Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2015, Article ID 862387, 16 pages
http://dx.doi.org/10.1155/2015/862387

Research Article

Applying Partial Power-Gating to Direction-Sliced Network-on-Chip

Feng Wang, Xiantuo Tang, and Zuocheng Xing

National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Feng Wang; kikiwang007@gmail.com

Received 23 March 2015; Revised 30 June 2015; Accepted 26 July 2015

Academic Editor: Chi-Ying Tsui

Network-on-Chip (NoC) is one of critical communication architectures for future many-core systems. As technology is continually scaling down, on-chip network meets the increasing leakage power crisis. As a leakage power mitigation technique, power-gating can be utilized in on-chip network to solve the crisis. However, the network performance is severely affected by the disconnection in the conventional power-gated NoC. In this paper, we propose a novel partial power-gating approach to improve the performance in the power-gated NoC. The approach mainly involves a direction-slicing scheme, an improved routing algorithm, and a deadlock recovery mechanism. In the synthetic traffic simulation, the proposed design shows favorable power-efficiency at low-load range and achieves better performance than the conventional power-gated one. For the application trace simulation, the design in the mesh/torus network consumes 15.2%/18.9% more power on average, whereas it can averagely obtain 45.0%/28.7% performance improvement compared with the conventional power-gated design. On balance, the proposed design with partial power-gating has a better tradeoff between performance and power-efficiency.

1. Introduction

In recent years, Network-on-Chip has been proposed as the mainstream architecture to connect various on-chip resources in many-core systems, such as chip-multiprocessors (CMPs) and multiprocessor system-on-chips (MPSoCs). As in most other VLSI designs, power-efficiency has also been one of the critical constraints in current NoC design. The power situation of NoC design will become much more serious as the technology is continually scaling down and the working frequency is increasing. For example, the percentage of router power reaches up to 28% in Intel Teraflop chip [1] and 19% in 36-core SCORPIO chip [2]. The overall router power-consumption consists of dynamic switching power and static leakage power. The dynamic power is consumed when the router transfers packets and its clock circuits switch periodically. However, there is significant consumption of static leakage power even without any packet transfers as long as the router is power-on. In recent multi/many-core systems, the leakage power has already become a major portion of power dissipation, and its proportion will further increase along with technology parameter scaling down. In the simulation of Samih et al. [3], the leakage power increases rapidly, from 11.2% of the total router power-consumption in the 65 nm technology to 33.6% in the 32 nm technology, when working at 1.1 V voltage and 2.0 GHz frequency.

In a many-core system, the demand of network throughput may be quite low in most of execution time and rarely reaches the peak/saturation point. Actually, real-world applications exhibit sparse and bursty network traffic [4, 5]: only few communication-intensive phases that consume large network bandwidth and other phases that inject few packets into the network. Hence, there are many opportunities in which some routers and links keep in idleness, and they can be utilized to solve leakage power crisis. As a representative technique to mitigate leakage power, power-gating controls the power supply of the gated circuit blocks by switching the transistors which are inserted between the VDD and the gated blocks. According to the traffic characteristic, some recent researches have employed power-gating in current NoC design to mitigate the increasing leakage power, which will be introduced in the next section. Besides, clock network also consumes a great deal of chip dynamic power because

the clock is fed to most of sequential circuit blocks on chip, and the clock must switch periodically. As a well-known low-power technique, clock-gating is utilized to reduce chip dynamic power [6] and has been implemented in recent NoC designs [1, 7, 8]. The Intel Teraflop chip [1] uses the multilevel clock-gating policy and the sleep transistor circuits to reduce both dynamic and leakage power, and it is controlled at full-chip, tile-slice, and individual tile levels based on workloads. In another recent clock-gating design, Mullins [7] applies clock-gating to the on-chip routers at two levels: local clock-gating and router level clock-gating.

As more cores are integrated, the requirement of low on-chip latency will become even more pronounced so as not to impact system performance. However, due to the disconnection problem, an injected packet which meets a sleeping downstream router has to suspend transfer and wait for the sleeping router being awakened in the conventional power-gated NoC. Then successive wake-up delays may be inevitably incurred to the packet. Aiming at the disconnection problem, we propose partial power-gating approach to obviate the performance decline in the power-gated NoC. This approach is composed of a direction-slicing scheme, an improved routing algorithm, and a deadlock recovery mechanism. The direction-slicing scheme divides each router into two parts (i.e., two slices) and keeps one slice always-on to construct an active subnet for providing a fully connected path while applying power-gating to another for saving possible leakage power. Meanwhile, the routing algorithm is improved to support packet transfers when the gated slice switches its power supply in power-on/off status. Based on improved architecture and routing algorithm, a novel deadlock recovery mechanism is adopted to solve deadlock in our sliced network. For the recovery mechanism, a deadlock escape path is constructed in the network interface through adding some additional components and links. Furthermore, clock-gating is utilized in the gated slice to save partial clock power. This direction-sliced NoC with partial power- and clock-gating can achieve the purpose of power-saving and has less impact on performance than the conventional power-gated design.

The rest of this paper is organized as follows. Section 2 summarizes some related works and expatiates on our motivation. Section 3 explains the details of the direction-sliced NoC with partial power-gating. Section 4 presents our evaluation methodology and discusses the simulation results. Finally, we conclude this paper in Section 5.

2. Related Work and Motivation

Although low-power can be achieved by reducing the overhead of router resources in current NoC design (such as the bufferless router [9]), the low-cost NoC still has both obvious performance loss and limited power decline. Power-gating is a well-known technique to mitigate leakage power in low-power VLSI design, especially for circuit blocks that exhibit enough idleness. Several recent studies have adopted power-gating technique in NoC design. Since router buffers consume the largest portion of the NoC's leakage power, power-gating is utilized to reduce buffer leakage power in [10]. Instead of turning the whole buffer off, it keeps partial buffer active to store incoming flits at all times and then avoids negative performance affection. Besides adopting power-gating on router buffers, Matsutani et al. [11] proposed an ultrafine-grained power-gating policy to control the power supply of all router components. This policy based on look-ahead routing can detect the arrival of packets two hops ahead so as to hide partial wake-up delay.

Unlike component-based power-gating, some other studies prefer to apply power-gating to entire router (router level power-gating). A router parking method is proposed in [3] to power-gate routers when the connected core is idle, but it needs to flush private caches before turning off routers, which may cause serious performance decrease. The node-router disconnection referred to by [12] severely limits power-gating being effectively used in on-chip routers due to the limitation of break-even time, long wake-up delay. The problem is that a packet injected by an active core/sending-router may meet a sleeping receiving-router and has to suspend transfer and wait for the sleeping receiver being awakened due to the disconnection between the active sender and the sleeping receiver. This disconnection situation makes conventional power-gating unprofitable for the high-performance NoC design.

Originating the idea of guaranteeing full connectivity, these researches [12–14] provide a fully connected path to avoid the disconnection problem. The NoRD [12] introduces a bypass ring path to connect all nodes completely; then packets can be transmitted in the bypass ring to avoid long wake-up latency when meeting a sleeping router. However, The packets' average latency brought by the bypass ring may inevitably increase proportionately with the square of network scale. As a low-power instance in the multiple networks, the Catnap [13] guarantees full connectivity by keeping one subnet always active, while keeping other subnets power-gated to reduce unnecessary leakage power. Profiting from the excellent path diversity in the Clos network, the MP3 [14] utilizes minimal router resources and links to ensure all nodes are fully connected, and other remaining full-routers and partial-routers can be power-gated to achieve leakage power-saving. However, the Catnap merely increases the efficiency of power-gating in the architecture of multiple networks, and the MP3 is only applicable to Clos and other indirect networks.

Three typical network slicing schemes are proposed in [15], channel-slicing, dimension-slicing, and bit-slicing, which slice each network node across multiple chips or modules to alleviate pin and area limitations. For the channel-slicing, a w-bit-wide node is split into k independent slices, each of which has w/k-bit-wide channels. There are not any connections between these slices. For the dimension-slicing, the network node is sliced based on ports' dimensions; those ports associated with one dimension are contained in an appointed slice. Besides, additional data channels must be added between the slices to communicate with each other. Finally, for the bit-slicing, a w-bit-wide node is divided across k w/k-bit-wide slices; each slice packaged in a separate module contains a w/k-bit-wide portion of the router

datapath. Unlike the channel-slicing, control information must be distributed to all slices so as to act in unison. Except the three slicing schemes above, we propose a new slicing scheme named direction-slicing for bidirectional network. The direction-slicing divides the router into two slices based on the directions in a dimension (input or output direction), and each slice only has a unidirectional channel in each dimension. How to make direction-slicing will be shown in Section 3.1.

In order to avoid the disconnection issue in the conventional power-gated NoC, as a fire-new viewpoint, we start to investigate network-slicing and power-gating in combination. Essentially, the Catnap [13] is a commendable instance of applying power-gating to channel-sliced network (multiple network is an instance of channel-slicing). For the dimension-slicing, it is hard to construct a sliced subnet to guarantee full connectivity; then it is not fit for the intention of avoiding disconnection. Applying power-gating to bit-sliced network has been wri tten in our other paper, and we do not introduce it at present. When utilizing power-gating into the direction-sliced network, we split the router into two different power domain slices across channels' input or output directions, since each channel is bidirectional. In this sliced network, we can maintain the power domain of some slices being ever-on to guarantee full connectivity and adopt power-gating for other slices to save some unnecessary leakage power. The detail of employing power-gating to the sliced network will be introduced in the following section.

3. Low-Power Direction-Sliced NoC Design

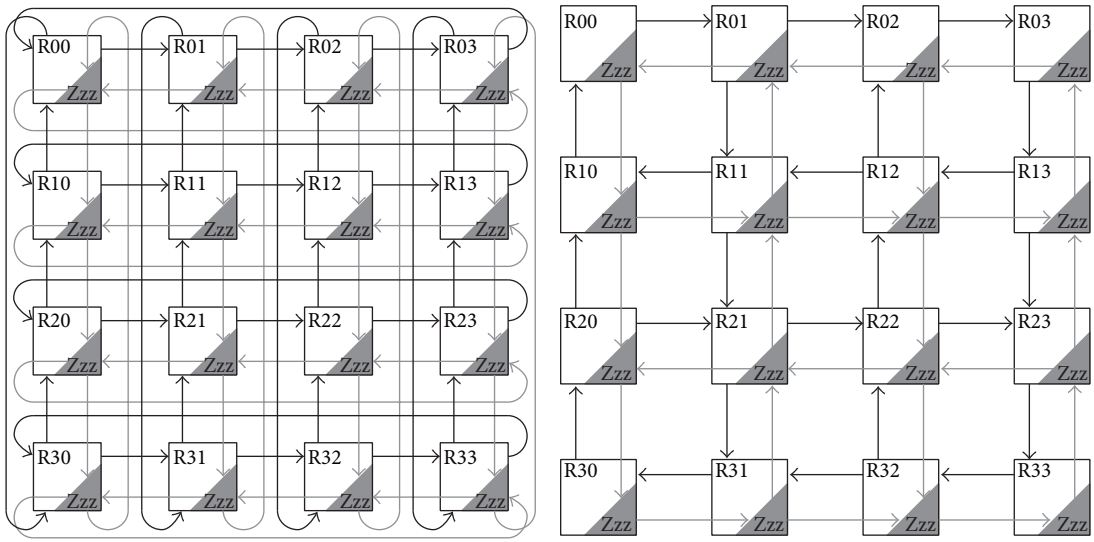
In this section, we first propose the direction-sliced architecture to suit for partial power-gating scheme. Next, we improve routing algorithm to support packets transmitting and design a deadlock recovery mechanism to cope with the routing improvement. Finally, we describe the detail of partial power-gating scheme for the sliced network.

3.1. Direction-Sliced Architecture. We employ the direction-slicing scheme into the 2D torus and mesh network, respectively. Figure 1(a) shows the direction-slicing in the 2D torus network. All $X+$ channels (*West_In* and *East_Out*), $Y-$ channels (*South_In* and *North_Out*), and *Local* channels in each router are split into ever-on slices and form the ever-on subnet to guarantee full connectivity, while remaining $X-$ channels (*East_In* and *West_Out*) and $Y+$ channels (*North_In* and *South_Out*) are the gated slices. In contrast with the 2D torus, Figure 1(b) shows the different direction-slicing in the 2D mesh network. Considering both symmetrical topology and simple routing design, all $X+$ channels in even rows, $X-$ channels in odd rows, $Y-$ channels in even columns, $Y+$ channels in odd columns, and *Local* channels are utilized to constitute the ever-on subnet, and other remaining channels which belong to gated slices can be power-gated to pursue the leakage power-saving.

Whether in the torus or mesh, each router is split into two power domains across input and output channels, and corresponding VC buffers, crossbars, and output latches are also divided into the two power domains, as shown in Figure 2(a). Since router's leakage power is mainly consumed by VC buffers, crossbars, and output latches (about 76.3% of leakage power in our simulation), it may cut down a lot of leakage power if 2/5 router resources are power-gated. In current power-gating technique, there are hardware overheads for the gating switches and the distribution of the gating signal wires; the area overheads of the well-designed power-gating blocks are typically within 4–10% depending on the circuit level optimizations [16, 17]. Then our sliced design only increases about 1.6–4.0% of hardware overhead owing to the partial power-gating. Moreover, in our simulation, all flip-flops and links in clock networks consume about 39.2% of total power when in zero-load. Then, except for adopting partial power-gating, we also adopt clock-gating technique to save flip-flops' dynamic power in clock networks when corresponding gated slice is powered-off, and this method is similar to the router-level clock-gating in [7]. Because the resumption period of clock-gating is far less than the wake-up period of power-gating, we can reuse the wake-up signal as the early valid signal to resume the clock input and the power supply, as shown in Figure 2(b). The latch in clock-gating circuit can be used to prevent glitches on the control signal from propagating to the gated slice's clock pin [18]. Eliminating the latch can slightly reduce power dissipation and area cost, whereas the latch-free design has a significant drawback: if the control signal is not stable at its new value before the falling clock edge, glitches on the control signal can corrupt the clock signal to the gated slice.

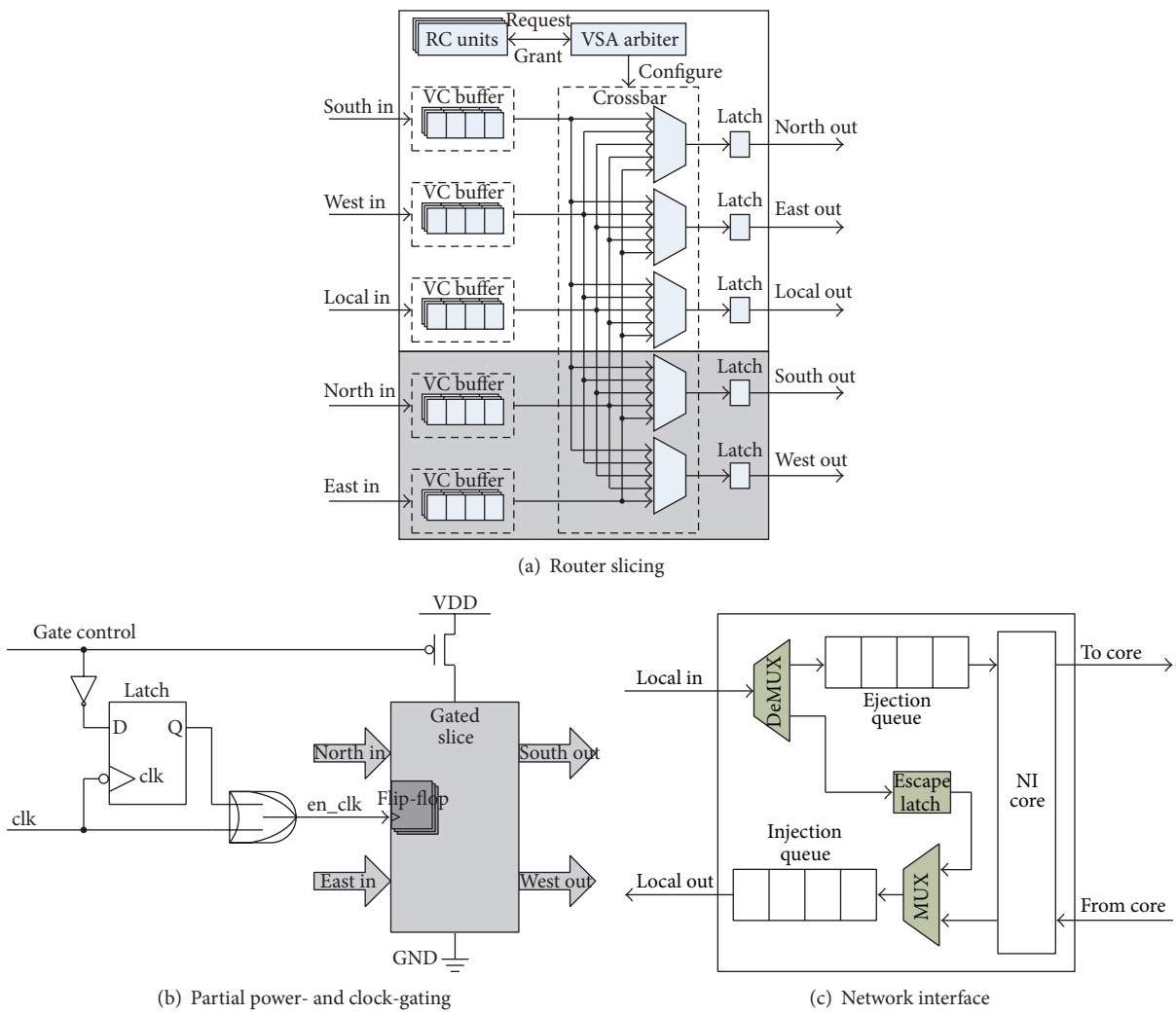
In the on-/off-chip network, deadlock situation can be avoided by wise routing design, such as *XY* Dimension Ordered Routing (DOR-*XY*) in the 2D normal mesh network. However, the ever-on subnet in the sliced mesh may inherently lead to deadlock, since the restricted unidirectional channel combined with minimal path routing may form some resource circular dependencies. In the bidirectional or unidirectional torus, all channels in a row/column form a closed ring, and packets routed in the ring may encounter a deadlock caused by the circular dependency. Then, we improve network interface (NI) structure to achieve deadlock recovery, add an escape path in NI to release deadlock situation, and retransmit blocked packets from the local input port. The escape latch is used to temporarily buffer the blocked packets when the recover process is activated. As shown in Figure 2(c), three modules (escape latch, multiplexer, and demultiplexer) are added to implement an escape path. In our area evaluation, the added modules only increase 2.2% of total router area (not including circuit blocks for power-gating and clock-gating). Both resource dependence situation and deadlock recovery mechanism will be introduced in Section 3.4.

3.2. Routing Algorithm. If a sliced router is partially on, some input/output ports should be forbidden to be utilized. Then, the routing algorithm should be fit for two cases. On the one hand, it needs to support packets routing on the ever-on subnet when the gated slice is powered-off; on the other hand, it must be able to utilize all possible ports when



(a) 2D torus (b) 2D mesh

Figure 1: Direction-sliced torus and mesh network.



(a) Router slicing

(b) Partial power- and clock-gating (c) Network interface

Figure 2: Direction-sliced router architecture for partial power-gating and clock-gating.

Input: *cur* - current node coordinate, *dest* - destination node coordinate, *pre_dir* - preceding routing direction;
Output: *dir* - output direction;
(1) **begin**
(2) $\Delta x \leftarrow (dest.x - cur.x)$;
(3) $\Delta y \leftarrow (dest.y - cur.y)$;
(4) **if** *Sliced router is partially-on* **then**
(5) **if** $\Delta x \neq 0$ **then**
(6) $dir \leftarrow X+$;
(7) **else**
(8) $dir \leftarrow \Delta y \neq 0$ **?** $Y-$ **:** *Local*;
(9) **else if** *Sliced router is fully-on* **then**
(10) **if** $(\Delta x > 0$ ***and*** $\Delta x \leq Dim_x/2)$ ***or*** $\Delta x \leq -Dim_x/2$ **then**
(11) $dir \leftarrow X+$;
(12) **else if** $(\Delta x < 0$ ***and*** $\Delta x > -Dim_x/2)$ ***or*** $\Delta x > Dim_x/2$ **then**
(13) $dir \leftarrow X-$;
(14) **else**
(15) **if** $(\Delta y > 0$ ***and*** $\Delta y < Dim_y/2)$ ***or*** $\Delta y < -Dim_y/2$ **then**
(16) $dir \leftarrow Y+$;
(17) **else if** $(\Delta y < 0$ ***and*** $\Delta y \leq -Dim_y/2)$ ***or*** $\Delta y \geq Dim_y/2$ **then**
(18) $dir \leftarrow Y-$;
(19) **else**
(20) $dir \leftarrow Local$;
(21) **if** $(pre\_dir = X-$ ***and*** $dir = X+)$ ***or*** $(pre\_dir = Y+$ ***and*** $dir = Y-)$ **then**
(22) $dir \leftarrow pre\_dir$;
(23) **return** *dir*;

Algorithm 1: Routing algorithm for the sliced 2D torus network.

the sliced router is fully on. For the convenience of description, we use the abbreviation *UniMesh/UniTorus* to indicate the ever-on subnet in the sliced mesh/torus when all gated slices are deactivated.

The UniTorus still has perfect symmetry, and its routing algorithm can be simply designed when all sliced routers are partially on. At first, packets can be routed in the X-dimension unidirectional ring until reaching the node which locates in the same Y-dimension ring with the destination node. Then, packets should make a dimension turn and be transmitted in the Y-dimension unidirectional ring until reaching the destination node. If the gated slice has been awakened, packets can be switched to any possible output port by utilizing normal DOR-XY algorithm. The routing algorithm of the sliced 2D torus is shown in Algorithm 1. Noteworthily, the algorithm should forbid routing from the gated direction to the ever-on direction within the same dimension, so that the forward and backward routing, which may happen repeatedly before the gated slice is activated, can be avoided. When a packet is waiting for the gated slice to be awakened, we can simply solve the issue by routing the packet to its preceding direction. The solution is illustrated at lines 21-22 in the pseudocode of Algorithm 1.

The routing algorithm in the sliced mesh is also composed of two parts, which correspond to the partially on or fully on situation in a router. We still adopt normal DOR-XY algorithm as one algorithm part when gated slice has been awakened. However, the other algorithm part is more complicated than the part in the sliced torus. In the UniMesh, due to lack of other half reversed channels, we cannot give a minimal routing path which only takes *Manhattan Distance* hops like the minimal deterministic routing in a normal mesh. Sometimes, packet should take some additional hops to reach its destination node. Extremely, as shown in Figure 3(a), the DOR-XY routing only needs 4 hops (red dashed arrow line) from source node Src to destination node Dest, while the UniMesh needs 8 hops (blue arrow line) since the pairs of Src and Dest locate in "bad" row and column (i.e., packets between the pairs cannot use minimal path to reach Dest and must detour to the adjacent nodes to find a potential path). It should take two extra detours to search a shorter routing path. One detour is from node Src to node 21, and the other is from node 12 to node Dest.

According to the ever-on subnet in Figure 1(b), we firstly give the X direction higher routing priority than the Y direction to ensure a deterministic routing (or vice versa) and then design a routing algorithm for the partially on router as shown in Algorithm 2. In the algorithm part, we detailedly itemize all possible cases and return a most reasonable routing direction according to the relative position and the parity flag between current and destination node. The parity flag functions *odd()* and *even()* are, respectively, used to distinguish the channel direction with row and column coordinate value for the current and destination node. In despite of network size and relative position between source and destination node, it merely incurs no more than 2 detours or 6 additional hops. The kernel idea of the routing algorithm is taking one or two detours and "borrowing" some adjacent channels. Subsequently, we use three representative examples to explain how the routing algorithm works.

Figures 3(b) and 3(c) represent two typical situations about how to "borrow" the $Y+$ channels. Since Dest locates

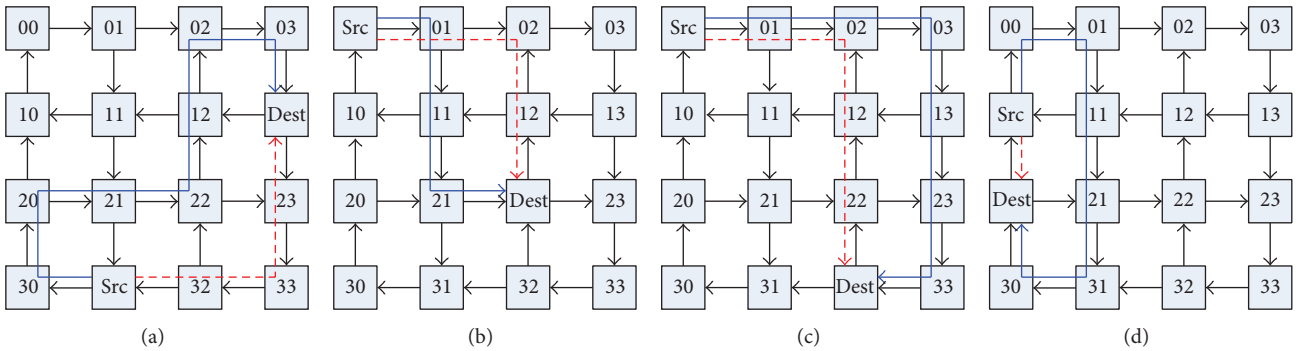


Figure 3: Four routing path examples by using UniMesh routing and normal DOR-XY routing.

Input: *cur* - current node coordinate, *dest* - destination node coordinate;
Output: *dir* - output direction;
// function *even*() and *odd*() distinguish even or odd number;
(1) **begin**
(2) $\Delta x \leftarrow (dest.x - cur.x)$;
(3) $\Delta y \leftarrow (dest.y - cur.y)$;
(4) **if** $\Delta x < 0$ ***and*** $\Delta y < 0$ **then**
(5) **if** $even(cur.y)$ **then**
(6) $dir \leftarrow odd(cur.x)$ **?** $Y+$ **:** $Y-$;
(7) **else**
(8) $dir \leftarrow (\Delta x = -1$ ***and*** $odd(dest.x)$ ***and*** $odd(dest.y))$ **?** $Y-$ **:** $X-$;
(9) **else if** $\Delta x > 0$ ***and*** $\Delta y > 0$ **then**
(10) **if** $odd(cur.y)$ **then**
(11) $dir \leftarrow odd(cur.x)$ **?** $Y+$ **:** $Y-$;
(12) **else**
(13) $dir \leftarrow (\Delta x = 1$ ***and*** $even(dest.x)$ ***and*** $even(dest.y))$ **?** $Y+$ **:** $X+$;
(14) **else if** $\Delta x > 0$ ***and*** $\Delta y < 0$ **then**
(15) **if** $odd(cur.y)$ **then**
(16) $dir \leftarrow even(cur.x)$ **?** $Y-$ **:** $X-$;
(17) **else**
(18) $dir \leftarrow (\Delta x = 1$ ***and*** $odd(dest.x))$ **?** $Y-$ **:** $X+$;
(19) **else if** $\Delta x < 0$ ***and*** $\Delta y > 0$ **then**
(20) **if** $even(cur.y)$ **then**
(21) $dir \leftarrow odd(cur.x)$ **?** $Y+$ **:** $X+$;
(22) **else**
(23) $dir \leftarrow (\Delta x = -1$ ***and*** $even(dest.x))$ **?** $Y+$ **:** $X-$;
(24) **else if** $\Delta x = 0$ ***and*** $\Delta y > 0$ **then**
(25) $dir \leftarrow odd(cur.x)$ **?** $Y+$ **:** $(even(cur.y)$ **?** $X+$ **:** $(cur.x = 0$ **?** $Y-$ **:** $X-))$;
(26) **else if** $\Delta x = 0$ ***and*** $\Delta y < 0$ **then**
(27) $dir \leftarrow even(cur.x)$ **?** $Y-$ **:** $(odd(cur.y)$ **?** $X-$ **:** $(cur.x = (Dim_x - 1)$ **?** $Y+$ **:** $X+))$;
(28) **else if** $\Delta x > 0$ ***and*** $\Delta y = 0$ **then**
(29) $dir \leftarrow even(cur.y)$ **?** $X+$ **:** $(even(cur.x)$ **?** $Y-$ **:** $(cur.y = (Dim_y - 1)$ **?** $X-$ **:** $Y+))$;
(30) **else if** $\Delta x < 0$ ***and*** $\Delta y = 0$ **then**
(31) $dir \leftarrow odd(cur.y)$ **?** $X-$ **:** $(odd(cur.x)$ **?** $Y+$ **:** $(cur.y = 0$ **?** $X+$ **:** $Y-))$;
(32) **else**
(33) $dir \leftarrow Local$;
(34) **return** dir;

Algorithm 2: Routing algorithm for the partially on router in the sliced 2D mesh network.

in "bad" column (the column only has $Y-$ channels), packets should "borrow" the adjacent column's $Y+$ channels to reach Dest. In Figure 3(b), both Src and Dest own the $X+$ channels, and Src locates in the $X-$ direction of Dest; then packets between the node pairs can use the $Y+$ direction routing in advance (one column ahead of the destination column) when they reach node 11 along the $X+$ channels. The reason is that Dest node's $X+$ input channels locate ahead of the destination column, and then it does not incur any additional hop. And yet, in Figure 3(c), Src owns the $X+$ output channels, Dest

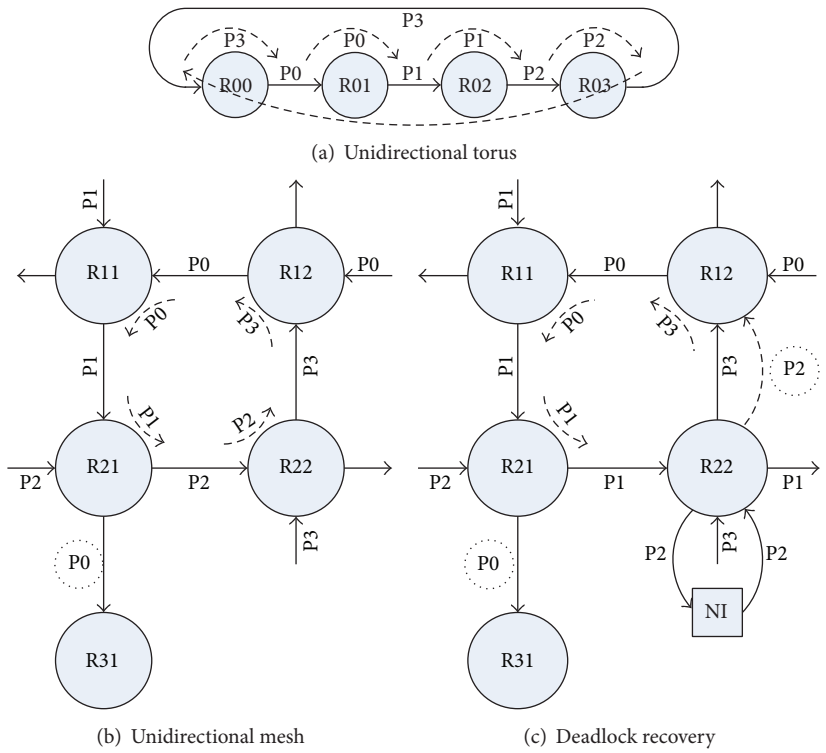


Figure 4: Routing deadlock in unidirectional torus (a) and unidirectional mesh (b), deadlock recovery diagram (c).

owns the $X-$ input channels, and Src locates in the $X-$ direction of Dest; then packets between the node pairs have to transfer in the $Y+$ direction after taking one additional hop in the $X+$ direction. On the contrary, because Dest node's $X-$ input channels locate behind destination column, packets should take an extra detour to reach Dest. How to arbitrate the two situations is illustrated at lines 8, 13, 18, and 23 in the pseudocode of Algorithm 2.

The case in Figure 3(d) represents the situation that Src and Dest locate in the same row/column and do not have straight channels from Src to Dest. In this situation, packets should be routed to the adjacent row/column and "borrow" some channels to approach Dest through a detour path. Since Src locates in "bad" row (i.e., without $X+$ output port) and Dest locates in "bad" row (i.e., without $X-$ input port), packets should make two extra detours to reach Dest. Unlike the situation in Figure 3(a), the two detours cannot provide any chance to make a hop deduction, and 6 additional hops are inevitably incurred. Our pseudocode from lines 24 to 31 is used to resolve how to make a decision in this situation.

3.3. Deadlock Recovery. In a network, circular dependency can occur as each packet holds a resource (i.e., current router's buffer) while requesting another resource (i.e., downstream router's buffer) and results in a routing deadlock [15, 19]. For the normal or sliced 2D torus with DOR-XY, the circular dependency does not occur from the Y-dimension to the X-dimension but may occur within either the X-dimension rings or the Y-dimension rings [20]. Figure 4(a) presents a circular dependency in the X-dimension ring; each packet (P0–P3) holds current router's input channel and wants to request its downstream router's input channel which has been occupied by another packet. For the ever-on subnet in the sliced 2D mesh, since all rows/columns only own the unidirectional channels, the resource circular dependency is hard to be avoided by an unsophisticated routing algorithm. A four-node circular dependency is illustrated in Figure 4(b) to highlight the dependency situation that may occur in the ever-on subnet of the mesh. Each packet (P0–P3) reserves current router's input channel and desires to make a turn to reach downstream router's input channel, while this downstream channel is occupied by a downstream packet and cannot be reallocated for current packet.

Then, we use a deadlock recovery mechanism to solve the resource dependency in our sliced network. There are two key processes to any deadlock recovery mechanism: detection and recovery [15, 20, 21]. The detection process is accomplished with timeout counters as proposed in [15, 21]: a counter is associated with each input channel, which is reset when packets make progress in the input channel. However, if the counter reaches a predetermined threshold (about 32 cycles in our simulation), input channel is considered to be blocked and the recovery process is activated. Our recovery mechanism relied on the improvement of NI architecture, as shown in Figure 2(c). There are some differences between packet-based and flit-based flow control for the implement of our deadlock recovery. In this paper, we only use the deadlock situation in the unidirectional mesh to describe the detail of deadlock recovery mechanism (the dimension ring of the torus has similar recovery mechanism).

At first, we consider the deadlock recovery in the unidirectional mesh with packet-based flow control. When a certain input channel detects a deadlock situation, the whole packets blocked in this channel should be routed to the local output port and should be temporarily buffered in the escape latch. Noteworthily, all ejected packets should have an escape tag to tell the NI multiplexer where to eject (the escape latch or the ejection queue). Afterwards, the escape packet in the escape latch can be reinjected into the injection queue as a normal packet, if the queue is not full and not busy in receiving packets from NI core. Thus, the circular dependency situation is broken. Besides, we give the packets in the escape latch a higher priority than local injected packets to be pushed into the injection queue. And finally, the input channels occupied by the blocked packets can be gradually released for other requesters, and the deadlock issue is resolved. As shown in Figure 4(c), when the $X+$ input channel in node 22 detects a deadlock, current router transmits the escape packet P2 into the local ejection port. Subsequently, P2 is reinjected into the local input port and waits to be transmitted. Due to the banishment of P2, the circular dependency between P0 and P3 is broken, and the blocked packets P1, P0, and P3 can be transmitted to their downstream router in sequence. After P3 releases the $Y-$ input channel in node 12, P2 can be routed to its downstream router from the local port in router 22. Finally, it solves the routing deadlock and avoids the shortcoming of conventional recovery mechanism which needs to discard the congested packet and retransmit it from source node (packet retransmission means more performance loss and power-waste).

However, for the unidirectional mesh with flit-based flow control, each flit composing an escape packet should own an escape tag as well, and two policies must be adopted to keep all flits in order. First, when an escape packet's head flit has started an escape process, current router should prohibit other escape processes in order to protect the ongoing escape process. After the tail flit finishes its escape, current router is resumed to deal with other escape processes. This policy keeps the order of flits between different escape packets owned by different input channels. Second, the escape latch should wait for all flits belonging to the same escape packet to be received, and then it is permitted to push out the whole flits in successive cycles when meeting the injection condition. This policy keeps the order of flits between escape packet and local injected packet.

3.4. Partial Power-Gating Scheme. In our sliced network, partial power-gating can be achieved by switching on/off the power supply of the gated slices; then each gated slice may be in any one of the three power states: *Active*, *Sleep*, and *Wake-up*, as shown in Figure 5. The state transition conditions are needed to arbitrate when a gated slice should switch from one power state to another state. Thus, we select the *Maximum Buffer Occupancy* (MBO) as the congestion metric to indicate whether the gated slice needs to be awakened to keep or improve performance. The MBO is suggested as an appropriate congestion metric in [13]. If traffic becomes dense and heavy, more buffer entries may be occupied by the increased packets in some input ports, and the corresponding MBO value increases at the same time. In our design, we utilize the dual congestion thresholds to avoid the frequent switches between difference state transitions: if the MBO is greater than the predetermined upper threshold T_{up}, then current router is considered to be congested and needs to be awakened to mitigate the congestion situation, whereas, current router is considered to be light-loaded when the MBO is less than the predetermined lower threshold T_{low}. If a router keeps in light-load for the T_{idle} cycles, its gated slice should be turned off to save partial leakage power. Besides, we utilize the look-ahead routing [11] as the backup scheme to activate the gated slices early, since some slices which locate in the congested path may fail to be awakened in time before the in-flight packet arrives. In the three-stage pipeline router, wake-up delay of up to 5 cycles can be hidden by sending an early wake-up signal to the partially on downstream router which is two hops away from current router.

In each router, there is a need for a gate controller to control the wake-up and sleep procedures for gated slice. The controller should check the MBO in current router for congestion arbitration and also receive possible wake-up request signals (*Wake_Req*) from upstream routers which are one hop or two hops away. Meanwhile, it should send a gate control signal to switch on/off the power transistor of gated slice and then control the power supply of gated slice. Essentially, the gate controller is a Finite State Machine (FSM), and the power state transition for the FSM is presented in Figure 5. When a gated slice switches its power state, *Handshaking* mechanism should be applied to all adjacent routers for guaranteeing the correctness of packet transmission. On the aspect of wake-up process (*Sleep* $\rightarrow$ *Wake-up* $\rightarrow$ *Active*), if current router is partially on and its gate controller receives a wake-up request signal *Wake_Req* = 1 or detects current router being congested, then the gated slice starts the wake-up process and sends the power state acknowledgement signals *Power_Ack* = 1 to all possible wake-up request routers after accomplishing the wake-up process. T_{wake} cycles are needed to charge the decoupling capacitance in the gated slice, and the wake-up counter W_{counter} is used to count the wake-up delay T_{wake} and decide when to send the acknowledgement signals *Power_Ack* = 1. When all possible wake-up request routers receive the acknowledgement signals *Power_Ack* = 1, they release the corresponding wake-up request signals (*Wake_Req* = 0) and update their list of the available outports for routing computation (RC) stage and switch allocation (SA) stage. On the aspect of sleeping process (*Active* $\rightarrow$ *Sleep*), there are two typical cases: (1) if a gated slice has been activated and keeps in idleness for T_{idle} cycles, it can be turned off to save leakage power; (2) if a fully on router keeps in light-load for T_{idle} cycles, its gated slice also can be turned off after all packets in the gated slice have been sent out. However, the gated slice should send some acknowledgement signals *Power_Ack* = 0 to all adjacent routers at (T_{idle}-3) cycles to early notify them to stop packet transmission before the SA stage and then switches off its power supply at T_{idle} cycles. This mechanism can prevent the probability that the gated slice may receive some in-flight packets after being switched

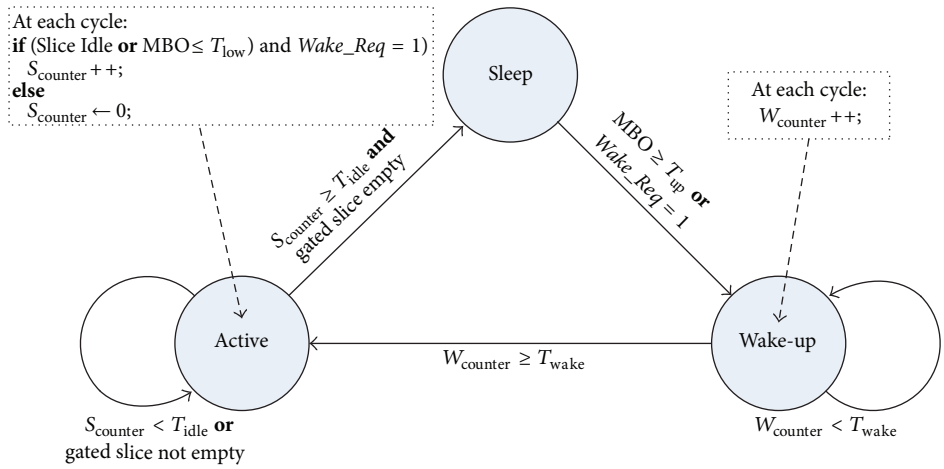


Figure 5: Power state transition for gated slice.

off. The sleep counter $S_{counter}$ is used to count the idle cycles in the fully on router and decides both when to switch off the power supply and when to send acknowledgement signals $Power\_Ack = 0$ to all adjacent routers.

4. Evaluation

In this section, we evaluate the following designs: (1) NoPG-Mesh and NoPG-Torus: baseline mesh and torus network without power-gating; (2) ConPG-Mesh and ConPG-Torus: applying conventional power-gating to each router in baseline mesh and torus network and being optimized with early wake-up [11]; (3) DSPG-Mesh and DSPG-Torus: applying partial power-gating to the sliced mesh and torus network and also being optimized with early wake-up. At the same time, DSPG-Mesh and DSPG-Torus adopt partial clock-gating to save the clock switching power during the sleeping cycles.

4.1. Simulator, Configuration, and Workload. At first, we modify a detailed cycle-accurate network simulator [22] to simulate the proposed design and other comparative designs. Both network configuration and workload pattern are shown in Table 1, and the router parameters are applied to each design. Then, we use ORION 2.0 [23] to estimate network power-consumption and router area and assume all designs implemented with 32 nm technology, 2.0 GHz frequency, 1.0 V operating voltage, SRAM-based input buffer and output latch, MULTREE-based crossbar, and 1.0 mm link between adjacent nodes. Besides, we assume that the idleness detection threshold T_{idle} = 8 cycles, the wake-up latency T_{wake} = 10 cycles, the upper threshold for congestion estimation T_{up} = 8 flits, the lower threshold for light-load estimation T_{low} = 2 flits, and the wake-up overhead is 12 cycles with static energy of itself (according to the power model referred to in the Catnap [13]).

In synthetic traffic simulation, we evaluate the proposed design and other comparisons with four synthetic traffics. To accurately evaluate latency and throughput, the simulator is warmed up for 10000 cycles under the load without taking measurements until steady-state is reached and then simulates for 100000 cycles to obtain steady results. In addition to the synthetic workloads, we also compare the performance of different designs with ten application traces. These traces are collected from the interface module within the GARNET [24] by running corresponding SPLASH-2 benchmarks on the full-system simulator GEMS. The configuration of GEMS simulator is shown in Table 2. The simulator with SPLASH-2 traces is warmed up for sufficiently large cycles and collects the results after running with a maximum of ten million cycles or until trace completion.

Table 1: Network configuration and workload.

Topology	8 × 8 mesh and torus
Router parameter (for each design)	Virtual channel flow control, 3-stage pipeline, 128-bit channel, 4 VCs/channel, and 4 buffer entries/VC
Synthetic traffic	Single flit per packet and Poisson distribution injection
SPLASH trace	Single flit per control packet and 5 flits per data packet

Table 2: GEMS simulator configuration.

Topology	8 × 8 mesh, 1 core, and 1 L2Cache bank per node
Private L1Cache	32 KB instruction & data cache, 4-way, 64 B/line, and LRU
Shared L2Cache	NUCA, 8-way, 64 banks, 512 KB/bank, 64 B/line, and LRU
Cache coherence	MOESI_CMP_Directory protocol
Memory controller	8, connected with nodes 2, 5, 16, 23, 40, 47, 58, and 61

4.2. Comparison across Full Range Synthetic-Load. In order to investigate the behavior of different power-gating schemes

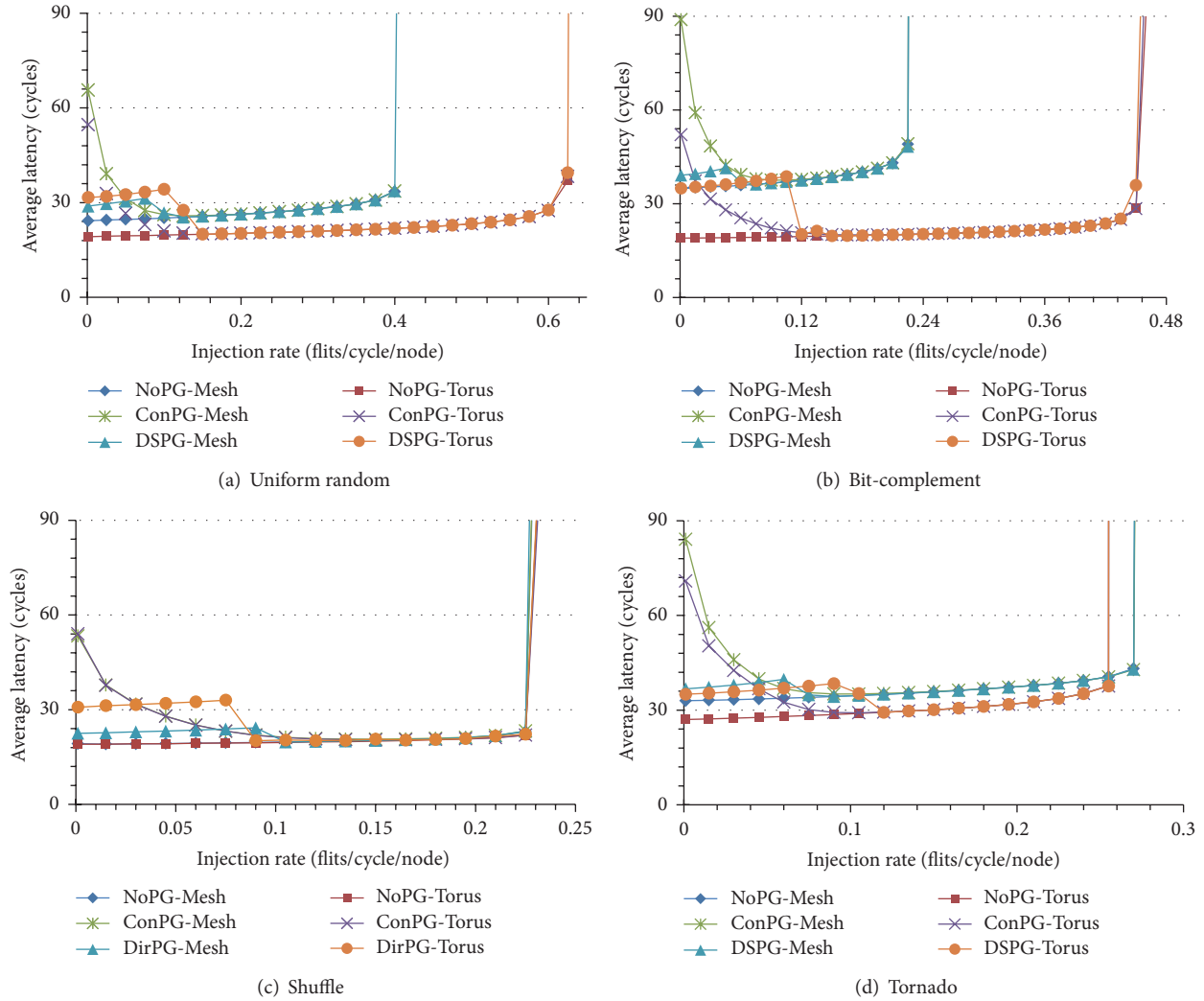


Figure 6: Performance comparison in four synthetic traffic loads.

more entirely, we utilize four synthetic traffics and vary the network load across the full range: from zero-load to saturation-load. Figures 6 and 7 present the performance and power-consumption for the *uniform random*, *bit-complement*, *shuffle,* and *tornado* traffic patterns. On the aspect of performance, typical latency behavior can be observed for NoPG-Mesh and NoPG-Torus, while disappointing results can be found for ConPG-Mesh and ConPG-Torus. At low-load range, packets in both ConPG-Mesh and ConPG-Torus are likely to experience cumulative wake-up delays, since they may meet most routers that are powered-off in their routing path and need to wait for their sleeping downstream routers to be awakened at each hop before being transferred. These wake-up delays are only partially hidden by early wake-up. As the injection rate increases, some routers in a routing path may already be activated by other packets and hold in power-on state, and corresponding wake-up delays can be avoided, so that the average latency in ConPG-Mesh and ConPG-Torus decreases as the load gradually increases. However, DSPG-Mesh and DSPG-Torus have the improved performance at low-load range, because the ever-on subnet can be used to support packet transfer at low-load. They only incur little performance penalty owing to few extra detours in the ever-on subnet. As the workload continually rises, the ever-on subnet begins to be congested, and the latency behavior of our sliced design gradually switches to the typical latency behavior. It is noteworthy that both DSPG-Mesh and DSPG-Torus can indeed reach the maximum throughput of their baselines. This means that all slices can be correctly awakened if needed, which is important and necessary for supporting the high-load phases during the application execution.

On the aspect of power-consumption, NoPG-Mesh and NoPG-Torus produce more dynamic power as the injection rate increases but consume the same leakage power in the entire load range. The typical power behaviors of baselines are linear. However, ConPG-Mesh and ConPG-Torus only save notable leakage power when the workload approaches the zero-load, since most routers can gain great many sleep opportunities. As the load continually rises, more and more routers should be awakened and keep active; thus ConPG-Mesh and ConPG-Torus have decreasing opportunities to

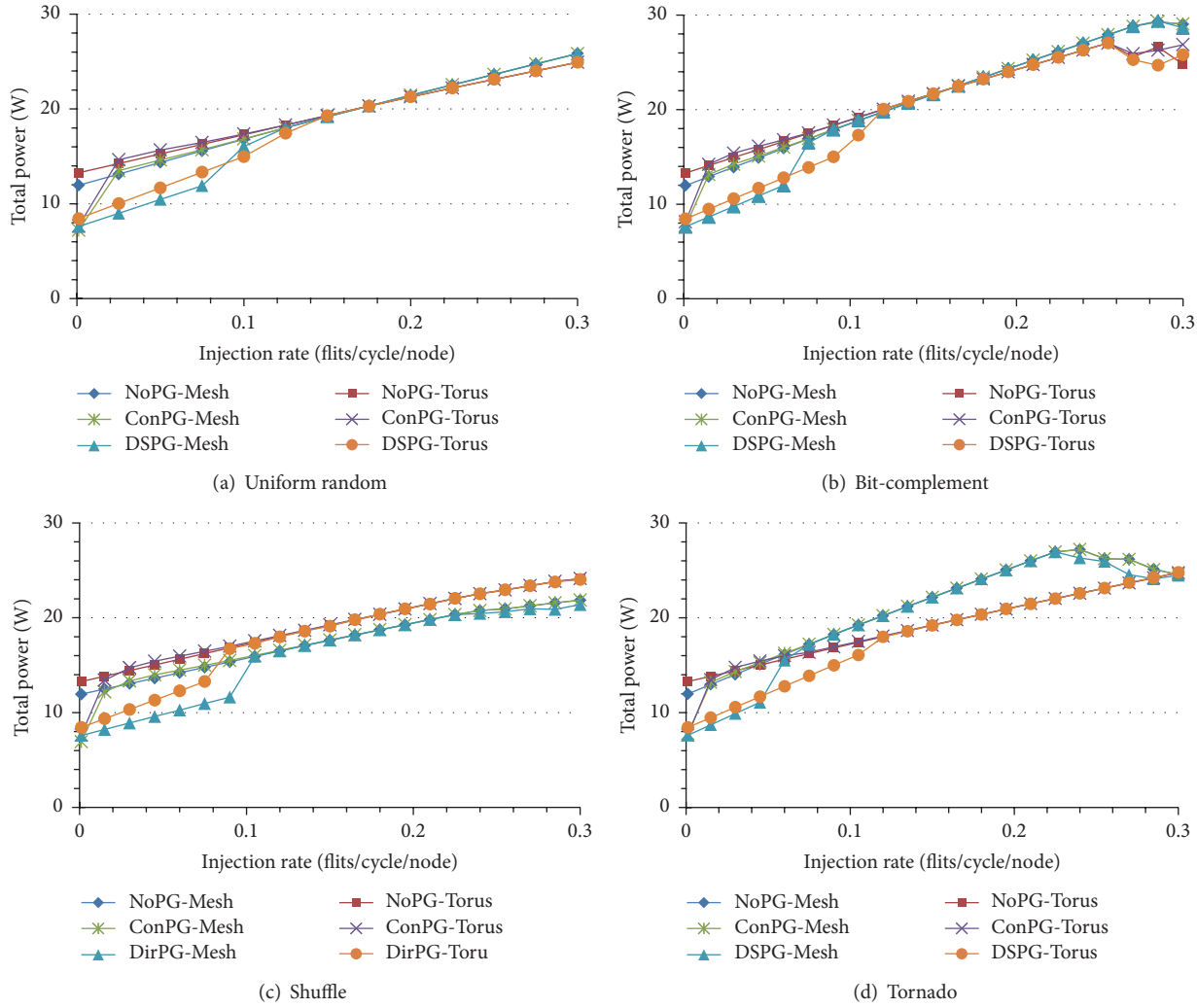
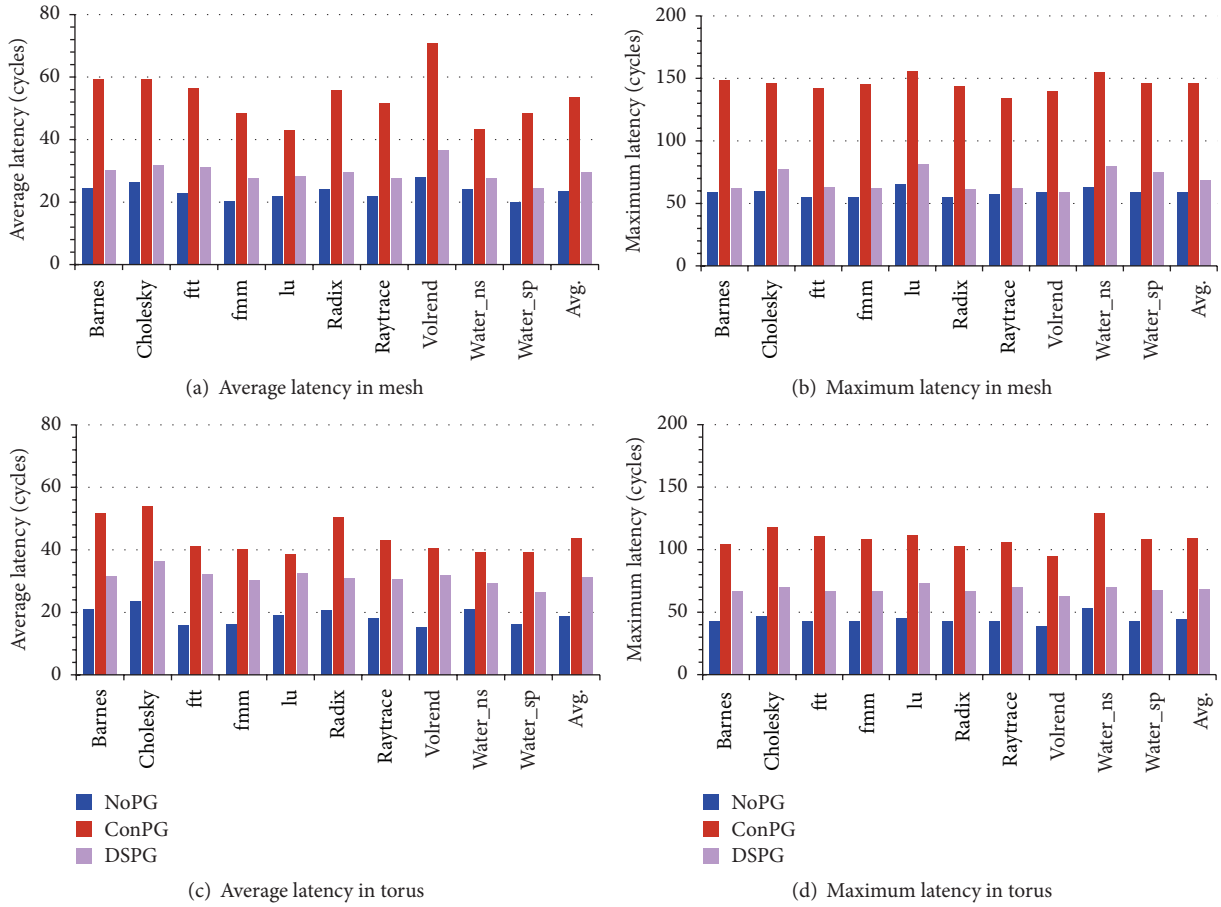


Figure 7: Total power-consumption comparison in four synthetic traffic loads.

obtain power-saving. Moreover, the wake-up overhead may counteract some power-saving, because gated blocks need to charge and discharge the decoupling capacitance frequently. As shown in Figure 7, the power of conventional power-gated design increases very fast and quickly approaches the power curves of its baselines. In contrast, DSPG-Mesh and DSPG-Torus save less leakage power than ConPG-Mesh and ConPG-Torus when near the zero-load, but they can prolong the power-saving range and gain more sleep opportunities for the gated slices during the low to medium load range. In addition, DSPG-Mesh and DSPG-Torus with partial clock-gating can remove some clock's switching power along with the powered-off slices. Similarly to former latency behavior, when more and more gated slices need to be awakened along with the increasing workload, the power curve of our sliced design gradually approaches the typical power curve.

Despite the fact that torus network has better performance than the mesh network in each synthetic traffic, the latency gap between DSPG-Mesh and NoPG-Mesh is smaller and more steady than the gap between DSPG-Torus and NoPG-Torus before the latency behavior switches. The key reason is the impact of average hops which will be explained in Section 4.5. As shown in Figure 6, the maximum latency gaps between DSPG-Mesh and NoPG-Mesh are 6.4, 5.8, 4.6, and 6.0 cycles for the *uniform random*, *bit-complement*, *shuffle,* and *tornado,* respectively, while they are 14.5, 19.3, 13.6, and 9.8 cycles between DSPG-Torus and NoPG-Torus. In the torus network, the latency gap in the *tornado* is less than the gap in other three traffics, because the communication pattern of the *tornado* is more suitable in the sliced torus than other three traffics'.

4.3. Comparison in Application Workloads. In order to examine the practicability of our sliced design, we analyze the packet latency and power-consumption within different power-gating schemes by utilizing SPLASH application traces as the injected traffic. Figure 8 shows the average latency and the maximum latency in both mesh and torus networks for ten SPLASH application traces. Since NoPG-Mesh and NoPG-Torus do not adopt power-gating, they provide a lower bound of average and maximum latency for the mesh and the torus, respectively. The conventional power-gated



(a) Average latency in mesh

(b) Maximum latency in mesh

(c) Average latency in torus

(d) Maximum latency in torus

Figure 8: Latency comparison in SPLASH simulation.

design (ConPG-Mesh/ConPG-Torus), even with early wake-up optimization, still averagely increases 1.29X/1.34X average latency and even 1.48X/1.47X maximum latency compared with NoPG-Mesh/NoPG-Torus. In contrast, DSPG-Mesh and DSPG-Torus completely remove the wake-up latency from the critical path but may need to make some extra detours in the ever-on subnet. Consequently, in comparison with NoPG-Mesh/NoPG-Torus, the proposed design only has 26.0%/66.9% average increment on average latency and 16.2%/54.3% average increment on maximum latency. On average, this is equivalent to 45.0%/28.7% improvement for the average latency when compared with ConPG-Mesh/ConPG-Torus and 53.2%/37.7% improvement for the maximum latency. In average latency comparison, DSPG-Mesh even has better performance than DSPG-Torus and averagely decreases 5.5% of average latency.

Figure 9 presents the breakdown of total NoC power across ten SPLASH application traces and reveals the relative impact on each power component within different power-gating schemes. Several observations can be drawn from both Figures 8 and 9. First, except for *lu* and *water_nsquared* application traces, ConPG-Mesh and ConPG-Torus can gain large saving in leakage power but inevitably incur very large packet latency as mentioned above. Since the injection rate of these application traces is light and sparse, it makes most routers always inactive completely and results in very low leakage power in ConPG-Mesh and ConPG-Torus. However, under *lu* and *water_nsquared* application traces, ConPG-Mesh and ConPG-Torus increase total power-consumption significantly due to the increased injection rate (still low) and still incur very large packet latency. Moreover, they bring remarkable wake-up overhead as seen in *lu* and *water_nsquared* traces within the bar chart. Second, except for *water_nsquared* application trace, there is scarcely any wake-up overhead for each application trace in DSPG-Mesh and DSPG-Torus, since all gated slices do not need to be awakened to avoid network congestion. Meanwhile, they only incur tiny wake-up overhead in *water_nsquared* application trace owing to some short-term congestions. Third, DSPG-Mesh and DSPG-Torus save less leakage power than ConPG-Mesh and ConPG-Torus, but thay can obtain some additional dynamic power-saving due to the partial clock-gating. On the whole, although DSPG-Mesh and DSPG-Torus averagely consume 15.2%/18.9% more total power than ConPG-Mesh/ConPG-Torus, they still save 35.4%/35.5% of total power on average when compared with NoPG-Mesh/NoPG-Torus. At the same time, the sliced design has less impact on network performance as mentioned above. This means that, in comparison with the conventional power-gated design, the proposed NoC with partial power-gating has a better tradeoff when considering both performance and power-consumption.

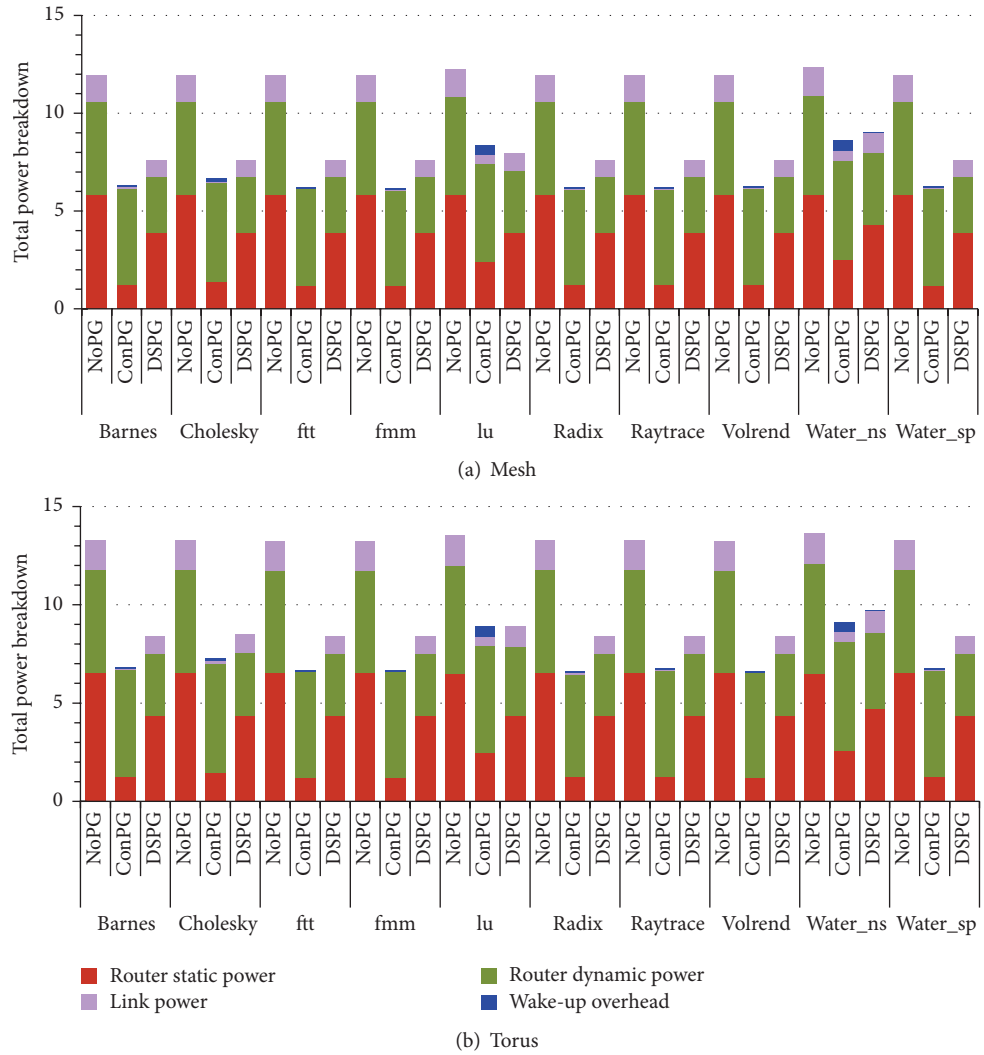
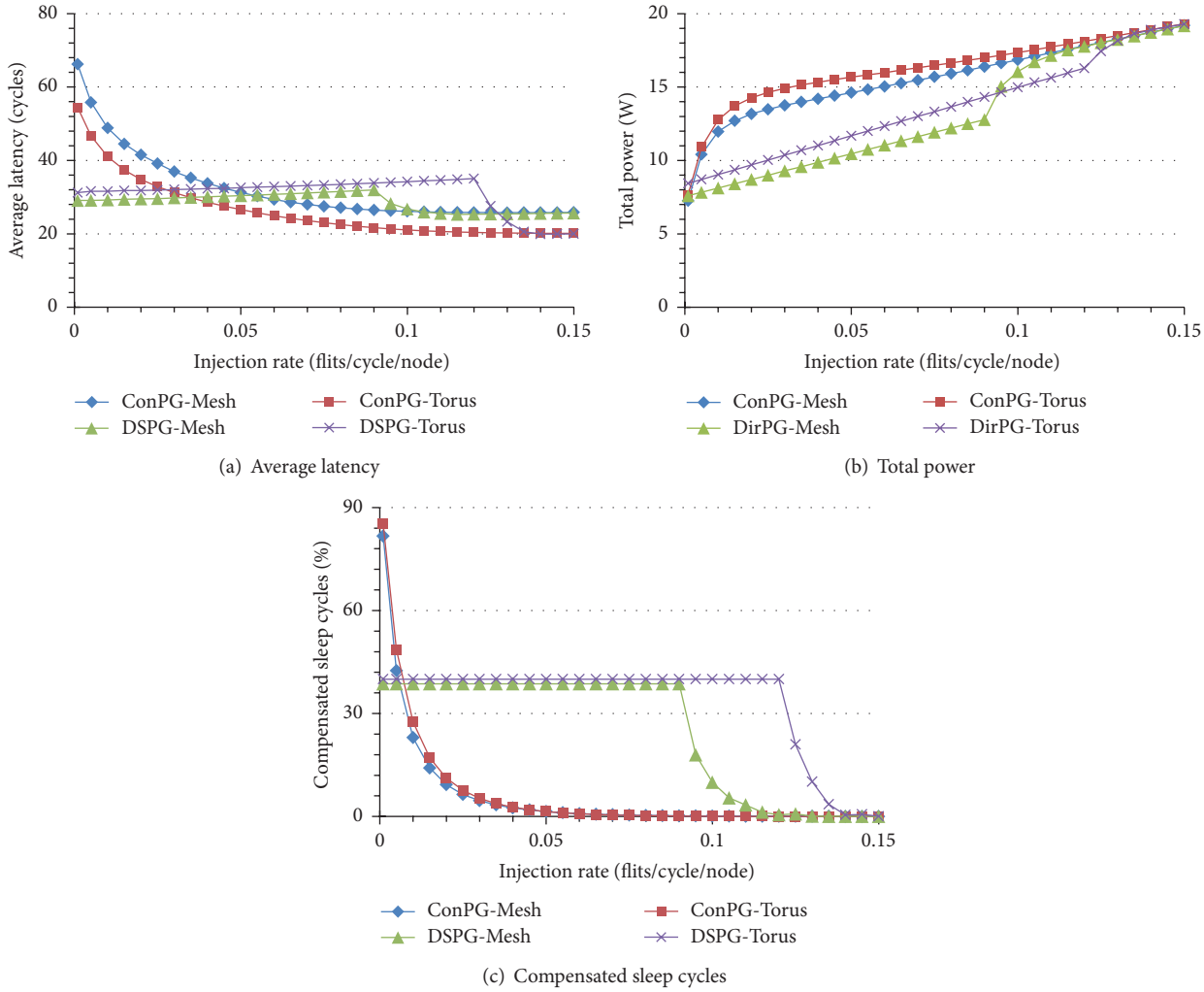


Figure 9: Power-consumption comparison in SPLASH simulation.

4.4. Impact on Compensated Sleep Cycles. Except for the absolute power-consumption, *compensated sleep cycles* (CSC) can be used to quantify the opportunity of power-gating, which is equal to the sum of idle cycles minus the break-even cycles ($T_{\text{breakeven}}$) accounted for every sleep period [13]. We present the CSC as a percentage of total cycles elapsed in the whole execution, which is the fraction of time when the gated routers or slices do not incur any leakage power. Figure 10 reveals the relationship between average latency, power-consumption, and CSC when using the uniform random traffic. Average latency has the similar decline trend as the CSC, because less sleep opportunities mean less waiting delay in the wake-up process, while power-consumption has the opposite trend relative to the CSC, since the CSC value is an indicator for the opportunity of power-saving. As shown in Figure 10(c), when the workload highly approaches zero-load, both ConPG-Mesh and ConPG-Torus have higher CSC value than DSPG-Mesh and DSPG-Torus due to the large sleep opportunities, while the sliced design needs to keep its fully connected subnets always-on and makes the CSC a fixed percentage (about 39.9% for DSPG-Torus and 38.6% for DSPG-Mesh) before other gated slices are awakened. As the load gradually increases, the CSC values in ConPG-Mesh and ConPG-Torus decrease greatly and quickly approach zero. The reason is that ConPG-Mesh and ConPG-Torus must activate their sleeping routers due to the increased packets and have to sacrifice the opportunity of sleep. However, the sliced design can keep high CSC value longer, since its fully connected subnet is always active to sustain an acceptable performance in low-load. Until the ever-on subnet becomes congested, DSPG-Torus and DSPG-Mesh need to awaken some gated slices to guarantee performance, and then their CSC values decrease obviously. Altogether, our sliced design can gain more power-saving opportunities than the conventional power-gated design when the traffic is not near the zero-load.

4.5. Impact on Hop Counts. Since packets in the ever-on subnet may not be transmitted with the minimal path as in normal network, the increased packet hops intuitively have



(a) Average latency

(b) Total power

(c) Compensated sleep cycles

FIGURE 10: The relationship between latency, power-consumption, and compensated sleep cycles.

some negative impact on performance. Under the uniform random traffic, Figure 11 presents the average hop counts within different network scales when all gated slices are powered-on/off in the mesh/torus network. Because the diameter of the unidirectional dimension ring increases as the network scale enlarges, the gap of average hop counts between torus and UniTorus increases, from 1.1 hops to 7.1 hops. However the average hop counts in the UniMesh are only a bit higher than the mesh (about 1.2 hops on average); the reason is that the UniMesh's extra detours are not more than 6 hops. This explains why the performance of the UniMesh is not obviously influenced by the extra detours. By this token, despite the fact that torus network has better performance than the mesh network, the UniMesh subnet is a more suitable selection than the UniTorus subnet above the 6×6 network since its average hop counts are shorter than the UniTorus'.

4.6. Discussion

(1) In our sliced design, if upper congestion threshold decreased, the latency behavior will switch early and the negative impact on performance can be alleviated; at the same time the CSC value will decrease early and the range of power-saving will shrink. Therefore, it is very important to select an appropriate congestion threshold.

(2) DSPG-Mesh has better performance than DSPG-Torus in the large scale network, but its algorithm in the routing computation units is relatively complex. However, DSPG-Torus sacrifices some performance for the sake of simple routing design. DSPG-Torus also can reduce average hops by adopting a complex routing algorithm, the same as the algorithm in DSPG-Mesh.

(3) The proposed direction-slicing scheme is applicable to other regular networks with bidirectional channels, and it also employs a similar methodology to split each router.

(4) The ever-on subnet in Catnap [13] can also utilize the direction-slicing scheme with partial power-gating to further mitigate leakage power.

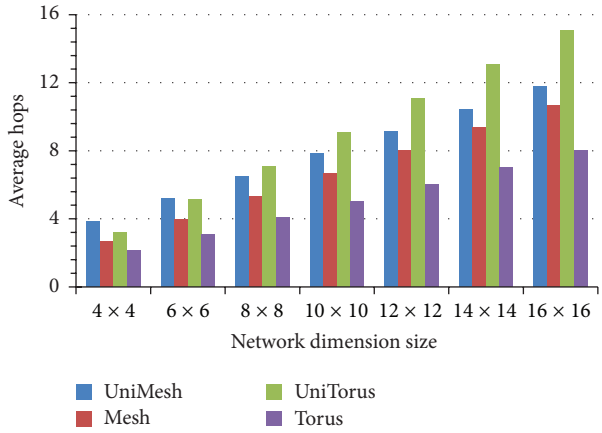


Figure 11: Average hop counts of four different topologies in different network scales.

5. Conclusion

Current and future many-core systems require NoC to be well-designed to guarantee both performance and power-efficiency. However, as technology continually is scaling down, leakage power takes up a larger proportion of total power, and it is increasingly important to reduce the leakage power for the power-efficient NoC design. Power-gating as a representative low-power technique can be applied to NoC to mitigate this increasing leakage power. However the disconnection problem severely limits power-gating to be effectively utilized due to its negative impact on performance. In this paper, we propose a novel partial power-gating approach to obviate the disconnection problem in the power-gated NoC. The approach mainly includes a direction-slicing scheme, an improved routing algorithm, and a deadlock recovery mechanism. First, we utilize direction-slicing scheme to split a router into two slices and adopt different power supply modes for them: keeping one slice always-on to construct an active and fully connected subnet, while utilizing both power-gating and clock-gating to gate the other one to save possible leakage power and clock power. Second, we redesign the corresponding routing algorithm to support packets transmitting on sliced network. And third, we provide a novel deadlock recovery mechanism to solve the deadlock situation which may happen due to the improved architecture and routing algorithm.

In synthetic traffic simulation, results show that the sliced network with partial power-gating is more power-efficient at low-load range and has better performance behavior than the conventional power-gated design. Both DSPG-Mesh and DSPG-Torus can reach their maximal throughput as the baselines, but at low-load range they incur a bit of added latency which is more acceptable than the wake-up delays in ConPG-Mesh and ConPG-Torus. Moreover, since the sliced design can keep high CSC value longer at low-load range, it can prolong the power-saving range and gain more sleep opportunities for gated slices than the conventional power-gated design. In the application simulation, despite the fact that DSPG-Mesh/DSPG-Torus would averagely consume 15.2%/18.9% more total power, DSPG-Mesh/DSPG-Torus can improve network performance by 45.0%/28.7% on average when compared with ConPG-Mesh/ConPG-Torus. In conclusion, the proposed design with partial power-gating can attain the objectives of high-performance and low-power.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper was supported in part by the National High-Tech 863 Project of China under Grant no. 2009AAOIZ102, in part by the National Natural Science Foundation of China under Grant no. 60873016 and no. 61170083, and in part by the Doctor Program Foundation of Education Ministry of China no. 20114307110001.

References

[1] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[2] B. K. Daya, C.-H. O. Chen, S. Subramanian et al., "SCORPIO: a 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering," in *Proceedings of the ACM/IEEE 41st International Symposium on Computer Architecture (ISCA '14)*, pp. 25–36, ACM, June 2014.

[3] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *Proceedings of the 19th IEEE International Symposium on High Performance Computer Architecture (HPCA '13)*, pp. 508–519, IEEE, February 2013.

[4] P. Bogdan, M. Kas, R. Marculescu, and O. Mutlu, "QuaLe: a quantum-leap inspired model for non-stationary analysis of NoC traffic in chip multi-processors," in *Proceedings of the 4th ACM/IEEE International Symposium on Networks on Chip (NOCS '10)*, pp. 241–248, IEEE, Grenoble, France, May 2010.

[5] P. Bogdan and R. Marculescu, "Non-stationary traffic analysis and its implications on multicore platform design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 508–519, 2011.

[6] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *Proceedings of the 40th Design Automation Conference*, pp. 622–627, IEEE, June 2003.

[7] R. Mullins, "Minimising dynamic power consumption in on-chip networks," in *Proceedings of the International Symposium on System-on-Chip*, pp. 1–4, IEEE, November 2006.

[8] A. Pullini, F. Angiolini, P. Meloni et al., "NoC design and implementation in 65 nm technology," in *Proceedings of the First International Symposium on Networks-on-Chip (NOCS '07)*, pp. 273–282, May 2007.

[9] C. Feng, Z. Lu, A. Jantsch, and M. Zhang, "A 1-cycle 1.25 GHz bufferless router for 3d network-on-chip," *IEICE Transactions on Information and Systems*, vol. 95, no. 5, pp. 1519–1522, 2012.

[10] M. R. Casu, M. K. Yadav, and M. Zamboni, "Power-gating technique for network-on-chip buffers," *Electronics Letters*, vol. 49, no. 23, pp. 1438–1440, 2013.

[11] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Performance, area, and power evaluations of ultrafine-grained run-time power-gating routers for CMPs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 520–533, 2011.

[12] L. Chen and T. M. Pinkston, "NoRD: node-router decoupling for effective power-gating of on-chip routers," in *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '12)*, pp. 270–281, IEEE Computer Society, Vancouver, Canada, December 2012.

[13] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: energy proportional multiple network-on-chip," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13)*, pp. 320–331, ACM, Tel-Aviv, Israel, June 2013.

[14] L. Chen, L. Zhao, R. Wang, and T. M. Pinkston, "MP3: Minimizing performance penalty for power-gating of Clos network-on-chip," in *Proceedings of the 20th IEEE International Symposium on High Performance Computer Architecture*, pp. 296–307, IEEE, February 2014.

[15] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*, The Morgan Kaufmann Series in Computer Architecture and Design, Elsevier, 2004.

[16] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of the 2004 International Symposium on Lower Power Electronics and Design (ISLPED '04)*, pp. 32–37, ACM, August 2004.

[17] H. Jiang, M. Marek-Sadowska, and S. R. Nassif, "Benefits and costs of power-gating technique," in *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '05)*, pp. 559–566, IEEE, October 2005.

[18] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 3, pp. 415–420, 2000.

[19] S. Ma, N. E. Jerger, and Z. Wang, "Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip," in *Proceedings of the 18th IEEE International Symposium on High Performance Computer Architecture*, pp. 1–12, IEEE, February 2012.

[20] M. Shin and J. Kim, "Leveraging torus topology with deadlock recovery for cost-efficient on-chip network," in *Proceedings of the 29th IEEE International Conference on Computer Design (ICCD '11)*, pp. 25–30, IEEE, Amherst, Mass, USA, November 2011.

[21] A. Lankes, T. Wild, A. Herkersdorf, S. Sonntag, and H. Reinig, "Comparison of deadlock recovery and avoidance mechanisms to approach message dependent deadlocks in on-chip networks," in *Proceedings of the 4th ACM/IEEE International Symposium on Networks on Chip (NOCS '10)*, pp. 17–24, IEEE, Grenoble, France, May 2010.

[22] L. Jain, B. Al-Hashimi, M. Gaur, V. Laxmi, and A. Narayanan, "NIRGAM: a simulator for NoC interconnect routing and application modeling," in *Proceedings of the Design, Automation and Test in Europe Conference*, pp. 16–20, Nice, France, April 2007.

[23] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE '09)*, pp. 423–428, IEEE, Nice, France, April 2009.

[24] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: a detailed on-chip network model inside a full-system simulator," in *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS '09)*, pp. 33–42, IEEE, April 2009.