a complex system in today's MP-SoC and CMP designs. Implementing multiple on-chip buses in a hierarchical architecture or in a separated manner may alleviate this scalability constraint, but it requires application-specific grouping of processing elements and design of different communication protocols to meet the application requirements. Furthermore, whenever a new application needs to be designed for, or a new set of peripherals needs to be added, a chip designed with only simple buses will lack means of efficiently determining feasibility, not to mention optimality [5]. In addition, attempts to guarantee quality of service (QoS) for system performance will be a manually intensive task. Therefore, bus-based design needs to be exchanged with a method that is flexible, scalable, and reusable.

Since the latest process technology allows for more processors and more cores to be placed on a single chip, the emerging MP-SoC and CMP architectures, which demand high throughput, low latency, and reliable global communication services, cannot be met by current dedicated bus-based on-chip communication infrastructure. Trying to achieve such designs with a bus structure could be problematic for a number of reasons including timing closure, performance issues, and scalability. Specifically, as the feature size of modern silicon devices shrinks below 50 nanometers, global interconnection delays constrain attainable processing speed. Device parameter variations further complicate the timing and reliability issues. A paradigm shift focusing on communication-centric design, rather than computation-centric design, seems to be the most promising approach to address these communication crises [6–11]. Consequently, in the past few years, a new methodology called network on chip has been introduced as a means of solving these issues by introducing a structured and scalable communication architecture.

In the sequel, Section 2 will introduce the NoC architecture and its function layers. In Section 3, we will discuss the NoC design methodologies. Then, a bidirectional network-on-chip (BiNoC) architecture will be given in Section 4. Finally, conclusion will be drawn in Section 5.

2. Network-on-Chip Architecture and Function Layers

Network on chip is the term used to describe an architecture that has maintained readily designable solutions in face of communication-centric trends. In this section, we will briefly review some concepts on the design of an NoC communication system. Moreover, the NoC function can be classified into several layers, which will be introduced sequentially.

2.1. Network-on-Chip Architecture. A typical NoC architecture consists of multiple segments of wires and routers as shown in Figure 1. In a tiled, city-block style of NoC layout, the wires and routers are configured much like street grids of a city, while the clients (e.g., logic processor cores) are placed on city blocks separated by wires. A network interface (NI) module transforms data packets generated from the client

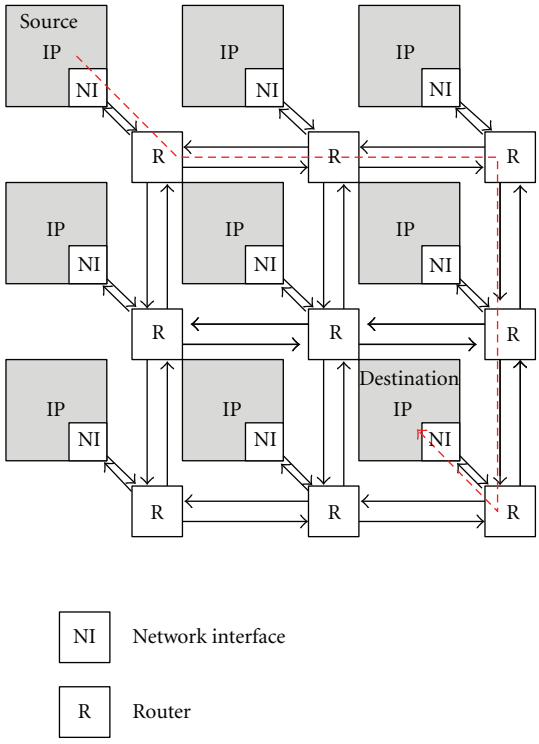


Figure 1: Typical NoC architecture in a mesh topology.

logic (processor cores) into fixed-length flow-control digits (flits). The flits associated with a data packet consist of a header (or head) flit, a tail flit, and a number of body flits in between. This array of flits will be routed toward the intended destination in a hop-by-hop manner from one router to its neighboring router.

In a city-block style NoC, each router has five input ports and five output ports corresponding to the north, east, south, and west directions as well as the local processing element (PE). Each port will connect to another port on the neighboring router via a set of physical interconnect wires (channels). The router's function is to route flits entering from each input port to an appropriate output port and then toward the final destinations. To realize this function, a router is equipped with an input buffer for each input port, a 5×5 crossbar switch to redirect traffic to the desired output port and necessary control logic to ensure correctness of routing results as shown in Figure 2.

Usually, for each data packet, the corresponding head flit specifies its intended destination. After examining the head flit, the router control logic will determine which output direction to route all the subsequent (body and tail) flits associated with this data packet according to the routing algorithm applied.

2.2. Network-on-Chip Function Layers. The NoC function can be classified into several layers: application, transport, network, data link, and physical layers. An NoC router should contain both software and hardware implementations to support functions of the layers.

2.2.1. Application Layer. At the application layer, target applications will be broken down into a set of computation and communication tasks such that the performance factors like energy and speed can be optimized. Placement of cores on an NoC has to be optimized to reduce the amount of total communication or energy but at the same time recognizing the limitations of any one particular link. The task mapping and communication scheduling problem is an instance of a constrained quadratic assignment problem which was known to be NP-hard [12]. Given a target application described as a set of concurrent tasks with an NoC architecture, the fundamental questions to answer are (1) how to topologically place the selected set of cores onto the processing elements of the network and (2) how to take into consideration the complex effects of network condition, which may change dynamically during task execution, such that the metrics of interest are optimized [13]. To get the best tradeoff between power and performance, application mapping and scheduling should be considered with several kinds of architecture parameters.

2.2.2. Transport Layer. To prevent buffer overflow and to avoid traffic congestion, some management schemes should be applied to guide the transport of packets in an NoC. The transport layer addresses the congestion and flow control issues [14]. Key performance metrics of an NoC include low packet delivery latency and high-throughput rate, and these metrics are critically impacted by network congestions caused by resource contentions. Accordingly, contention resolution is a key to avoid network congestions [14]. One of the most crucial issues for the contention resolution is, under a premise of a deadlock- and livelock-free routing algorithm, to enhance the utilization efficiency of available network resources in order to come up with a better communication performance.

2.2.3. Network Layer. Network topology or interconnect architecture is an important issue in this layer, which determines how the resources of network are connected, thus, refers to the static arrangement of channels and nodes in an interconnection network. Irregular forms of topologies can be derived by mixing different forms of communication architectures in a hierarchical, hybrid, or asymmetric way by clustering partition, which may offer more connectivity and customizability at the cost of complexity and area. In addition, optimization of a topology, which affects the connectivity of the routers and the distance of any one core to the other, is difficult. Furthermore, the tradeoff between generality and customization that, respectively, facilitate scalability and performance is important. As future designs become more complex, the non-recurring costs of architecting and manufacturing a chip will become more and more expensive. A homogenous NoC is one where the cores and routers are all the same, while a heterogeneous NoC selects individual cores from an IP library and may have its communication architecture customized to suit the needs of an application. Since NoC designs must be flexible enough to cover a certain range of applications, most of the state-of-the-art NoC designs use a mesh or torus topology because of its performance benefits

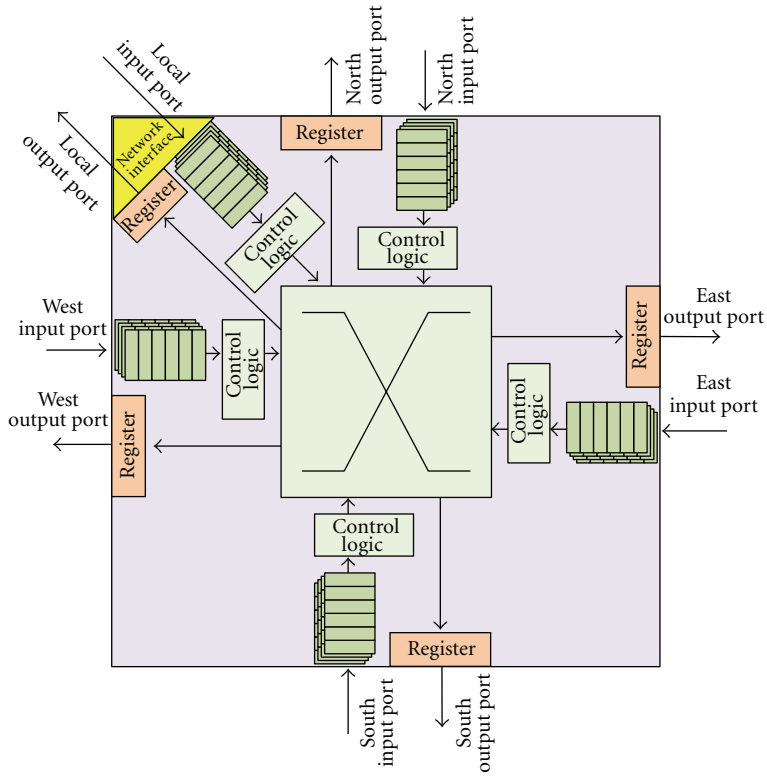


Figure 2: Typical NoC router architecture.

and high degree of scalability for two-dimensional systems, yet it may not achieve the best performance for a single application [15, 16].

In addition, the network layer also needs to deal with the routing data between processing elements. First, packetizing algorithms deal with the decomposition of a message into packets at source nodes and their assembly at destination nodes. Then, the transmission of packets can be executed by the choice of routing algorithms based on different network topologies [6]. Routing algorithm determines the path strategy of a packet from its source node to the destination node. Determining packet routes and resolving conflicts between packets when the same route is requested, with respect to improving on-chip communication performance, are two of the important responsibilities of a router.

Conventional design of a router consists of circuit-switched fabrics and an arbitration controller. In each arbitration decision, more than one path can be constructed by the crossroad switch as long as no contention exists between these paths. For most existing switch designs, virtual-channel flow-control-based router design, which provides better flexibility and channel utilization with smaller buffer size, is a well-known technique from the domain of multiprocessor networks [17–24].

2.2.4. Data Link and Physical Layers. The main purpose of data-link layer protocols is to increase the reliability of the link up to a minimum required level, under the assumption that the physical layer by itself is not sufficiently reliable [14]. The emphasis on physical layer is focused on signal drivers and receivers, as well as design technologies for resorting and pipelining signals on wiring. In addition, as technology advanced to ultradeep submicron (DSM), smaller voltage swings and shrinking feature size translate to decreased noise margin, which cause the on-chip interconnects less immune to noise and increase the chances of nondeterminism in the transmission of data over wires (transient fault) [2, 25–28]. Electrical noise due to crosstalk, electromagnetic interference (EMI), and radiation-induced charge injection will likely produce timing error and data errors and make reliable on-chip interconnect hard to achieve.

Error control schemes and utilization of the physical links to achieve reliability are the main concern of these layers. First, a credible fault model must be developed. Then, an error control scheme that is low power, low area, high bandwidth, and low latency must be designed. In NoC design, packet-based data transmission is an efficient way to deal with data errors because the effect of errors is contained by packet boundaries that can be recovered on a packet-by-packet basis.

3. Network-on-Chip Design Methodologies

This section discusses several prevalent NoC design methodologies, such as flow control, routing, arbitration, quality of service, reliability, and task scheduling.

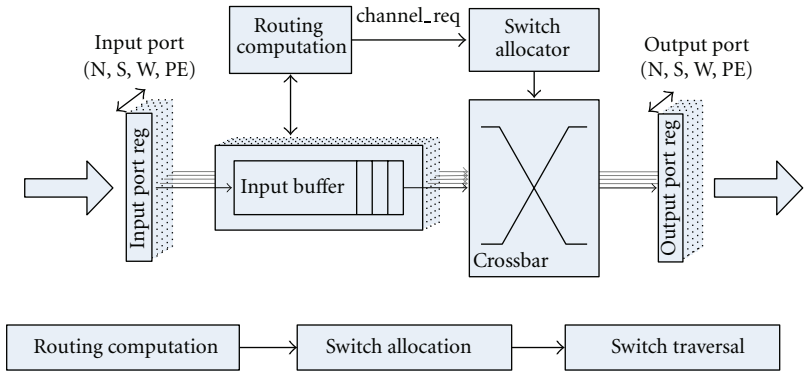


Figure 3: Typical router design based on wormhole flow control.

3.1. Flow-Control Mechanism. The performance of NoC communication architecture is dictated by its *flow-control* mechanism. Adding buffers to networks significantly improves the efficiency of a flow-control mechanism since a buffer can decouple the allocation of adjacent channels. Without a buffer, the two channels must be allocated to a packet (or flits) during consecutive cycles, or the packet must be dropped or misrouted [6]. More specifically, with *buffered flow control*, when a packet arrives at a router, it must first occupy some resources, such as channel bandwidth and buffer capacity, depending on the flow-control methodology. Each router must juggle among multiple input data streams from multiple input ports and route them to appropriate output ports with the highest efficiency.

Buffered flow-control methods can be classified into *packet-buffer flow control* and *flit-buffer flow control* based on their granularity of buffer allocation and channel bandwidth allocation [6]. Since allocating resources in unit of flit can achieve more storage utilization efficiency than that in unit of packet. Two types of *flit-buffer flow-control* architectures are commonly used in NoC: the *wormhole* flow control and the *virtual-channel* flow control.

3.1.1. Packet-Buffer Flow Control. Packet-buffer flow-control allocates network resources in a packet-by-packet basis. Examples are *store-and-forward flow control* and *virtual-cut-through flow control*. In store-and-forward method, each node must ensure that it has already received and stored an entire packet before forwarding it to the downstream node. While the virtual-cut-through scheme can forward a packet as long as there is enough buffer space to receive a packet at the downstream node. As a result, virtual cut through introduces lower communication delay than store and forward does. However, packet-buffer flow control needs larger size of buffer space in one node because of its inefficient use of buffer storage. In addition, allocating channels in units of packets will increase contention latency.

3.1.2. Wormhole Flow-Control-Based Router. Wormhole flow control improves performance through a finer granularity of message allocation at flit level instead of packet level. This technique allows more efficient use of buffer than the packet-buffer flow-control mechanism since the buffer size in each router can be reduced significantly [29, 30]. A typical three-stage pipelined NoC router architecture based on wormhole flow control is shown in Figure 3. Every input port has a FIFO-based input buffer, which can be seen as a single virtual channel used to hold blocked flits. To facilitate wormhole flow-control-based routing [6], the *routing computation* (RC) module will send a channel request signal to the *switch allocator* (SA) for data in each input buffer. If the downstream buffer at a neighboring router has vacant space, SA will allocate the channel and route the data flits through the crossbar switch toward the designated downstream router at the *switch traversal* (ST) stage.

However, wormhole flow-control-based switching technique saves buffer size at the expense of throughput since the channel is owned by a packet, but buffers are allocated on a flit-by-flit basis. As such, an idle packet may continue block a channel even when another packet is ready to use the same channel, leading to inefficient resource utilization. This is the well-known *head of line* (HoL) blocking problem. Therefore, virtual-channel flow-control-based router architecture was proposed to reduce blocking effect and to improve network latency.

3.1.3. Virtual-Channel Flow-Control-Based Router. Virtual-channel flow control assigns multiple virtual paths, each with its own associated buffer queue, to the same physical channel; thus, it increases throughput by up to 40% over wormhole flow control and helps to avoid possible deadlock problems [19, 31, 32]. A virtual channel flow-control router architecture as shown in Figure 4 can be seen as a remedy to the shortcoming of the wormhole flow-control scheme. By multiplexing multiple virtual-channels into the same input buffer, an idle packet will no longer block other packets that are ready to be routed using the shared physical channel. In a typical virtual-channel flow-control-based router, the flits are routed via a four-stage pipeline: *routing computation*, *virtual-channel allocation*, *switch allocator*, and *switch traversal*.

One incoming flit that arrives at a router is first written to an appropriate input virtual-channel queue and waits to

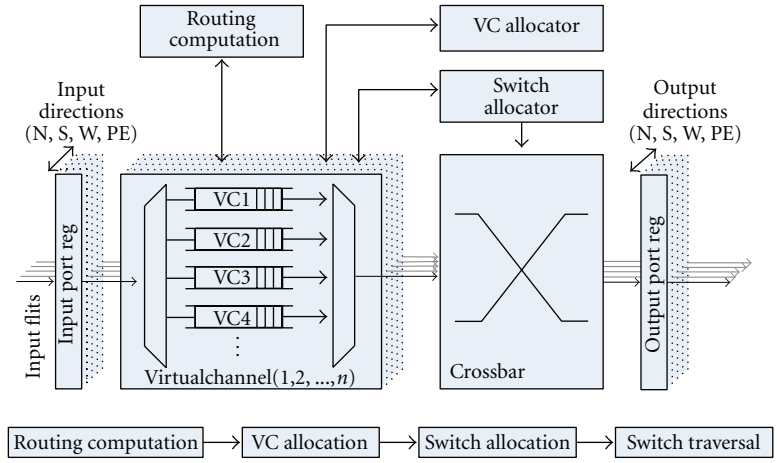


Figure 4: Typical router design based on virtual-channel flow control.

be processed. When a head flit reaches the top of its virtual-channel buffer queue and enters the RC stage, it is decoded by the RC module and generates an associated direction request. The direction request of this flit is then sent to the VA module to attain virtual channel at the downstream router. There might be some contentions among packets that request for the same virtual channel at the downstream router. The loser packets will be stalled at the VA stage, and the following flit in the previous stage will also be blocked due to this contention failure. Note that the processes of RC and VA actually take place only on the head flit. The subsequent body flits and tail flit of a packet simply accede to the routing decision acquired by the head flit and require no further processing at the RC and VA stages. Once a decision on the output virtual-channel selection is made at the VA stage, the SA module will assign physical channels to intrarouter flits. Flits granted with a physical channel will traverse through the crossbar switch to the input buffer of the downstream router during the ST stage, and the process repeats until the packet arrives at its destination.

3.2. Routing and Arbitration Techniques. A general problem pertaining to the routing and arbitration algorithms can be stated as follows: given an application graph, which can be represented by a unique traffic pattern, and a communication architecture, find a decision function at each router for selecting an output port that achieves a user-defined objective function.

3.2.1. Problem Decomposition. The above problem has three main parts: a traffic pattern, an NoC communication architecture, and an algorithm which best satisfies a set of user-defined objectives. First, the traffic patterns known ahead of time can be dealt with by a scheduling algorithm. On the other hand, dynamic or stochastic traffic patterns rely on the use of a routing algorithm with a varying degree of adaptation to route packets. Our focus will be on the patterns not known ahead of time.

Second, NoC communication architectures can have different topologies. The most common one is a regular 2D mesh, frequently used to display the behavior of adaptive routing algorithms. Other work, such as [33], deal with irregular regions in meshes. Our focus is independent of topology.

The third part deals with the algorithms themselves and the objectives to achieve. Two primary algorithms used to determine where and when a packet will move are routing and arbitration. A routing algorithm decides which direction each input packet should travel. Arbitration is the process of deciding which input packet request should be granted when there are more than one input packet requests for the same output port.

3.2.2. State of the Art. A typical router in an NoC is responsible for moving the received packets from the input buffers, with its routing and arbitration algorithms, to the output ports. The decisions which a router makes are based on the information collected from the network. Centralized decisions refer to making decisions based on the information gathered from the entire network [34]. Distributed decisions refer to making decisions based only on the information generated by the local router or nearby routers. Distributed routing, the focus of this paper, allows NoCs to grow in size without worrying about the increasing order of complexity within a centralized routing unit. An example of centralized routing is the AntNet algorithm [35], which depends on global information to make routing decisions, thus, needs extra ant buffers, routing tables, and arbitration mechanisms at each node.

There are some distributed routing algorithms which only rely on local information. They have been proposed as being efficient and still maintaining low overhead and high scalability. Routing algorithms in this category include deterministic and adaptive algorithms. Under realistic traffic patterns which pose the problem of hotspot traffic congestion areas, XY deterministic routing failed to avoid hotspots and resulted in high-average latencies [36]. Adaptive routing guides the router to react to hotspots created by different

traffic patterns, by allowing a packet at the input buffer to request more than one output port or direction [37]. While minimal routing algorithms prevent livelock from occurring, adaptive routing introduces the possibility of deadlock, which can be prevented by applying odd-even turn model restrictions to the routing decision [38].

As presented in [36], the DyAD router dynamically switches from deterministic to adaptive routing when congestion is detected, since deterministic routing achieves low packet latency under low packet injection rates. Neighboring nodes send indication to use adaptive routing when their buffers are filled above a preset threshold. Under these conditions, the router dictates that packets are routed in the direction with more available input buffer slots. This minimal adaptive algorithm, used in the presence of hotspots and increasing congestion rates, pushes back the saturation point of the traffic in the network. Another extension of adaptive routing is the neighbors-on-path (NoP) algorithm [39], which allows each router to monitor two hops away the input buffers of the routers in order to detect potential congestion earlier. By earlier detection of the buffer fill level, routes can avoid congestion better. DyXY is an algorithm which utilizes a history of buffer fill levels to make decisions [40]. The algorithms presented in [41, 42] utilize variants of buffer fill level to make decisions.

In addition to making a routing decision based on the buffer information of downstream packets, the other part of a router's decision making is the arbitration of packets. When multiple input packets are designated to be forwarded to the same next hop destination, arbitration algorithms such as round-robin or first-come first-serve (FCFS) have been proposed to resolve the output port contention. These arbitration algorithms could be designed to relieve upstream buffers with higher congestion. contention-aware input selection (CAIS) algorithm [43] is an improved arbitration algorithm that contributes to reduce the routing congestion situation by relieving hotspots of upstream traffic, determined by requests from the upstream traffic.

More works have been proposed to deal with some variance of the routing or arbitration algorithms. Sometimes, we categorize the former ones as methods of congestion avoidance; in other words, they evaluate downstream network conditions to avoid sending packets towards the congested areas so as not to aggravate the congestion conditions. We categorize the latter as methods of congestion relief; in other words, they evaluate upstream network conditions to determine which area had the most congestion to send first in order to quickly diffuse the congested situation.

3.3. Quality-of-Service Control. There is a wide range of possibilities for implementing guaranteed services on a network. Referring to the state-of-the-art QoS mechanisms for NoCs, they can be categorized into two types of schemes: connection oriented (circuit switching) and connection less (packet-switching).

3.3.1. Connection-Oriented Scheme. In connection-oriented schemes, guaranteed-service (GS) packets traverse on some particular channels or buffers that were reserved for them. Specifically, the connection path between the source and destination pair of GS packets is built at the time before they are injected onto the network [44–51]. However, this kind of static preallocation may result in high service latency and does not consider hotspots created by temporal shifts in data requirements, thus, leads to a rather unscalable NoC.

Connection-oriented QoS mechanism is reliable to achieve QoS requirement, since connections are created guaranteeing tight bounds for specific flows. Two types of the programming models for constructing the set-up phase were presented: centralized programming and distributed programming. Centralized programming sets up the reservations by a configuration manager which takes over all the resources in the network. On the contrary, distributed program models let all the resource reservations to be handled by each local router. The centralized method is simpler to achieve while it is only suitable for small-size systems. Despite the hardware overhead in routers, distributed program models have acquired popularity in a large system because of its better flexibility.

However, connection-oriented QoS mechanism comes with greater hardware overhead in control and storage for resource reservations and poor scalability because complexity grows with each node added. Furthermore, bandwidth usage is inefficient, and resource allocation has to be considered on a worst case basis. Moreover, the set-up phase of guaranteed traffic presents a timing overhead which may result in inefficiency for nondeterministic applications.

3.3.2. Connection-Less Scheme. The connection-less scheme is an alternative way to support different service levels in NoCs where the resource authorities are prioritized according to the QoS requirement of a traffic flow [48]. This is a distributed technique which allows traffic to be classified into different service levels. These service levels can often coincide with different virtual channels inside the switch. As two traffic flows with different QoS requirements are presented on the same channel simultaneously, the higher prioritized flow can interrupt the lower one and traverse this channel antecedently [48, 52]. It is more adaptive to network traffic and potential hotspots and can better utilize the network.

Different from the connection-oriented schemes, connection-less schemes do not execute any resource reservation. In contrast, multiple traffic flows share the same priority or the same resource, thus, could cause unpredictable conditions [53]. The traffic with higher service level is guaranteed in a relative fashion in a connection-less scheme by prioritizing each type of traffic flow. However, while the connection-less scheme provides a coarser QoS support as the connection-oriented schemes, they can offer a better adaptation of communication to the varying network traffic. Furthermore, better bandwidth utilization and less hardware cost can be achieved since the traffic is allocated with network resources dynamically. With the consideration of performance requirements for each service level, a network designer can select an appropriate bandwidth implemented in an NoC to both meet the QoS constraints and save the wiring cost [48, 54, 55].

Although connection-oriented communication guarantees tight bounds for several traffic parameters, an erroneous decision of resource reservation might cause an unexpected performance penalty. While in a connection-less network, a nonoptimal priority assignment has less degradation of throughput though it provides coarse QoS support. As pointed out in [20], guaranteed services require resource reservation for the worst case in a connection oriented, which causes a lot of wasted resource. In addition, some quantitative modeling and comparison of these two schemes, provided in [56], has shown that under a variable-bit-rate application, connection-less technique provides a better performance in terms of the end-to-end packet delay. These comparisons can help to design an application-specific NoC using a suitable QoS scheme.

3.4. Reliability Design. The trend towards constructing large computing systems incorporated with a many-core architecture has resulted in a two-sided relationship involving reliability and fault tolerance consideration. While yield has always been a critical issue in recent high-performance circuitry implementation, the document of the International Technology Roadmap for Semiconductor (ITRS) [57] states that "*Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification and test*." The general principle of fault tolerance for any system can be divided in two categories:

(1) employment of hardware redundancy to hide the effect of faults,

(2) self-identification of source of failure and compensating the effect by appropriate mechanism.

If we can make such a strategy work, a system will be capable of testing and reconfiguring itself, allowing it to work reliably throughout its lifetime.

3.4.1. Failure Types in NoC. Scaling chips, however, increase the probability of faults. Faults to be considered in an NoC architecture can be categorized into permanent (hard fault) and transient fault (soft fault) [13, 58]. The former one reflects irreversible physical changes, such as electro-migration of conductor, broken wires, and dielectric breakdowns. In this case, permanent damages in a circuit cannot be repaired after manufacture. Therefore, the module which is suffering a permanent fault should turn off its function and inform neighboring modules of this information. Then, rerouting packets with an alternative path will be re-calculated deterministically or dynamically according to the need. However, this may induce nonminimal path routing and increase the complexity of routing decision. Hardware redundancy such as spare wire or reconfigurable circuitry can also be used to avoid using of faulty modules [59–62]. In the latter case, several phenomena, such as neutron and alpha particles, supply voltage swing, and interconnect noise, induce the packet invalid or misrouted. Usually, a transient fault is modeled with a probability of bit error rate under an adequate fault model. In an NoC system, intrarouter or interrouter functionality errors may happen, to understand how to deal with the most common sources of failures in an NoC; Park et al. provided comprehensive fault-tolerant solutions relevant to all stages of decision making in an NoC router [63].

3.4.2. Reliability Design in NoC. A number of fault-tolerant methods were proposed in [64, 65] for large-scale communication systems. Unfortunately, these algorithms are not suitable for an NoC, because they will induce significant area and resource overhead. Dumitras et al. proposed a flood-based routing algorithm for NoC, named *stochastic communication*, which is derived from the fault-tolerance mechanism used in the computer network and distributed database fields. Such stochastic-communication algorithm separates computation from communication and provides fault tolerance to on-chip failures [57, 66]. However, to eliminate the high communication overhead of flood-based fault tolerance algorithm, Pirretti et al. promoted a redundant random-walk algorithm which can significantly reduce the overhead while maintaining a useful level of fault tolerance [67]. However, the basic idea of sending redundant information via multipath to achieve fault tolerance may cause much higher traffic load in the network, and the probabilistic broadcast characteristic may also result in additional unpredictable behavior on network loading.

Therefore, in a distributed NoC router considering practical hardware implementation, the error control scheme used to detect/correct interrouter transient fault in an NoC is required to have smaller area and shorter timing delay. An error control code that adapts to different degrees of detection and correction and has a low timing overhead will ease its integration into a router. The fault-tolerant method utilizing error detection requires an additional retransmission buffer specially designed for NoCs when the errors are detected. Error control schemes, such as the Reed-Solomon code proposed by Hoffman et al., have been used on NoCs [68]. But as their results show, the long delay would degrade the overall timing and performance of an NoC router.

3.5. Energy-Aware Task Scheduling. The availability of many cores on the same chip promises a high level of parallelism to expedite the execution of computation-intensive applications. To do so, a program must first be represented by a *task graph* where each node is a coarse-grained task (e.g., a procedure or a subroutine). Often, a task needs to forward its intermediate results to another task for further processing. This intertask data dependency is represented by a directed arc from the origin task to the destination task in the task graph. Tasks that have no intertask data dependency among themselves can be assigned for multiple processor cores to execute concurrently. As such, the total execution time can be significantly shortened.

A *real-time application* is an application in which execution time must be smaller than a *deadline*. Otherwise, the computation will be deemed a failure. To implement an application on an MC-NoC platform for parallel execution, each task in the task graph will be *assigned* to a processor core. Depending on the city-block distance between two tiles,

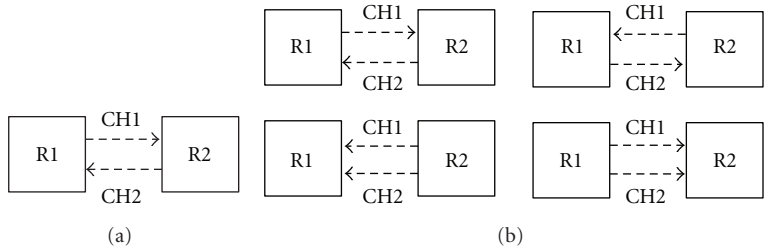


Figure 5: Channel directions in a typical NoC and proposed BiNoC.

intertask communication will take different amount of communication delay. For a particular application, proper task assignment will reduce communication delay while maximizing parallelism such that the total execution time can be minimized. For a real-time application, if the total execution time is less than the predefined deadline of the application, the *slacks* between them could be exploited to reduce energy consumption.

The execution time of a task may vary depending on the clock frequency the processor core is running. One technique to adjust the clock frequency of individual time on an MC-NoC is dynamic voltage scaling (DVS). When the clock frequency slows down, often the associated energy consumed by a running task is also reduced. Hence, in addition to assigning tasks to the processor cores located at appropriate tiles, another design objective would be to use DVS to save some energy while conforming to the deadline constraint, with perhaps smaller slacks.

Previously, it has been shown that the minimum energy multiprocessor task scheduling problem is NP-hard [69–71]. For real-time applications, it was proposed that execution of some tasks can be slowed down using DVS on corresponding tiles without violating the deadline timing constraint [72]. Several DVS-enabled uniprocessors have been implemented. Test results running real-world applications showed significant power saving up to 10 times [73]. For multiprocessor core systems implemented to execute a set of real-time dependent tasks, Schmitz et al. [74–76] presented an iterative synthesis approach for DVS-enabled processing element based on genetic algorithms (GA). They proposed a heuristic PV-DVS algorithm specifically for solving the voltage scaling. Kianzad et al. improved the previous work by combining assignment, scheduling, and power management in a single GA algorithm [77]. However, GA-based design optimization suffers slow convergence and lower desired quality. Chang et al. [78] proposed using Ant Colony Optimization (ACO) algorithm. Common to these approaches is that when PV-DVS is applied for power reduction, it is applied to one task (tile) at a time and is done after assignment and scheduling. Zhang et al. [79] and Varatkar and Marculescu [80] proposed using a list scheduling algorithm to find an initial task schedule, and the DVS problem was solved by integer linear programming. The idea behind these methods is to maximize the available slack in a schedule so as to enlarge the solution space of using DVS. However, the communication infrastructures used in these works are either a point-to-point interconnect or abus architecture. Hu and Marculescu [81] proposed an energy-aware scheduling (EAS) algorithm that considers the communication delay on an NoC architecture. However, DVS frequency adjustment was not considered.

4. Bidirectional Network-on-Chip (BiNoC) Architecture

A bidirectional channel network-on-chip (BiNoC) architecture is proposed in this section to enhance the performance of on-chip communication. In a BiNoC, each communication channel allows itself to be dynamically reconfigured to transmit flits in either direction. This added flexibility promises better bandwidth utilization, lower packet delivery latency, and higher packet consumption rate. Novel on-chip router architecture is developed to support dynamic self-reconfiguration of the bidirectional traffic flow. The flow direction at each channel is controlled by a channel-direction-control protocol. Implemented with a pair of finite state machines, this channel-direction-control protocol is shown to be of high performance, free of deadlock, and free of starvation.

4.1. Problem Description. In a conventional NoC architecture, each pair of neighboring routers uses two unidirectional channels in opposite direction to propagate data on the network as shown in Figure 5(a). In our BiNoC architecture, to enable the most bandwidth utilization, data channels between each pair of routers should be able to transmit data in any direction at each run cycle. That is, four kinds of channel-direction combinations should be allowed for data transmission as shown in Figure 5(b). However, current unidirectional NoC architectures, when facing applications that have different traffic patterns, cannot achieve the high bandwidth utilization objective.

Note that the number of bidirectional channels between each pair of neighboring router in BiNoC architecture is not limited to two. The more the channels that can be used, the better the performance results. In order to provide a fair comparison between our BiNoC and the conventional NoC that usually provided two fixed unidirectional channels, only two bidirectional channels were used in BiNoC as illustrated in Figure 5.

4.2. Motivational Example. As shown in Figure 6(a), an application task graph is typically described as a set of concurrent tasks that have already been assigned and scheduled

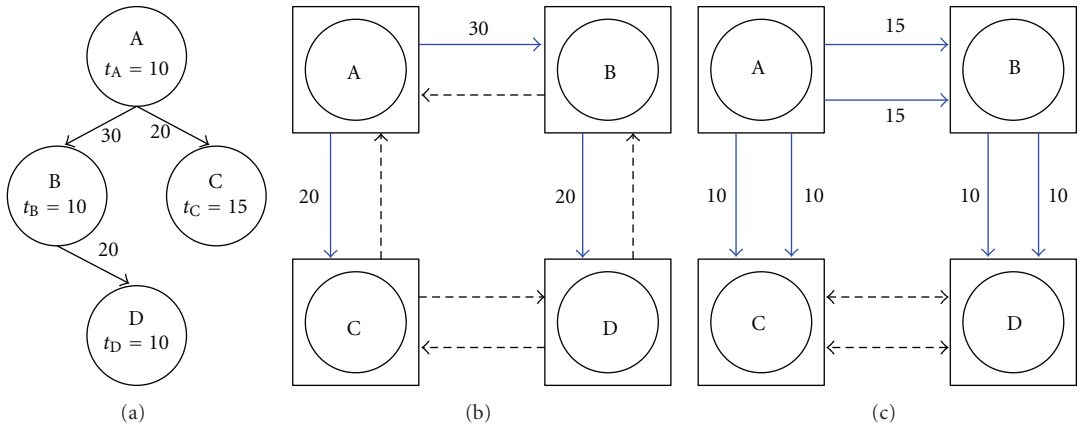


Figure 6: Example of task graph mapping on typical NoC and BiNoC.

onto a list of selected PEs. Each vertex represents a task with a value t_j of its computational execution time, and each edge represents the communication dependence with a value of communication volume which is divided by the bandwidth of a data channel.

For the most optimized mapping in a 2×2 2-dimensional mesh NoC as shown in Figure 6(b), the conventional NoC architecture in this case can only use three channels during the entire simulation and result in a total execution time of 80 cycles. However, if we can dynamically change the direction of each channel between each pair of routers like the architecture illustrated in Figure 6(c), the bandwidth utilization will be improved and the total execution time be reduced to 55 cycles. Figure 7 shows the detailed execution schedules, where the required communication time between nodes in BiNoC is extensively reduced.

4.3. Channel Bandwidth Utilization. During the execution of an application, the percentage of time when a data channel is kept busy is defined as channel bandwidth utilization U. To be more specific,

$$U = \frac{\sum_{t=1}^{T} N_{\text{Busy}}(t)}{T \times N_{\text{Total}}}, \tag{1}$$

where T is the total execution time, N_{Total} is the total number of channels available to transmit data, and $N_{\text{Busy}}(t)$ is the number of channels that are busy during clock cycle t. It is obvious that $U \leq 1$.

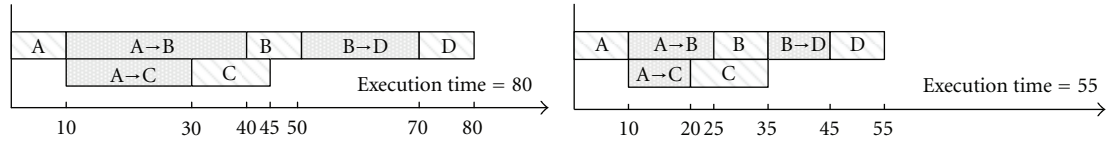
We have developed a cycle-accurate NoC simulator to evaluate the performance of a given NoC architecture. Additional implementation details of this NoC simulator will be elaborated in later sections. Using this simulator, we measured the channel bandwidth utilizations of a conventional NoC with respect to three types of synthetic traffic patterns: uniform, regional, and transpose. The channel utilization against different traffic volumes is plotted in Figure 8 under both XY and odd-even routings.

Figures 8(a) and 8(b) plot the bandwidth utilizations of a conventional NoC router with virtual-channel flow control. Four virtual-channel buffers, each with a depth of 8-flits, are allocated in each flow direction. Figures 8(c) and 8(d) give the percentage of time in which exactly one channel is busy and another channel is idle among time intervals when there is at least one channel busy. Figures 8(e) and 8(f) give the percentage of time that a bidirectional channel may help alleviating the traffic jam when exactly one channel is busy and the other is idle. Figures 8(a), 8(c), and 8(e) results are obtained using XY routing; and Figures 8(b), 8(d), and 8(f) use odd-even routing.

From Figures 8(a) and 8(b), it is clear that, even with the most favorable uniform traffic pattern, the channel bandwidth utilization peaks under XY routing and odd-even routing are only around 45% and 40%, respectively, under heavy traffic. For the transpose traffic pattern under XY routing, which is considered the worst case scenario, falls even below 20%. In other words, in a unidirectional channel setting even with two channels between a pair of routers, at most one channel is kept busy on average during normal NoC operation despite the deterministic routing algorithm such as XY or adaptive routing algorithm such as odd-even.

One possible cause of the low-bandwidth utilization as shown in Figures 8(a) and 8(b) is due to few bottleneck channels that take too long to transmit data packets in the designated direction. To validate this claim, we examine how often both channels between a pair of routers are kept busy simultaneously. In Figures 8(c) and 8(d), the percentage of time in which exactly one channel is busy and the other is idle given that one or both channels are busy is plotted, respectively, under XY and odd-even routings. As traffic load increases, it is clear that a significant amount of traffic utilizes only a single channel while the other channel is idle.

However, the situation where one channel is busy and the other is idle could be the case where there are no data that need to transmit in the opposite direction of the busy channel. It does not reveal whether there are additional data packets waiting in the same direction as the busy channel. These data packets are potential candidate to take advantage of the idle channel if the idle channel's direction can be reversed. In Figures 8(e) and 8(f), the percentage of time, in which there are data packets needed to transmit along the same direction as the busy channel while the other channel remains idle out



(a) (b)

Figure 7: Detailed execution schedules of typical NoC and BiNoC.

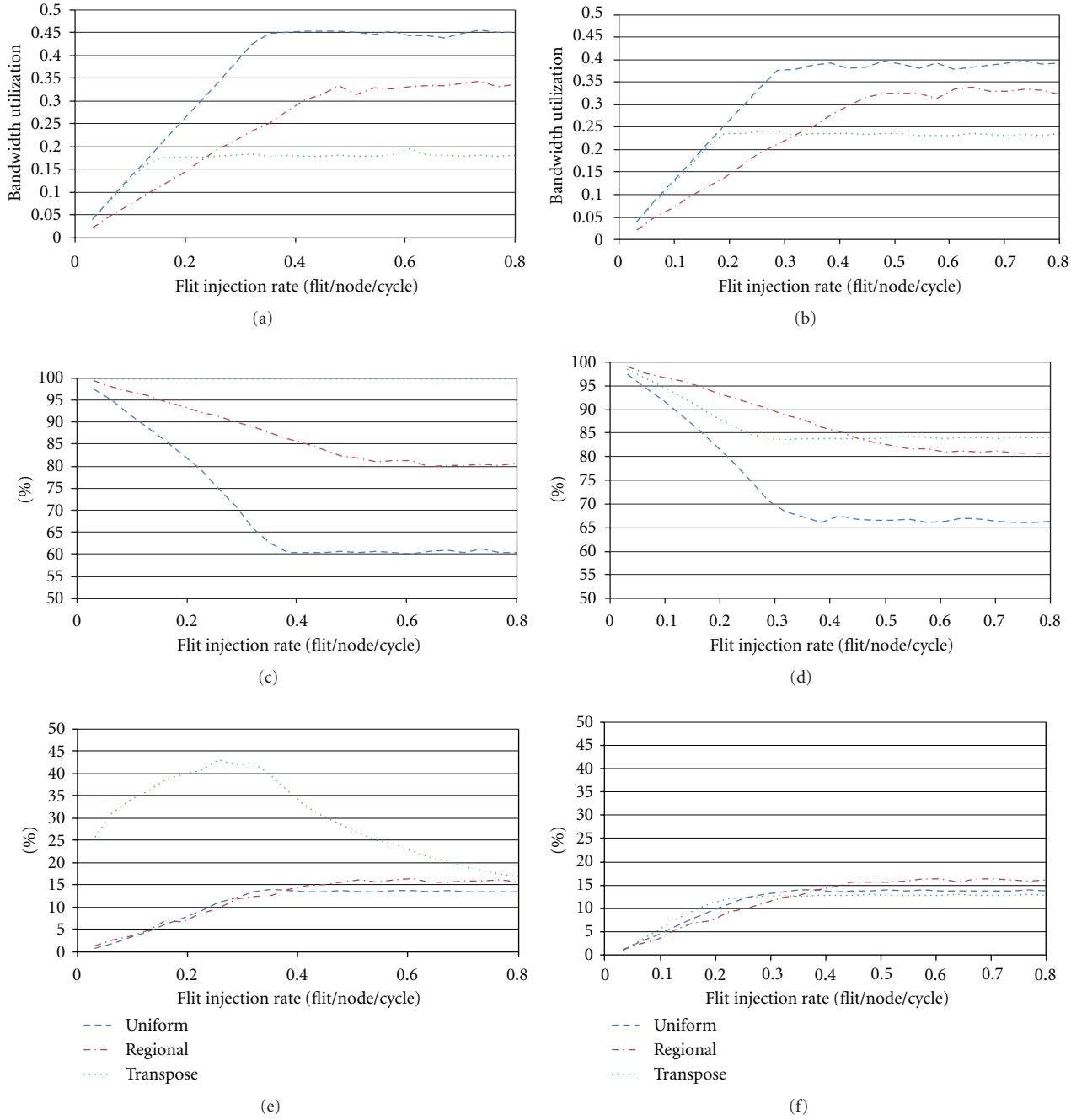


Figure 8: Bandwidth utilization analysis of a conventional NoC router.

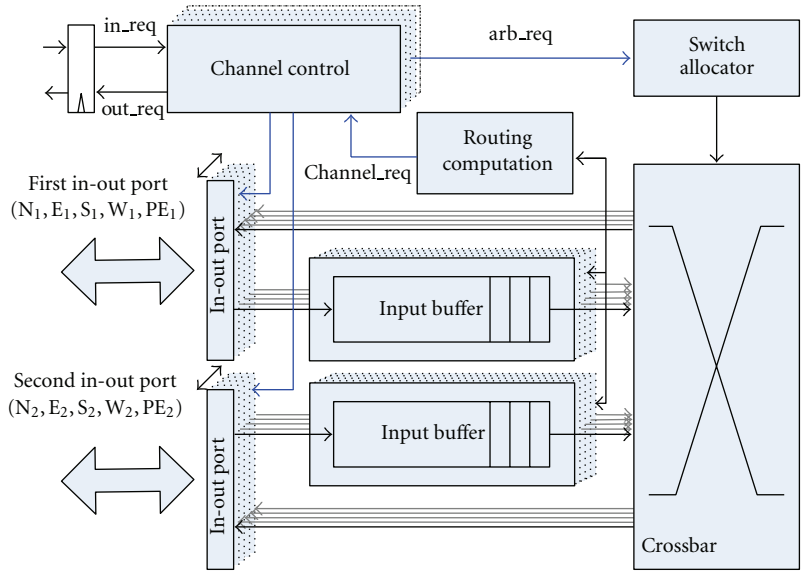


Figure 9: Proposed BiNoC router with wormhole flow control.

of all situations where exactly a single channel is busy, is plotted. An important observation is that, for large traffic volume, this situation happens for about 15% of time despite the type of traffic patterns or the routing methods.

Figure 8 gives ample evidence that the unidirectional channel structure of current NoC cannot fully utilize available resources (channel bandwidth) and may cause longer latency. This observation motivates us to explore the BiNoC architecture that offers the opportunity to reverse channel direction dynamically to relieve the high traffic volume of a busy channel in the opposite direction.

4.4. Design Requirements. Bidirectional channels have been incorporated into off-chip multiprocessor high-speed interconnecting subsystems over years. Recently, bidirectional on-chip global interconnecting subsystems have also been studied quite extensively for supporting electronic design automation of system-on-chip platforms [82–85]. Hence, physical layer design of an NoC channel to support bidirectional data transmission should be of little difficulty. The real challenge of embracing a bidirectional channel in an NoC is to devise a distributed channel-direction-control protocol that would achieve several important performance criteria as follows.

(1) *Correctness*. It should not cause permanent blockage of data transfer (deadlock, starvation) during operation.

(2) *High Performance*. Its performance should be *scalable* to the size of the *NoC* fabric and *robust* with respect to increasing traffic volume. In addition, it is desirable that the performance enhancement can be achieved across different characteristics of application traffic patterns.

(3) *Low Hardware Cost*. The hardware overhead to support the bidirectional channel should be small enough to justify the cost effectiveness of the proposed architecture.

4.5. The Proposed BiNoC Router Design. To realize a dynamically self-reconfigurable bidirectional channel NoC architecture, we initially modified the input/output port configuration and router control unit designs based on the conventional router using wormhole flow control as we proposed in [86]. In order to dynamically adjust the direction of each bidirectional channel at run time, we add a *channel control* module to arbitrate the authority of the channel direction as illustrated in Figure 9.

Each bidirectional channel which is composed of an in-out port inside is the main difference from the conventional router design where unidirectional channel employs a hard-wired input port or output port. However, the total number of data channels is not changed as its applicable bandwidth for each transmission direction is doubled.

In our design, each channel can be used as either an input or an output channel. As a result, the width of a channel request signal, *channel_req*, generated from the RC (routing computation) modules is doubled. Two bidirectional channels can be requested in each output direction. In other words, this router is able to transmit at most two packets to the same direction simultaneously which decreases the probability of contentions.

The channel control module has two major functions. One is to dynamically configure the channel direction between neighboring routers. Since the bidirectional channel is shared by a pair of neighboring routers, every transition of the output authority is achieved by a channel-direction-control protocol between these two routers. The control protocol can be implemented as FSMs. The other responsibility is that whether the channel request (*channel_req*) for the corresponding channel is blocked or not will depend on the current status of channel direction. If the channel is able to be used, the *arb_req* will be sent to the switch allocator (SA) to process the channel allocation.

The most important point of this architecture is that we can replace all the unidirectional channels in a conventional

NoC with our bidirectional channels. That will increase the channel utilization flexibility without requiring additional transmission bandwidth compared to the conventional NoC.

5. Conclusion

In the first part of this paper, we introduced the detail of an on-chip interconnection framework, namely, network on chip (NoC), used in the design of multiprocessor system-on-chip (MPSoC) and chip multiprocessor (CMP) architectures. Then, the NoC architecture and its function layers were reviewed, and some prevalent NoC design methodologies were discussed. Last, we proposed a novel bidirectional channel NoC (BiNoC) backbone architecture, which can be easily integrated into most conventional NoC designs and successfully improve the NoC performance with a reasonable cost.

Acknowledgments

This work was partially supported by the National Science Council, under Grants 99-2220-E-002-041 and 100-2220-E-002-012.

References

[1] F. N. Najm, "Survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, 1994.

[2] H. O. Ron, K. W. Mai, and A. Fellow, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, 2001.

[3] ARM, *AMBA Specification Rev 2.0*, ARM Limited, 1999.

[4] IBM, *32-bit Processor Local Bus Architecture Specification Version 2.9*, IBM Corporation.

[5] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Transactions on Computers*, vol. 35, no. 1, pp. 70–78, 2002.

[6] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, Waltham, Mass, USA, 2004.

[7] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[8] M. Kistler, M. Perrone, and F. Petrini, "Cell multiprocessor communication network: built for speed," *IEEE Micro*, vol. 26, no. 3, pp. 10–23, 2006.

[9] L. Seiler, D. Carmean, E. Sprangle et al., "Larrabee: a many-core x86 architecture for visual computing," *IEEE Micro*, vol. 29, no. 1, pp. 10–21, 2009.

[10] D. Wentzlaff, P. Griffin, H. Hoffmann et al., "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, 2007.

[11] J. Howard, S. Dighe, Y. Hoskote et al., "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers, (ISSCC '10)*, pp. 108–109, San Francisco, Calif, USA, February 2010.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, Calif, USA, 1979.

[13] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer*, vol. 28, no. 1, pp. 3–21, 2009.

[14] G. DeMicheli and L. Benini, *Networks on Chips: Technology and Tools*, Morgan Kaufmann, Waltham, Mass, USA, 2006.

[15] S. Kumar, A. Jantsch, and J. P. Soininen, "Network-on-chip architecture and design methodology," in *Proceedings of the International Symposium on Very Large Scale Integration*, pp. 105–112, April 2000.

[16] C. Grecu, M. Jones, P. P. Pande, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, 2005.

[17] A. M. Rahmani, M. Daneshtalab, A. Afzai-Kusha, S. Safari, and M. Pedram, "Forecasting-based dynamic virtual channels allocation for power optimization of network-on-chips," in *Proceedings of the 22nd International Conference on VLSI Design—Held Jointly with 7th International Conference on Embedded Systems*, pp. 151–156, New Delhi, India, January 2009.

[18] N. Kavaldjiev, G. J. M. Smit, and P. G. Jansen, "A virtual channel router for on-chip networks," in *Proceedings of the IEEE International SOC Conference*, pp. 289–293, September 2004.

[19] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, 1992.

[20] E. Rijpkema, K. G. W. Goossens, and A. Radulescu, "Trade-offs in the design of a router with both guaranteed and best-effort services for networks-on-chip," in *Proceedings of the Design Automation and Test in Europe Conference*, pp. 350–355, March 2003.

[21] H. S. Wang, L. S. Peh, and S. Malik, "A power model for routers: modeling alpha 21364 and InfiniBand routers," *IEEE Micro*, vol. 23, no. 1, pp. 26–35, 2003.

[22] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proceedings of the 31st Annual International Symposium on Computer Architecture*, pp. 188–197, June 2004.

[23] K. Kim, S. J. Lee, K. Lee, and H. J. Yoo, "An arbitration look-ahead scheme for reducing end-to-end latency in networks on chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems, (ISCAS '05)*, pp. 2357–2360, May 2005.

[24] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the Design Automation and Test in Europe Conference*, pp. 250–256, March 2000.

[25] R. Hegde and N. R. Shanbhag, "Toward achieving energy efficiency in presence of deep submicron noise," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 4, pp. 379–391, 2000.

[26] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.

[27] N. Cohen, T. S. Sriram, N. Leland, D. Moyer, S. Butler, and R. Flatley, "Soft error considerations for deep-submicron CMOS circuit applications," in *Proceedings of the IEEE International Devices Meeting, (IEDM '99)*, pp. 315–318, December 1999.

[28] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proceedings of the International Conference on Dependable Systems and Networks, (DNS '02)*, pp. 389–398, June 2002.

[29] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, no. 4, pp. 187–196, 1986.

[30] P. Kermani and L. Kleinrock, "Virtual cut-through: a new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, 1979.

[31] L. S. Peh, W. J. Dally, and P. Li-Shiuan, "Delay model for router microarchitectures," *IEEE Micro*, vol. 21, no. 1, pp. 26–34, 2001.

[32] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, 1987.

[33] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Routing table minimization for irregular mesh NoCs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pp. 1–6, Nice, France, April 2007.

[34] M. A. Yazdi, M. Modarressi, and H. Sarbazi-Azad, "A load-balanced routing scheme for NoC-based systems-on-chip," in *Proceedings of the 1st Workshop on Hardware and Software Implementation and Control of Distributed MEMS, (DMEMS '10)*, pp. 72–77, Besan, TBD, France, June 2010.

[35] M. Daneshtalab, A. A. Kusha, A. Sobhani, Z. Navabi, M. D. Mottaghi, and O. Fatemi, "Ant colony based routing architecture for minimizing hot spots in NOCs," in *Proceedings of the Annual Symposium on Integrated Circuits and System Design*, pp. 56–61, September 2006.

[36] J. Hu and R. Marculescu, "DyAD—smart routing for networks-on-chip," in *Proceedings of the 41st Design Automation Conference*, pp. 260–263, June 2004.

[37] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *Journal of the ACM*, vol. 41, no. 5, pp. 874–902, 1994.

[38] G. M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, 2000.

[39] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Neighbors-on-path: a new selection strategy for on-chip networks," in *Proceedings of the IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia, (ESTIMEDIA '06)*, pp. 79–84, Seoul, Korea, October 2006.

[40] M. Li, Q. A. Zeng, and W. B. Jone, "DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of the Design Automation Conference*, pp. 849–852, July 2006.

[41] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network-on-chip," in *Proceedings of the Design Automation and Test in Europe Conference*, pp. 1126–1127, December 2003.

[42] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of the 42nd Design Automation Conference, (DAC '05)*, pp. 559–564, June 2005.

[43] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, "Improving routing efficiency for network-on-chip through contention-aware input selection," in *Proceedings of the Asia and South Pacific Design Automation Conference, (ASP-DAC '06)*, pp. 36–41, January 2006.

[44] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, (DATE '04)*, pp. 890–895, February 2004.

[45] K. Goossens, J. Dielissen, and A. Rădulescu, "The Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[46] P. Vellanki, N. Banerjee, and K. S. Chatha, "Quality-of-service and error control techniques for mesh-based network-on-chip architectures," *ACM Very Large Scale Integration Journal*, vol. 38, no. 3, pp. 353–382, 2005.

[47] N. Kavaldjiev, G. J. M. Smit, P. G. Jansen, and P. T. Wolkotte, "A virtual channel network-on-chip for GT and BE traffic," in *Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, pp. 211–216, Karlsruhe, Germany, March 2006.

[48] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, no. 2-3, pp. 105–128, 2004.

[49] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs," in *Proceedings of the 21st International Conference on Computer Design, (ICCD '03)*, pp. 536–539, October 2003.

[50] D. Bertozzi and L. Benini, "Xpipes: a network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, pp. 18–31, 2004.

[51] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proceedings of the Design, Automation and Test in Europe, (DATE '05)*, pp. 1226–1231, March 2005.

[52] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, "Providing QoS to connection-less packet-switched NoC by implementing diffServ functionalities," in *Proceedings of the International Symposium on System-on-Chip*, pp. 37–40, November 2004.

[53] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, "Evaluation of current QoS mechanisms in networks on chip," in *Proceedings of the International Symposium on System-on-Chip, (SOC '06)*, pp. 1–4, Tampere, Finland, November 2006.

[54] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Efficient link capacity and QoS design for network-on-chip," in *Proceedings of the Design, Automation and Test in Europe, (DATE '06)*, pp. 1–6, March 2006.

[55] P. Vellanki, N. Banerjee, and K. S. Chatha, "Quality-of-service and error control techniques for network-on-chip architectures," in *Proceedings of the ACM Great lakes Symposium on VLSI, (GLSVLSI '04)*, pp. 45–50, April 2004.

[56] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, "Quantitative modelling and comparison of communication schemes to guarantee quality-of-service in networks-on-chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems, (ISCAS '05)*, pp. 1782–1785, May 2005.

[57] P. Bogdan, T. Dumitras, and R. Marculescu, "Stochastic communication: a new paradigm for fault tolerant networks on chip," *VLSI Design*, vol. 2007, Article ID 95348, 17 pages, 2007.

[58] M. Ali, M. Welzl, S. Hessler, and S. Hellebrand, "A fault tolerant mechanism for handling permanent and transient failures in a network on chip," in *Proceedings of the 4th International Conference on Information Technology-New Generations, (ITNG '07)*, pp. 1027–1032, Las Vegas, Nev, USA, April 2007.

[59] M. Yang, T. Li, Y. Jiang, and Y. Yang, "Fault-tolerant routing schemes in RDT(2,2,1)/α-based interconnection network for networks-on-chip designs," in *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks, (I-SPAN '05)*, pp. 1–6, December 2005.

[60] T. Lehtonen, P. Liljeberg, and J. Plosila, "Online reconfigurable self-timed links for fault tolerant NoC," *VLSI Design*, vol. 2007, Article ID 94676, 13 pages, 2007.

[61] H. Kariniemi and J. Nurmi, "Fault-tolerant XGFT network-on-chip for multi-processor system-on-chip circuits," in *Proceedings of the International Conference on Field Programmable Logic and Applications, (FPL '05)*, pp. 203–210, August 2005.

[62] T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel, "Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures," in *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, (DSD '07)*, pp. 527–534, Lübeck, Germany, August 2007.

[63] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring fault-tolerant network-on-chip architectures," in *Proceedings of the 2006 International Conference on Dependable Systems and Networks, (DSN '06)*, pp. 93–104, Philadelphia, Pa, USA, June 2006.

[64] Y. Hatanaka, M. Nakamura, Y. Kakuda, and T. Kikuno, "A synthesis method for fault-tolerant and flexible multipath routing protocols," in *Proceedings of the International Conference on Engineering of Complex Computer Systems*, pp. 96–105, September 1997.

[65] W. Stallings, *Data and Computer Communications*, Prentice Hall, New York, NY, USA, 2007.

[66] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 225–232, January 2003.

[67] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pp. 46–51, February 2004.

[68] J. Hoffman, D. A. Ilitzky, A. Chun, and A. Chapyzhenka, "Architecture of the scalable communications core," in *Proceedings of the First International Symposium on Networks-on-Chip, (NOCS '07)*, pp. 40–49, Princeton, NJ, USA, May 2007.

[69] E. S. H. Hou, N. Ansari, and H. Ren, "Genetic algorithm for multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 2, pp. 113–120, 1994.

[70] C. M. Krishna and K. G. Shin, *Real-Time Systems*, WCB/McGraw Hill, New York, NY, USA, 1997.

[71] H. El-Rewini, H. H. Ali, and T. Lewis, "Task scheduling in multiprocessing systems," *Computer*, vol. 28, no. 12, pp. 27–37, 1995.

[72] T. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," in *Proceedings of the Hawaii International Conference on System Sciences*, pp. 288–297, January 1995.

[73] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," in *Proceedings of the 38th Design Automation Conference*, pp. 828–833, June 2001.

[74] M. T. Schmitz and B. M. Al-Hashimi, "Considering power variations of DVS processing elements for energy minimisation in distributed systems," in *Proceedings of the 14th International Symposium on System Synthesis (ISSS '01)*, pp. 250–255, October 2001.

[75] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 514–521, March 2002.

[76] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "Iterative schedule optimization for voltage scalable distributed embedded systems," *ACM TECS*, vol. 3, no. 1, pp. 182–217, 2004.

[77] V. Kianzad, S. S. Bhattacharyya, and G. Qu, "CASPER: an integrated energy-driven approach for task graph scheduling on distributed embedded systems," in *Proceedings of the IEEE 16th International Conference on Application-Specific Systems, Architectures, and Processors, (ASAP '05)*, pp. 191–197, July 2005.

[78] P. C. Chang, I. W. Wu, J. J. Shann, and C. P. Chung, "ETAHM: an energy-aware task allocation algorithm for heterogeneous multiprocessor," in *Proceedings of the 45th Design Automation Conference, (DAC '08)*, pp. 776–779, Anaheim, Calif, USA, June 2008.

[79] Y. Zhang, X. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," in *Proceedings of the 39th Design Automation Conference*, pp. 183–188, June 2002.

[80] G. Varatkar and R. Marculescu, "Communication-aware task scheduling and voltage selection for total systems energy minimization," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design, (ICCAD '03)*, pp. 510–517, November 2003.

[81] J. Hu and R. Marculescu, "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, (DATE '04)*, pp. 234–239, February 2004.

[82] J. Lillis and C. K. Cheng, "Timing optimization for multisource nets: characterization and optimal repeater insertion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 2-3, pp. 322–331, 1999.

[83] S. Bobba and I. N. Haj, "High-performance bidirectional repeaters," in *Proceedings of the Great Lakes Symposium on Very Large Scale Integration*, pp. 53–58, March 2000.

[84] A. Nalamalpu, S. Srinivasan, and W. P. Burleson, "Boosters for driving long onchip interconnects—design issues, interconnect synthesis, and comparison with repeaters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 1, pp. 50–62, 2002.

[85] H. Ito, M. Kimura, K. Miyashita, T. Ishii, K. Okada, and K. Masu, "A bidirectional- and multi-drop-transmission-line interconnect for multipoint-to-multipoint on-chip communications," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 1020–1029, 2008.

[86] Y. C. Lan, S. H. Lo, Y. C. Lin, Y. H. Hu, and S. J. Chen, "BiNoC: a bidirectional NoC architecture with dynamic self-reconfigurable channel," in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip, (NoCS '09)*, pp. 266–275, May 2009.