Bufferless Network-on-Chips With Bridged Multiple Subnetworks for Deflection Reduction and Energy Savings

Xiyue Xiang, Purushottam Sigdel, and Nian-Feng Tzeng, *Fellow, IEEE*

Abstract—A bufferless network-on-chip (NoC) can deliver high energy efficiency, but such a NoC is subject to growing deflection when its traffic load rises. This article proposes Deflection Containment (DeC) for the bufferless NoC to address its notorious shortcomings of excessive deflection for performance improvement and energy savings. With multiple subnetworks bridged by an added link between two corresponding routers, DeC lets a contending flit in one subnetwork be forwarded to another subnetwork instead of deflected. Microarchitecture of DeC routers is rectified to shorten the critical path and lift network bandwidth. Its Cadence RTL implementations with a $15-nm$ process are conducted respectively for mesh-based NoCs and torus-based NoCs. Additionally, different sized DeC-NoCs are evaluated extensively and compared with previous bufferless designs (BLESS and MinBD), uncovering that DeC with two bridged subnetworks (dubbed DeC2) for 8X8 mesh-based NoCs can lower deflection drastically by some 90 percent and energy consumption by upto 51 percent under real benchmark traffic loads, in comparison to BLESS. Under various synthetic traffic models and workloads, 16X16 torus-based DeC2-NoC sustains up to 2.33X loads when compared with its mesh-based counterpart, exhibiting the same clock rate and taking only negligible more power and area according to our full layout results.

Index Terms—Bufferless network-on-chips (NoCs), routing deflection, flits, interconnects, NoC routers, path diversity

1 Introduction

As the chip multicore processor (CMP) continues to upscale its core count, its involved on-chip interconnect (NoC) may soon dominate overall chip power consumption. The bufferless NoC has been pursued as a compelling design alternative for CMPs to lower network power and energy consumption as a result of removing buffers [4], [5], [6], when compared with its traditional buffered counterpart [7], [12].

Without buffers, a bufferless router has only pipeline registers, each of which is able to hold one flit. The operation of bufferless routers is thus simple: all flits must move forward through the pipeline and be sent out to some output ports at the end of each cycle. Whenever multiple flits require the same output port, only one flit (which usually has the highest priority) is granted, with the remaining contending flits either deflected to the unclaimed ports [4], [5] or simply dropped [8]. A dropped flit is retransmitted later to ensure correctness. In this work, we focus on the bufferless NoC using deflection routing to resolve resource contention.

Existing bufferless NoCs often suffer from rapidly growing deflection rates when the load increases. Fig. 1 shows the packet network latency and the deflection rate of BLESS [5] — a state-of-the-art bufferless NoC, under uniform random traffic in a 16×16 mesh-based NoC and a torus-based NoC. When the network load increases, flit contention intensifies, hiking the deflection rate by nearly $48\times$ upon close saturation. High deflection also increases the average latency in delivering packets, since many flits then take unproductive extra hops before reaching their destinations. A deflected flit involves multiple extra cycles to reach its destination. Moreover, those deflected flits consume network bandwidth, possibly interfering with more flits to further amplify the deflection rate and easily causing network saturation. Additionally, high deflection calls for large reorder storage at the destination end (i.e., MSHR) [4], since flits are transmitted independently across the network. As a result, energy consumption rises and performance degrades. Therefore, deflection becomes the key stumbling block toward energy efficiency and good performance for bufferless NoCs.

In this paper, we propose Deflection Containment for bufferless NoCs to reach significant energy savings with commendable performance improvement through curbing deflection and lowering router complexity, achieved by adding a link to each router for bridging subnetworks (whose aggregate link width equals a given value, say, 256b). As depicted in Fig. 2, M (e.g., $M = 2$) subnetworks are bridged together at each node to form a bypass ring, which lets a contending flit be forwarded to another subnetwork without deflection so as to lower the latency penalty and to reduce bandwidth and energy waste. DeC with $M = 2$ subnetworks is denoted by DeC2.

We implement BLESS, MinBD [6], and DeC routers in Verilog to evaluate power, timing, and area via full synthesis

• *X. Xiang is with the Xilinx Inc., 2100 All-Programmable Dr., San Jose, CA 94086. E-mail: xiyuex@xilinx.com.*
• *P. Sigdel and N.-F. Tzeng are with the School of Computing and Informatics, the University of Louisiana at Lafayette, 301 E. Lewis Street, Lafayette, LA 70503. E-mail: {pxs9444, tzeng}@louisiana.edu.*

Manuscript received 17 Apr. 2019; revised 18 Nov. 2019; accepted 8 Dec. 2019. Date of publication 18 Dec. 2019; date of current version 10 Mar. 2020. (Corresponding author: P. Sigdel.)
Recommended for acceptance by C. SILVANO.
Digital Object Identifier no. 10.1109/TC.2019.2959307

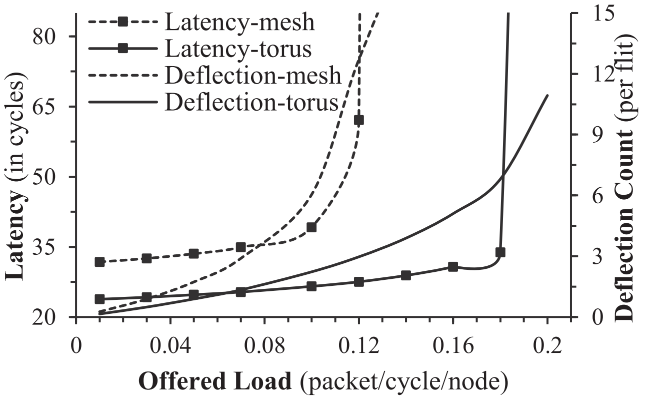

Fig. 1. A bufferless NoC suffers from ~48× (or ~45×) more deflection as the network load approaches saturation under mesh (or torus) sized 16×16, substantially prolonging latency and lowering its capacity.

using Cadence Encounter RTL Compiler with 15 *nm* FinFET-based Open Cell Library [18] (under $V_{dd} = 0.8\,\text{V}$ and $T = 25^{\circ}\text{C}$). Our synthesis implementation for each design includes logics for route computation, partial permutation network, port allocation, local ejection and injection, crossbar, side buffers, and pipeline registers. In addition, we also conduct RTL implementation of DeC router microarchitecture for the torus topology, which improves performance over its mesh counterpart with wrap-around connections between pairs of opposite edge routers of a mesh [13], [58], [60].

For comparison, we evaluate NoCs with DeC support using a modified cycle-accurate simulator [17] under various synthetic traffic loads and under instruction traces obtained from the representative phases of the PARSEC benchmark suite [61] extensively. Evaluated performance results of 16×16 mesh-based DeC2-NoCs under synthetic traffic loads are shown to markedly outperform those of compatible NoCs with BLESS and MinBD support, as a result of substantial deflection containment. Specifically, mesh-based DeC2-NoCs under multi-threaded PARSEC benchmark traffic loads enjoy drastic deflection reduction and substantial energy savings, in comparison to compatible BLESS and MinBD. Full layout results of 4×4, 8×8, and 16×16 DeC2-NoCs with the torus topology are conducted under the 15 *nm* process via Cadence Innovus for energy and speed comparison against their mesh-based counterparts. They reveal that a 16×16 torus-based DeC2-NoC sustains up to $2.33\times$ loads across various synthetic traffic patterns when compared with a mesh-based counterpart, taking only 2.8 percent more power and 2.6 percent more area. In contrast to conventional intuition, a torus-based DeC2-NoC and a compatible mesh-based DeC2-NoC exhibit the same clock rate, because the on-chip links of the former never slow down the clock according to our full layout results.

We have made the following contributions.

1. We identify that prior state-of-art bufferless routers easily lead to considerable deflection, and thus stark performance degradation, as the network load rises under both mesh and torus topologies.
2. We propose DeC to reduce power consumption and improve bufferless NoC performance by curbing deflection to mitigate the contention penalty and lower bandwidth waste using bridged multiple subnetworks.
3. A parallel port allocator is proposed to shorten the critical path of a bufferless router by 23 percent (down to 0.17 *ns* from 0.22 *ns*), according to our RTL synthesis using Cadence Encounter RTL Compiler with the open source NanGate 15 *nm* process.
4. We evaluate 4×4, 8×8, and 16×16 DeC-based NoCs extensively using our gathered instruction execution traces and random traffic loads for comparing with previous bufferless NoCs.
5. We derive full layouts of 4×4, 8×8, and 16×16 DeC2-NoCs with the torus topology under a 15 *nm* process for energy and speed comparison against their mesh counterparts, discovering a torus-based NoC to retain the same clock rate despite its larger mean link length (e.g., $20.2\,\mu m$ versus $15.0\,\mu m$ for the size of 16×16).

2 Related Work

2.1 Bufferless Network-on-Chips (NoCs)

The bufferless NoC (BLESS) was first proposed in 2009 [5] with the aim of eliminating the expensive buffers in conventional buffered NoCs. The bufferless NoC is advantageous over the buffered NoC for several reasons. First of all, due to the absence of buffers, it reduces the router area and power consumption significantly. Second, a bufferless NoC has much simpler control. Both injection and ejection are determined locally. Injection is granted merely based on the availability of the input link. The flit reaching its destination is removed from the network upon winning arbitration. Third, the bufferless NoC involves no flow control so that its routers do not need to maintain and exchange credits with their neighbors. At every cycle, flits do not wait in the upstream buffers when contention occurs. They are simply deflected to a neighboring node to resolve contention. Lastly, a bufferless NoC can deliver similar performance as its buffered counterpart, albeit enjoying extraordinary simplicity. Prior work has attempted at enhancing the bufferless NoC in different domains, as outlined below.

- *Router Complexity Reduction*

CHIPPER [9] consolidates arbitration and switch allocation into a single stage with a partial permutation network, replacing the expensive crossbar employed by BLESS [5]. Based on CHIPPER, MinBD [6] added a centralized buffer in each bufferless router to temporarily hold contended flits to reduce the contention penalty. Meanwhile, deflection routing is adopted in a hierarchical ring network [46], [47]. All of them suffer from degraded performance due to limited path diversity, resulting from few routing combinations in the partial permutation network [5], [6] or a limited number of ports [46], [47]. Hence, they are unappealing, especially for high-throughput applications, such as Convolutional Neural Networks and GPGPU.

- *Congestion Control*

Buffered NoCs often use credits to provide in-network backpressure between neighboring routers. Flits can be stored in buffers when the downstream routers are congested, providing multiple cycles of slack to steer flits away from the congested area. On the contrary, a bufferless NoC has no in-network backpressure in the absence of credits. However, congestion in such a NoC can quickly propagate back to the source node, triggering end-to-end backpressure (despite its absence of in-network backpressure). The cache/memory controller will then stop issuing new requests to alleviate congestion in the bufferless NoC.

Source throttling in bufferless NoCs is explored thoroughly when the metrics of interest exceeds a threshold value [19], [20], [21], [48], [49], [50], [51], [52], by throttling new injection. CFC [19] throttles the injection based on buffer availability at the destination. Later, Sun [20] extends CFC by dynamically adjusting the credit amount assigned to each core for further performance improvement. Daya [49] and Nychis [50] perform throttling based on the starvation rate (i.e., the inability of injecting new packets). Chang [48] and Nychis [50] identify that applications respond differently to source throttling, with the former throttling the application with higher network intensity based on misses per thousand instructions and the latter throttling the application with fewer instructions per flit.

- *Other Prior Work*

Instead of deflection, flits can be dropped upon losing arbitration to resolve contention. Retransmitting dropped flits is triggered to ensure correctness, which often incurs a long latency and a higher buffer requirement at the source node as flits need to be retained until successful delivery. SCARAB [8] utilizes MSHRs at intermediate nodes to buffer in-flight flits opportunistically when the ejection port is idle, to avoid an increase in the processor side buffering requirement. Also, a dedicated circuit-switched network is utilized to trigger retransmission from the source. Carpool [53] enables multicast in a bufferless NoC through adaptively forking multiple cast flit replica and opportunistically merging the hotspot flits to reduce network contention.

2.2 Multiple On-Chip Networks

Multiple on-chip networks have been adopted in silicon prototypes mainly to isolate different message classes for deadlock-freeness and quality of service, as evidenced in OpenPiton [54], Xeon Phi [55], SCORPIO [41], Tilera [24], TRIPS [26], and RAW [25]. Meanwhile, research effort on optimizing different traffic classes in multiple on-chip networks has been made to achieve better performance and lower power consumption [23], [28], [32], [33], [34], [35], [36], [37], [38].

- *Multiple Buffered NoCs*

Earlier studies [33], [34], [38] categorize packets into critical and non-critical ones, letting them transferred in two separated networks in order to optimize power consumption. They employ dynamic voltage frequency scaling (DVFS) to slow down the non-critical network with lower voltage and frequency to save energy. Usually, critical packets are transferred through the regular network with low latencies, whereas non-critical packets are transferred in the network with low power. They differ in terms of identifying the criticality of a packet. Specifically, Deja Vu switching [33] considers control packet as critical and data packets as non-critical, with the non-critical network being circuit-switched to compensate the penalty of DVFS. Flores [34] considers short packets to be critical and long packets as non-critical. NoC-NoC [38] treats data which are needed right away as critical and other data as non-critical.

Flit-reservation flow control [32] uses a separate network to reserve buffers and channels prior to the arrival of a data packet. Mishra [28] uses two separate networks to carry two different types of applications—a high bandwidth, low frequency network for bandwidth-sensitive applications and another low latency, high frequency network for latency-sensitive applications. Natalie [37] explores the underutilized resources existing on the silicon interposer, which forms a separate network for carrying memory traffic.

A multiple-network design is also amiable in power gating for energy savings, since individual subnets can be power-gated without compromising full connectivity [23]. In addition, multiple networks provide opportunities for traffic partitioning [35] and load balancing [36], and fault-tolerance [30].

- *Buffered NoCs With Auxiliary Bufferless Network*

Ordered NoC [39], Runahead [40], and SCORPIO [41] aid a light-weighted bufferless NoC on top of a regular buffered NoC. Like SCORPIO [41], Ordered NoC [39] utilizes a separate bufferless network with fix latency, to maintain global ordering on top of an unordered buffered network[1]. In addition, Runahead [40] uses a bufferless NoC to speed up the delivery of latency-critical packets, with the bufferless NoC being lossy such that flits will be dropped in the presence of contention (i.e., without retransmission).

2.3 Summary

Most of previous work investigated into multiple buffered NoCs, with multiple bufferless NoCs yet to be explored. Recently, a few attempts at pursuing multiple on-chip networks in the bufferless domain have been made [39], [40], [41]. Unlike our DeC, the bufferless NoCs which have been pursued earlier are considered to be auxiliary ones and cannot be used standalone. In contrast, our DeC employs multiple independent subnetworks to resolve deflection, which is a notorious issue in the bufferless NoC, as discussed in detail next.

3 DeC Design

The major issue of a bufferless NoC is its potentially excessive deflection when the load rises [22]. To reduce deflection, our DeC partitions a network into multiple subnetworks (with aggregated data path width of multiple subnetworks equal to that of the network without partition) which are bridged with bypass channels to contain deflection as a result of increased path diversity.

3.1 Design Approach

DeC aims to reduce the deflection rate through exploring increased network path diversity. Fig. 2 illustrates the overall DeC structure, which consists of M identical subnetworks (e.g., $M = 2$). The data path of each subnetwork is narrower,

1. Although directory-based coherence can work atop an unordered network, many Snoopy coherence relies on an ordered interconnect to ensure correctness, such as ARM AMBA [42], AXI Coherence Extensions [43], AMD Hypertransport [44], and Intel Quickpath Interconnect [45].

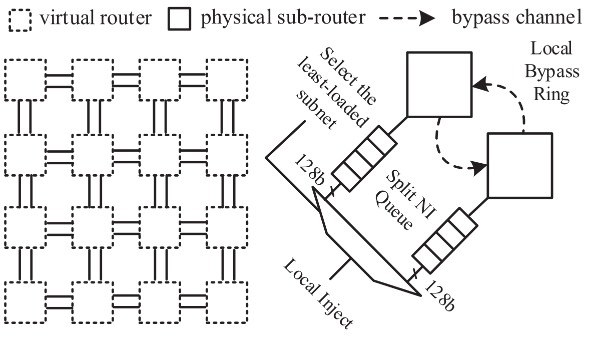

Fig. 2. DeC involving multiple sub-networks (left), which are bridged with a bypass ring at each node (right). Each DeC router comprises multiple physical sub-routers (e.g., $M = 2$, as shown).

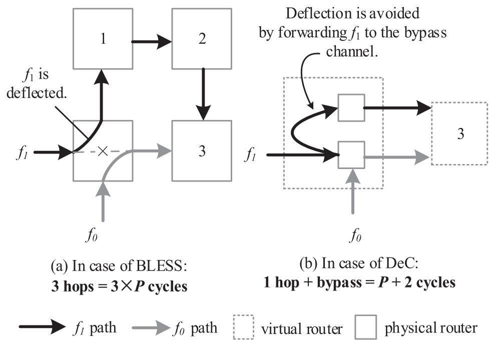
1
2
3
f_1 is deflected.
f_1
f_0
Deflection is avoided by forwarding f_1 to the bypass channel.
3
f_1
f_0
(a) In case of BLESS: 3 hops = 3×P cycles
(b) In case of DeC: 1 hop + bypass = P + 2 cycles
f_1 path
f_0 path
virtual router
physical router

Fig. 3. DeC can effectively avoid deflection by forwarding a contending flit (e.g., f1) to another subnetwork via the bypass channel.

equal to $1/M$ of that of the network with one subnetwork. Subnetworks are bridged together at every node to constitute a bypass ring with uni-directional channels (referred to as the "bypass channels"). A contending flit in one subnetwork, which would otherwise be deflected, can then be forwarded via the bypass ring to another subnetwork at the same node. It allows the contending flit to explore the path diversity and compete for the desired port in another subnetwork with only one extra cycle latency (instead of multiple extra cycles upon deflection). The other flits follow regular deflection routing.

The network interface (NI) is very crucial as it determines the serialization latency for delivering a complete packet. Storing flits into a single injection queue may lead to well-known head-of-line blocking in the presence of multiple subnetworks. Also, subnetwork selection directly affects the network load balance. In DeC, as shown in Fig. 2, a node is connected to M subnetworks, each of which has a dedicated NI injection queue with its size equal to $1/M$ of that of the single network counterpart. Since a flit size in DeC is $1/M$ of that of its single network counterpart, the maximum number of entries in a split queue equals that in the NI injection queue of its counterpart. Multiple queues effectively overcome occurrence of head-of-line blocking without adding significant hardware complexity. Moreover, DeC adaptively selects the subnetwork with a low load to accommodate new flits for improving the load balance.

3.2 Advantages

- *More Paths, Less Deflection*

The deflection rate of a bufferless NoC largely depends on the network load and path diversity. When a network has more independent channels, the probability of contention reduces. Although a high-radix topology such as flattened butterfly [10], dragonfly [11], or MECS [27], exhibits lofty path diversity, all of them often come with steep hardware complexity, because such a topology calls for a large "full crossbar" at each constituent router which scales quadratically with respect to the number of ports. On the other hand, bridged multiple subnetworks can elevate path diversity without resorting to a large crossbar in each node, curbing hardware complexity.

- *Contention Penalty Mitigation*

When contention occurs in bufferless NoCs, a deflected flit may move further away from its destination, causing performance degradation and power waste. Fig. 3 shows a typical case when deflection occurs in a mesh-based bufferless NoC. Assume both f_0 and f_1 destine to Router 3, and f_0 has a higher priority. As contention occurs at Router 0, it takes 3 hops ($3 \times P$ cycles) to delivery f_1 if it is deflected, where P denotes the number of pipeline stages. In contrast, DeC sends the conflicting flit f_1 to another subnetwork through the bypass ring so that it can contend for the desired port again with only a single cycle delay (via traversing through the bypass link). The overall latency for delivering f_1 in this way is only $P + 2$ cycles, with one cycle for traversing through the bypass link and one cycle for arbitrating the output port in the other subnetwork. Under a 2-stage pipelined router with one cycle link latency (i.e., $P = 3$), DeC then mitigates the contention penalty by 4 cycles.

- *Improve Link Utilization and Load Balance*

Network bandwidth is a premium on-chip resource. An increase in the working data set size requires moving more data from off-chip storage to higher level of private and shared cache through NoCs. Multi-threading becomes very common to explore thread-level parallelism provided by underlying hardware in the shared memory semantics, at the cost of increased traffic in the on-chip network due to synchronization and coherence. Future super scalar processors with high memory level parallelism will likely to put even more pressure to the on-chip communication channel.

Many recent chips utilize wide channels to lift up bandwidth and improve on-chip network performance, given that 5 to 13 levels of metal in the current technology provide abundant wiring resources [56]. NoC with a wide channel is very efficient for transferring long packet, such as response packets of a read request or write-back with data payload, since injecting a packet then involves fewer flits. However, it often exhibits poor link utilization when transferring short control packets, as short packets may not utilize the entire link.

By partitioning a network into multiple independent physical subnetworks, DeC is able to use link more efficiently, leading to a better link utilization and increased throughput. Also, injecting new flits adaptively according to the network load, DeC always picks the least loaded subnetwork to accommodate the new flit, preventing unbalanced load distribution among subnetworks. Last but not least, the existence of a bypass channel at each node establishes a tunnel to divert flits from a heavy-loaded subnetwork to a light-loaded subnetwork, further improving load balance and avoiding congestion.

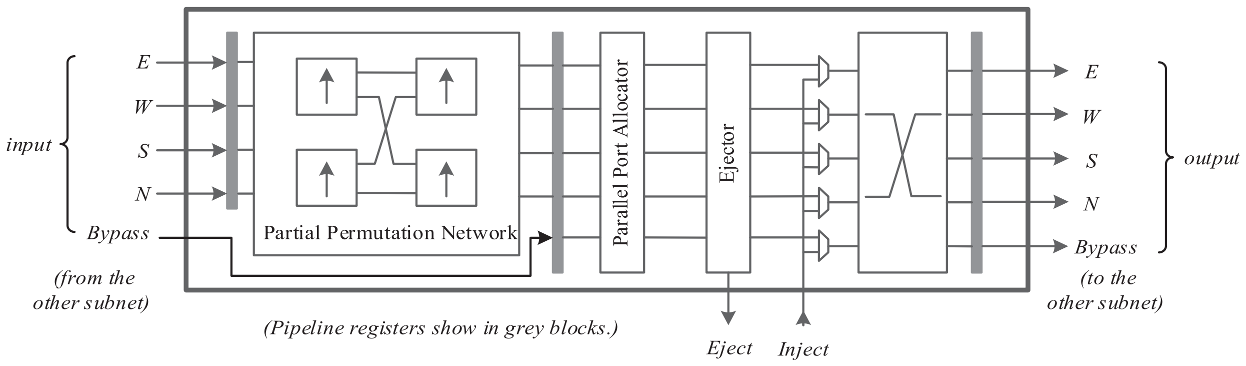

Fig. 4. DeC router microarchitecture.

4 Router Microarchitecture

Several bufferless NoC designs have been proposed, each of which addresses different issues. For example, BLESS [5] performs best-effort deflection routing. CHIPPER [4] reduces the router complexity, and MinBD [6] lowers the deflection rate based on CHIPPER. We adopt the previous design — BLESS, which provides the best performance among all known designs, as the baseline. Compared with BLESS, DeC achieves better performance and much lower power consumption through containing deflection and shortening the critical path.

4.1 Baseline Bufferless NoC — BLESS

In BLESS, flits are transmitted across the network independently. Each flit needs to carry a header that contains routing information and is sent through dedicated links. At the first stage, all incoming flits pass through a sorting network to determine their ranking. At the second stage, port allocation is performed sequentially for flits in a descending order of their ranking. When two flits contend for the same port, BLESS simply deflects the one with lower priority. Finally, a crossbar is used to send out flits according to port allocation results.

BLESS enforces a strict priority rule (i.e., Oldest-First) to make routing decisions, requiring all flits in a router to be fully sorted and allocating ports sequentially in the priority order to resolve contention. Two pitfalls implicate the design. First, full sort incurs high hardware complexity, as it often requires using a 3-stage permutation network. Second, sequential port allocation can lead to a long critical path in a router [4]. Especially, when the load is low, it is prohibitively expensive for strict enforcement of the priority rule. Inspired by [4], DeC relaxes the constraint by ensuring only the highest priority flit to obtain its requested port first, and it then performs parallel port allocation to resolve the sequential dependence during port allocation, as will be stated next.

4.2 DeC Overall Operation

The DeC router requires only relatively modest changes to support deflection containment and resolve the aforementioned issues. Fig. 4 shows the proposed DeC router microarchitecture, which has a two-stage pipeline. In the first stage, it uses conventional X-Y routing logics for the mesh network [13] to calculate the desired ports for all incoming flits, which are then sent to a sorting network to determine their priorities. The DeC router for torus-based NoCs follows lowest hop count considering the wrap-around connections between pairs of opposite edge routers. This is the only difference in comparison to a mesh-based NoC router. At the end of the first pipeline stage, the flit on the top channel possesses the highest priority. At the second stage, port allocation is performed in parallel. One of the contending flits is sent to *Bypass* port, instead of being deflected, such that it can contend for the desired port again after one cycle (via traversing the bypass ring) in another subnetwork. Then, one local-destined flit is selected and ejected from the network. The injection is granted as long as there is an available channel. When all channels in a router are occupied, injection is throttled. A throttled flit simply waits in its NI queue, attempting its injection again in the next cycle. Flits are finally directed to their allocated ports over a crossbar.

4.3 Parallel Port Allocator

BLESS allocates ports to flits sequentially (i.e., one by one in a strict priority order), causing a long critical path, as illustrated in Fig. 5a. On the contrary, port allocation in DeC takes place in parallel, as depicted in Fig. 5b. According to our RTL synthesis using Cadence Encounter RTL Compiler with 15-*nm* FinFET-based Open Cell Library [18], DeC reduces the critical path of port allocator from 148 *ps* to 26 *ps*. Port allocations in each channel is done in two steps, as illustrated in Fig. 5c and described below.

STEP 1: Grant All Non-Conflicting Requests. Based on the desired port request of each flit (*req*), DeC grants the *preferred port* if it does not incur contention with other flits. When $flit_i$ contends for the same port with another flit, port allocation is performed in parallel in STEP 2.

STEP 2: Contending Flits Claim Ports Based on the Availability. Based on the remaining port requests (*req′*) and *available ports*, DeC uses a simple *Lookup Table* to determine the allocated port. Specifically, *req′* tells each port allocator how many flits in lower channels failed to obtain the requested ports in STEP 1. Then, DeC performs port allocation in parallel in a predefined order as {*Bypass*, *North*, *South*, *East*, *West*}[2] from the *available ports*, where the first available port is given to the flit at the lower channel index. Finally, the allocated port is selected, depending on whether it causes any contention in STEP 1.

2. No local destined contention is resolved in the parallel port allocator, and it will be handled by the ejector, as described in Section 4.5.

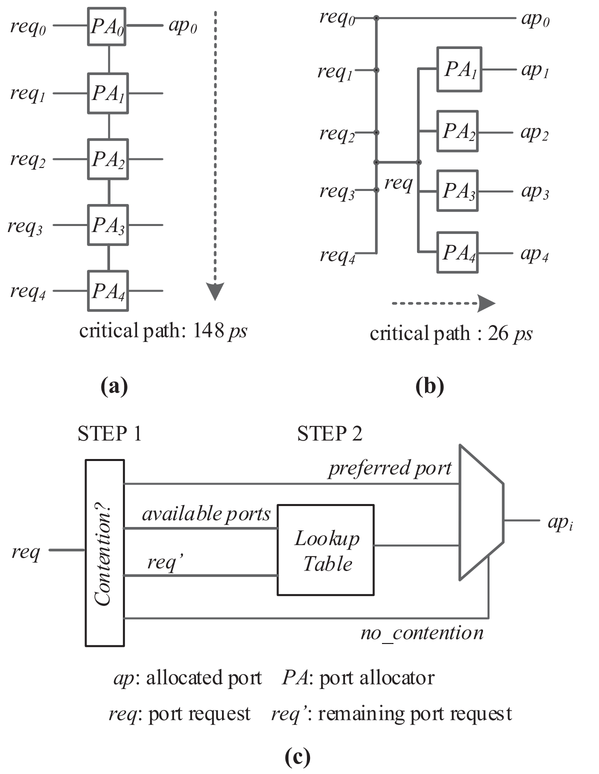

Fig. 5. (a) Sequential port allocation used by BLESS, (b) DeC performs parallel port allocation, and (c) the proposed parallel port allocator.

Example. Consider five flits with their desired destination ports of $\{flit_0 \rightarrow West,\ flit_1 \rightarrow East,\ flit_2 \rightarrow North,\ flit_3 \rightarrow East,\ flit_4 \rightarrow East\}$. In STEP 1, *West* port is allocated to $flit_0$, if $flit_0$ has the highest priority. Also, $flit_2$ can take *North* port since it leads to no contention. The unallocated ports after STEP 1 are $\{Bypass,\ South,\ East\}$. Port allocation in STEP 2 takes place in parallel. $flit_1$ takes *Bypass* port, since a flit in the lower channel (i.e., $flit_0$) has found a port in STEP 1. $flit_3$ takes the second available port — *South*, knowing that *Bypass* port has been taken by $flit_1$ as it fails to find a port in STEP 1. Likewise, $flit_4$ takes the third available port — *East*. Hence, final port allocation for all flits is: $\{flit_0 \leftarrow West,\ flit_1 \leftarrow Bypass,\ flit_2 \leftarrow North,\ flit_3 \leftarrow South,\ flit_4 \leftarrow East\}$. Note that $flit_1$, which would otherwise be deflected, takes the sub-optimal port of *Bypass*.

4.4 Flits Prioritization

DeC prioritizes flits from four neighboring routers, without considering any bypassed flit. This reduces the hardware cost and prevents a bypassed flit from interfering with others in the corresponding subnetwork. The newly injected local flit always has the lowest priority.

Since DeC chooses only the highest priority flit for the top most channel, a 2-stage partial permutation network based on Bitonic sorting [14] suffices, with each of its permutation blocks (i.e., those with arrows in Fig. 4) either passing or swapping two flits according to their time stamps. The remaining flits are partially sorted.

4.5 Ejection and Injection

To form a bypass ring at every node, DeC requires each router to have an additional Bypass port. The direct way of adding a port leads to a significant hardware cost hike, since the complexity of a crossbar scales quadratically with respect to the port count. To avoid this hike, local ejection and injection are handled separately in order to avoid complicating the crossbar. At each cycle, a locally destined flit is removed from the ejector. The injection is granted as long as there is at least one unoccupied channel in the router of the corresponding subnetwork. When multiple subnetworks have unoccupied channels, DeC picks the one with the lowest load to improve load balancing. Handling ejection prior to injection is likely to reduce injection throttling and improve throughput when a router is heavily loaded.

4.6 Router Microarchitecture Design Tradeoffs

The DeC router trades some accuracy for simplicity and parallelism, as a result of two aspects. First, partial sorting may cause some high priority flits to experience longer network latencies. Second, port allocation is oblivious to flits' preferences, except for the highest priority flit. However, a net performance gain is still observed, resulting from the following reasons.

1. The probability of contention is reduced significantly since multiple subnetworks increase path diversity.
2. Contention penalty is mitigated due to the existence of the bypass ring. A contending flit is likely to claim its desired port in another subnetwork with only one extra cycle delay.
3. Critical path reduction allows routers to operate at a higher frequency that lifts DeC bandwidth.
4. The routing decision of a torus-based router follows the X-Y routing mechanism and considers the flit destination to determine the forwarding direction (left or right, up or down) for performance improvement and energy savings.

5 Evaluation Methodology

The proposed DeC is evaluated using a modified cycle-accurate simulator [17]. Our simulator faithfully models a directory-based (i.e., MOESI) coherence protocol. We also model a perfect shared L2 cache to stress the network such that all L1 misses hit in their destination L2 slices. For multthreaded simulation for NoC evaluation, we use the gem5 simulator [62] to obtain execution traces and associated coherence traffic when running PARSEC benchmark applications [61] on multi-core systems. With each core running an instance, every simulation continues until every core has retired 20M instructions (when the results are found to reach steady states). Each synthetic simulation runs for 20M cycles as well. We ensure that NoCs are stressed enough before collecting the statistics. Table 1 lists the key parameters of our evaluation.

We implement BLESS, MinBD, and DeC routers in Verilog to evaluate power, timing, and area via full synthesis using Cadence Encounter RTL Compiler with 15 *nm* FinFET-based Open Cell Library [18] (under $V_{dd} = 0.8\,\mathrm{V}$ and $T = 25°\mathrm{C}$). Since dynamic power consumption occurs whenever a component is toggled (e.g., to read from /write to a register), we fully track the switching activity of each major component in a router during simulation. The statistics are then employed

TABLE 1
Key Parameter Used in Our Evaluation

Coherence Protocol	MOESI
Processor	OOO CPU, 128-entry instruction window
L1 Cache	64KB per core, 4-way set associative, 64B block size, 2-cycle latency, 16 MSHRs
L2 Cache	Perfected and shared LLC (S-NUCA) [29]
Network Topology	Mesh and Torus
Benchmark	Data packet = 64B, control packet = 16B
Synthetic Traffic	50% Data packets and 50% control packets

to calculate the real toggling rate to obtain the final dynamic power of the network. The overall power is simply the sum of static and dynamic power components. Our full RTL synthesis for each router design includes logics for route computation, partial permutation network, port allocation, local ejection and injection, crossbar, side buffers, and pipeline registers. The pipeline registers are clock-gated for reducing the dynamic power of the clock. We observe ~58 percent of power reduction after applying clock gating at the expense of ~14.2 percent of area increase for each design.

6 Results

In this section, we compare our mesh-based DeC with BLESS [5] and MinBD [6] in terms of performance metrics for 4×4, 8×8, and 16×16 NoCs, where BLESS is the most well-known bufferless design and MinBD is the most recent optimization. Performance metrics of interest include the deflection rate, flit latency, NoC throughput, and speedups. We faithfully modeled BLESS and MinBD based on what were described in their original papers. BLESS contains a 3-stage permutation network to prioritize the incoming flits and port allocation strictly honors flit ranking. MinBD performs dual-ejection at each hop and every router is furnished with one side buffer. The flit can stay in the side buffer for at most 2 cycles and we allow at most one flit to be reinjected from the side buffer at each cycle. DeCs with 1, 2, and 4 subnetworks are denoted as DeC1, DeC2, and DeC4, respectively.[3] In this result section, we assume a two-subnetwork setup, unless specified otherwise, as DeC2 provides more balanced trade-offs, to be outlined in Section 6.3.

In addition, we derive full layouts of 4×4, 8×8, and 16×16 DeC-NoCs with the torus topology using Cadence Encounter RTL Compiler under a 15 *nm* process and Cadence Innovus for energy and speed comparison against their mesh-based counterparts. Layout and comparative results are detailed in Section 6.5.

6.1 Under Real Trace Workloads

For multi-programmed simulation, we use the normalized speedup (denoted by the ratio of instructions per cycle (*IPC*) of an application when running under DeC (or MinBD) to that of the same application when running under the baseline bufferless network – BLESS.

3. We assume that aggregate link width for transferring data payloads equals 32 bytes for BLESS, MinBD, and DeC. Each flit carries a header, which is transferred on a separate link. The header size is 4 bytes for DeC and BLESS, and 3 bytes for MinBD (which does not require to carry the timestamp along with each data packet). Each subnetwork of DeC2 and DeC4 contains a 4-byte link for transferring the header.

We randomly construct 16 multithreaded workloads out of 6 applications from PARSEC benchmark suite [61]. Each real workload involves 16 (or 64) program instances executed on the 16 (or 64) nodes of 4×4 (or 8×8) mesh-based NoC. Each node contains a core, private L1, and a slice of the shared LLC to realize static non-uniform cache architecture (S-NUCA) [29]. Workloads are categorized as being Low and High, based on resultant network traffic intensity measured by MPKI (miss per kilo instructions). Fig. 6 shows the deflection count (per injected flit), latency, and speedup results of workload applications from the benchmark suite under DeC2 (i.e., DeC with two subnetworks). It illustrates the results of two representative applications with Low and High traffic loads for each NoC size. The measured traffic load of each scenario is given by the rightmost bar in its respective deflection count bar group. Shown latency and speedup results are normalized with respect to those of BLESS (= 1.0). The results of MinBD are also included in Fig. 6 for comparison.

From the figure, DeC2 is seen to reduce the deflection count (per flit) drastically, when compared with BLESS. In particular, it makes 8×8 NoC achieve some 90 percent deflection reduction under Low and High traffic loads (both exceeding 0.11 packet per cycle per node). For 4×4 NoC, DeC2 yields 85 percent deflection reduction under High load (of 0.08 packet per cycle per node). Drastic deflection reduction often translates to substantially lower latencies, especially for High load where DeC2 drops mean latency by 80 and 78 percent respectively for 4×4 and 8×8 NoCs. This confirms our DeC design goal of cutting down deflection to accelerate packet delivery in NoCs, thus improving execution performance as evidenced in Fig. 6. DeC2 speeds up execution under High load impressively by 21 and 8.3 percent for 8×8 and 4×4 NoCs, respectively, in comparison to BLESS. Instead of deflecting flits upon conflicts at a router, MinBD pushes a conflicted flit into the side-buffer of the router (if possible) for latency reduction. As can be seen in Fig. 6, however, MinBD yields latency reduction only under vary light network traffic (e.g., 4×4 NoC with Low network load). In fact, it slows down execution for 8×8 NoC under both Load and High loads (where deflection counts are far higher than those of DeC2 and BLESS). DeC2 outperforms BLESS and MinBD consistently under real benchmark traces, and its gains widen for heavier traffic loads.

- *Energy and Power Efficiency*

Fig. 7 depicts NoC energy breakdowns of each design under multi-threaded PARSEC traces in 4×4 and 8×8 mesh-based NoCs. For fair comparison, all designs are operated under the slowest clocks of MinBD. The figure reveals that DeC2 lowers energy consumption markedly when compared with BLESS, respectively by 21 and 47 percent for Low and High loads under 4×4 NoC, whereas energy consumption is reduced respectively by 29 and 51 percent for Low and High loads under 8×8 NoC. In general, a larger NoC size yields more energy consumption reduction by DeC2 for real traffic workloads, because packet delivery then takes higher energy so that better room for energy reduction exists.

MinBD also attains commendable energy reduction when compared with BLESS, as a result of adding a side buffer (Sidebuf) to curb deflection and of substituting a crossbar (Xbar) with a partial permutation network (PN) [6].

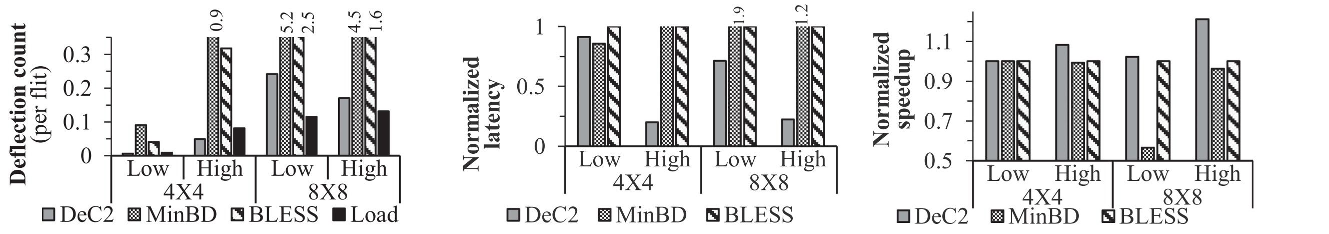

Fig. 6. Deflection count (per flit), latency, and speedup results of multi-threaded PARSEC benchmark workloads in 4×4 and 8×8 mesh-based NoCs, with the network load (in packet/cycle/node) included in the deflection count chart and every result for latency and speedup normalized to that of BLESS ($= 1.0$).

Nonetheless, DeC2 still reduces mean energy consumption by 20 percent (or 30 percent) for 4×4 (or 8×8) NoC in comparison to MinBD. The energy savings mainly results from significant deflection reduction. While clock gating is adopted to reduce clocking energy, pipeline registers still account for more than 50 percent of overall energy in every design. It is thus evident that buffers are not only expensive but also power-hungry, advantaging bufferless routers. Besides pipeline registers, the 3-stage full PN consumes substantial energy in BLESS as well. For MinBD, the additional side buffer and logics for redirecting and injecting flits are the second-largest energy consumer, taking ~10 percent of total energy in all workload categories. For DeC2, the crossbar consumes some 14 percent (or 20 percent) of overall energy in the Low (or High) multi-threaded workload for both network sizes, whereas the additional stage for supporting local injection and ejection takes about 7.4 percent (or 4.5 percent) of energy in the Low (or High) multi-threaded workload under both NoCs. Fig. 7 signifies that DeC is always far more energy-efficient than its BLESS and MinBD counterparts.

6.2 Under Synthetic Traffic Loads

In addition to evaluation under real application traces, we also measure DeC with different subnetwork configurations (i.e., DeC1, DeC2, and DeC4) to gauge its sensitivity to the number of subnetworks under uniform random traffic. This measure sheds light on its general characteristics across a wide range of network demands: from near-zero to saturation loads. From the measured results shown in Fig. 8, we arrive at following observations.

First, when the offered load is low (< 0.1), the deflection count is almost zero, with each packet then likely to take the shortest path (see Fig. 8a). The average latency in Fig. 8b is dominated by the serialization latency (due to injecting those constituent flits of a packet in sequence) and near-zero load latency. The latency of DeC2 is then slightly lower than those

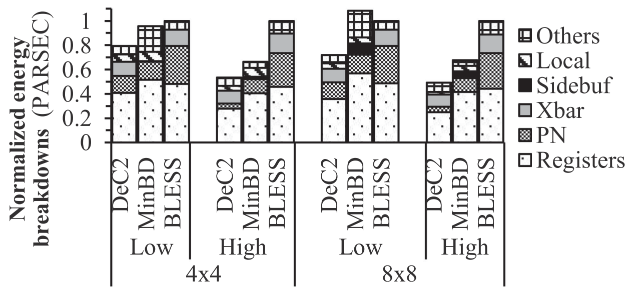

Fig. 7. Network energy breakdowns under PARSEC benchmark traces, with results all normalized against those of BLESS under 4×4 and 8×8 mesh-based NoCs.

of BLESS and MinBD since contention rarely occurs. If the offered load rises, the deflection counts of BLESS and MinBD increase promptly (see Fig. 8a), whereas DeC2 only encounters moderate hikes. DeC2 reduces the latency markedly when compared with BLESS (or MinBD) at the workload that nearly saturates BLESS (or MinBD).

Second, DeC2 has marginal improvement in throughput over BLESS and MinBD prior to their respective saturation loads (see Fig. 8c). However, DeC2 saturates at a much higher load than BLESS and MinBD, as it offers far better path diversity and load balance. Since DeC2 is less likely to exhibit contention, its new flit injection is less likely to fail, boosting up its throughput. In addition, an extra port at each node of DeC2 naturally enjoys higher bandwidth, improving NoC throughput.

Overall, DeC2 exhibits prominent gains on all metrics of interest and outperforms both MinBD and BLESS. Specifically, when compared to BLESS (or MinBD), DeC2 achieves mean reduction of 68 percent (or 77 percent) in the deflection count right before saturation, to yield substantial latency reduction accordingly.

6.2.1 *Sensitivity to the Number of Subnetworks*

DeC2 with two subnetworks achieves significant energy savings and performance gains, from Figs. 6 and 7. It is yet to unveil if more or fewer subnetworks with equivalent aggregated link width (of 256b) are preferred for DeC-based NoCs. To this end, results of two other variations: DeC1 and DeC4, which have respectively one subnetwork and four subnetworks, are included in Fig. 8. Two observations are made from the figure.

First, compared with BLESS and MinBD, DeC1, DeC2, and DeC4 enjoy far smaller deflection rates when workloads are below the saturation point of MinBD. DeC4 has the lowest deflection rate among all DeC variations. This confirms the fact that more subnetworks provide higher path diversity to better curb deflection.

Second, when the load is low, deflection reduction does not yield proportionally shortened latencies as the latency then is dominated by both the serialization latency and the distance between the source and the destination. In this case, the degree of contention is effectively alleviated by the bypass link. On the other hand, when the load increases, DeC1 saturates quickly due to limited path diversity. Although DeC4 saturates at a slightly higher load, it reduces deflection only by 4.0 percent, compared with DeC2. Meanwhile, DeC4 suffers from slightly lower throughput than DeC2 due to its doubled overhead in carrying the flit header (see Fig. 8c). Hence, DeC2 is preferred, to serve as our DeC representative design for comparison with its counterparts subsequently.

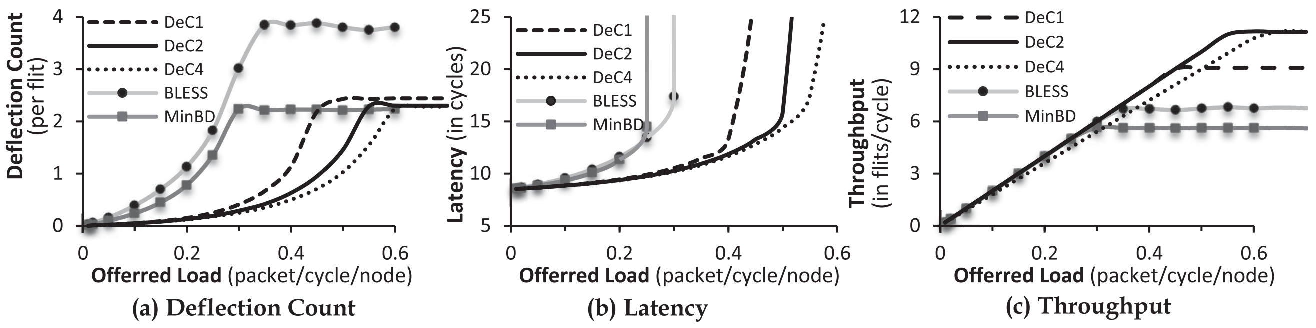

Fig. 8. Synthetic traffic results of BLESS, MinBD, and DeC with 1, 2, and 4 subnetworks for a 4×4 mesh.

6.3 Scalability

We adopt uniform random traffic in 8×8 and 16×16 mesh-based NoCs to study how the latency and the deflection count vary as the NoC size rises, with all network parameters held identical to those for evaluating the 4×4 NoC. The results for DeC2, MinBD and BLESS are delineated in Fig. 9, where DeC2 is demonstrated to outperform consistently for large sizes. Compared with MinBD in an 8×8 (or a 16×16) NoC, DeC2 reduces the deflection count by 4.8 (or 6.9) at the workload that nearly saturates MinBD (i.e., at the offered load of 0.13 (or 0.06)), exhibiting substantial latency gaps accordingly. Similar trends exist when comparing DeC2 with BLESS in 8×8 and 16×16 NoCs. Hence, DeC2 is a scalable solution, able to reach a larger gap in deflection reduction as the size rises.

6.4 Hardware Complexity

In order to evaluate area and timing results of different router designs accurately, we implement BLESS, MinBD, and DeC routers in Verilog and synthesize them via Cadence Encounter RTL Compiler with 15-*nm* FinFET-based Open Cell Library [18]. The RTL implementation results are summarized in Table 2, with those of DeC2 routers for both mesh-based and torus-based NoCs listed.

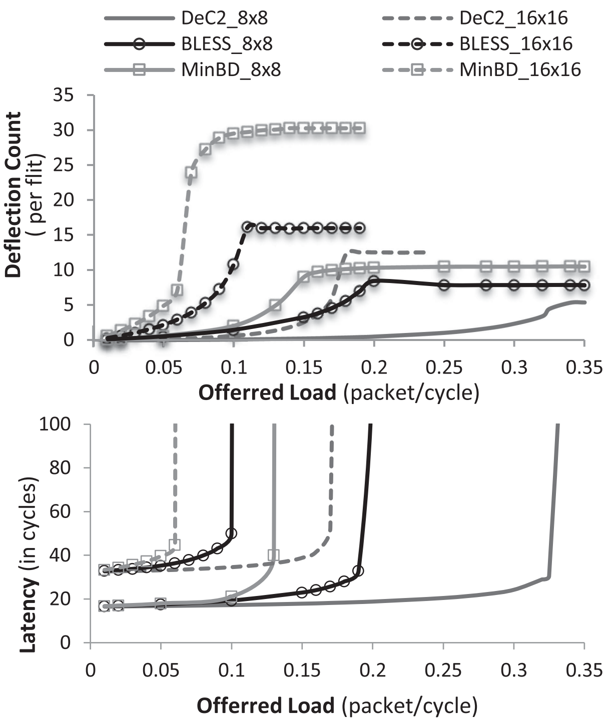

Fig. 9. Deflection count and latency results versus offered load for uniform random traffic under 8×8 and 16×16 mesh-based NoCs equipped with DeC2.

Timing. the long critical path of BLESS router is mainly caused by the first pipeline stage where a 3-stage permutation network is used to fully sort 4 incoming flits from neighboring routers. MinBD has the longest critical path due to three additional components in the first pipeline stage: dual-ejection, redirection of a flit to the side buffer, and reinjection of the flit back to the network. As DeC only uses a 2-stage partial permutation network in its first pipeline stage, the critical path is dictated by its second pipeline stage. Although DeC adds 44 *ps* latency overhead for local injection and ejection (as shown in Fig. 4), given that DeC reduces the latency of port allocator from 148 *ps* to 26 *ps*, DeC reduces the critical path markedly when compared with BLESS and MinBD, leading to its timing of 0.17 *ns*. This signifies the critical path reduction of 23 percent (from 0.22 *ns*) for BLESS, and of more pronounced reduction for MinBD (from 0.26 *ns*). Interestingly, the DeC2 router for torus-based NoCs takes only negligibly more area than that for mesh-based NoCs with its timing kept unchanged.

Area. DeC reduces the complexity of one single sub-router by some 42 percent in comparison to that of the BLESS router. As each DeC2 router contains two subrouters, it takes more area than a BLESS router. Although MinBD has the smallest area, it operates at a much slower clock rate than BLESS and DeC. The hardware cost of DeC stems from two reasons. First, since each flit needs to carry a header containing necessary information (along a separate control path) to make a routing decision, an extra control path is required for each subnetwork (when compared to a single network with the same link width and one single control path). Second, an additional bypass port exists to bridge subnetworks, establishing a bypass ring for deflection containment. However, as the fraction of chip's transistors to become dark silicon increases in the future, the chip area is considered to be relatively less essential than energy consumption [1]. Harvested energy may be reapplied to further improve performance. Since

TABLE 2
RTL Implementation Results of a Single Router Under a 15-nm Process

	BLESS	MINBD	DeC2 (mesh)	DeC2 (torus)
Timing (*ns*)	0.22	0.26	0.17	0.17
Area (μm)	12386	12386	14285	14352

TABLE 3
Overall Hardware Cost Comparison of Various Sized Torus- and Mesh-Based DeC2-NoCs, Implemented Under a 15-*nm* Process (via Cadence Encounter RTL Compiler and Cadence Innovus)

Item	Torus 4×4	Torus-8×8	Torus-16×16	Mesh-4×4	Mesh-8×8	Mesh-16×16
Time (*ns*)	0.17	0.17	0.17	0.17	0.17	0.17
Leakage power (μW)	8.14E+03	3.27E+04	1.31E+05	7.01E+03	3.06E+04	1.27E+05
Internal power (μW)	2.97E+05	1.20E+06	4.81E+06	2.48E+05	1.13E+06	4.67E+06
Net power (μW)	1.92E+05	7.74E+05	3.08E+06	1.59E+05	7.17E+05	3.01E+06
Dynamic power (μW)	4.89E+05	1.98E+06	7.89E+06	4.06E+05	1.85E+06	7.67E+06
Total power (μW)	4.97E+05	2.01E+06	8.02E+06	4.13E+05	1.88E+06	7.80E+06
Core dimension (μm^2)	570×566	1141×1136	2283×2268	530×528	1105×1101	2250×2242

DeC2 cuts down the deflection count and latency significantly (from Figs. 8 and 9), its increased area overhead is warranted and reasonable.

6.5 Torus-Based NoCs With DeC Support

DeC is readily applicable to torus-based NoCs for performance improvement and energy savings. With is bisection bandwidth doubled in comparison to its mesh counterpart [13], the torus has the potential for exhibiting higher offered loads, especially advantageous to bufferless NoCs. Albeit to its potential benefits, bufferless torus-based NoCs have never been explores, possibly due to intuitive concerns of (1) more complex routing control at each router as stated in Section 4.2 and (2) larger layout areas involved to slow down operating frequencies due to longer worst link length.

- *Router Implementation & Post-Synthesis Full Layouts*

We develop the torus-based NoC router of DeC2 and implement it in Verilog synthesis via Cadence Encounter RTL Compiler with a 15-*nm* process [18]. We also accomplish post-synthesis full layouts of 4×4, 8×8, and 16×16 torus-based DeC2-NoCs via Cadence Innovus. In contrast to conventional intuitions, our RTL implementation of the torus-based router reveals its fastest possible timing of $0.17\,ns$, which is identical to that of the mesh-based DeC2 router, as listed in Table 2. The full layout timing and other related details we obtained by post-synthesis via Cadence Innovus are summarized in Table 3. When compared with their mesh-based counterparts, the torus-based DeC2-NoCs enjoy identical fastest possible timing.

Timing. All sized tori (4×4, 8×8, 16×16) successfully synthesized for the clock period of 0.17 ns, which is the fastest possible timing of a single DeC2 router for mesh- and torus-based NoCs (see Table 2). The synthesized outcomes indicate that tori with the dimension up to 16×16 (which is the largest synthesizable size under our Cadence license) for 15-*nm* FinFET is not the limiting factor on the clock frequency, avoiding torus-based NoCs from suffering from slower clocking *due to longer worst link length* therein.

Area. After placement and routing with Cadence Innovus under 70 percent density for the design, the core area is almost of a square shape. The core area increases proportionally as the torus size rises. For example, the core area of the 16×16 torus is 4× (or 16×) that of the 8×8 (or 4×4) torus, as listed in Table 3. The torus takes slightly more area than its compatible mesh, whose routers along the edges are less complex. For example, a 16×16 torus-based DeC2-NoC takes 2.6 percent (i.e., 2283×2268 versus 2250×2242) more area than its mesh-based counterpart.

Power. Dynamic power values in Table 3 represent the sum of internal power and net power (i.e., power loss in the wire). In general, net power accounts for ~38 percent and internal power for ~60 percent, with the remaining ~2 percent due to leakage power. Separetely, due to its more complex edge routers, the torus DeC2-NoC consumes more power as compared to its mesh counterpart, but the gap shrinks as the size rises. For example, the 16×16 torus-based NoC consumes 2.8 percent more power (i.e., $8.02\text{E} + 06$ vs. $7.80\text{E} + 06$), as opposed to 6.7 percent for the size of 8×8, than its mesh-based counterpart.

- *Performance Under Real Execution Traffic Workloads*

The deflection count and latency results measured using 16 multi-threaded workloads from the PARSEC benchmark suite are illustrated in Fig. 10. Each real workload contains 16 (or 64) instances of execution traces to independently drive all cores of 4×4 (8×8) NoC for evaluation. From the figure, the deflection count (per flit) is 50 percent (or 4 percent) less under the torus than under the mesh for 8×8 NoC with Low (or High) traffic, resulting in latency reduction by 12 percent (or 20 percent) under the torus due to an increase in its bisection bandwidth. Although the deflection counts for Low traffic load under 4×4 NoC is extremely small (of 0.02), noticeable latency reduction (by 28 percent) results from the torus. Speedup improvement under torus-based NoC as compared to that under its mesh-based counterpart for Low (or High) traffic in 4×4 NoC and 8×8 NoC are 0.1 percent (or 1.1 percent) and 0.6 percent (or 2.7 percent), respectively.

Note that DeC2 exhibits limited speedups for 4×4 and 8×8 torus-based NoCs (over their mesh-based counterparts) due to the fact that network traffic loads under real execution traces are rather light (always ≤ 0.153 packet per cycle per node, as illustrated in the deflection count chart in Fig. 10); under such light loads, both NoC topologies have similar execution times for a given benchmark application, as confirmed also by the throughput results (over a wide range of NoC loads) depicted in Fig. 13 for synthetic workloads discussed next.

- *Performance Evaluation Under Synthetic Workloads*

Three different traffic types of synthetic workloads are considered in evaluating a range of torus-based DeC2- NoCs, including uniformly random (UR) traffic, Tornado (TR) traffic, and Bit Complement (BC) traffic [13], [59]. Under BC traffic, the destination address is the bit-wise inversion of the source address (i.e., $d_i = \sim\ s_i$), whereas the digits of the destination address under TR traffic are computed from the digits of the source address (i.e., $d_x = s_x + (k-1)/2$ mod k, with digits

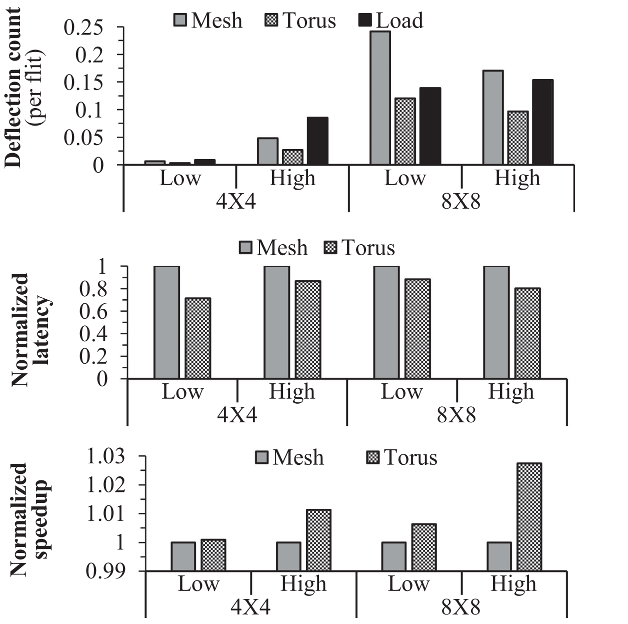

Fig. 10. Deflection count (per flit) and normalized latency in a mesh-based and torus-based NoCs employing DeC with two subnetworks under real traffic loads of PARSEC application traces, with the network load (in packet/cycle/node) included in the deflection count chart.

being radix k-numbers). A uniform random process chooses the destination address under UR traffic. Workloads of each traffic type generated for evaluation vary from near-zero to

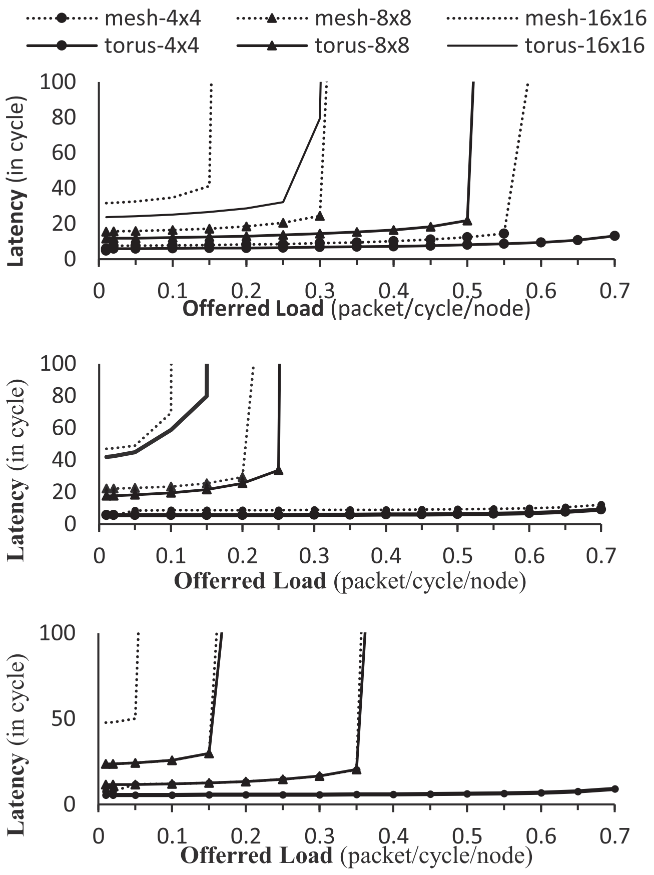

Fig. 11. Latency versus offered load (packet/cycle/node) for synthetic traffic governed by three different models: UR (top), TR (mid), and BC (bottom), under mesh-based and torus-based DeC2-NoCs sized 4×4, 8×8, and 16×16.

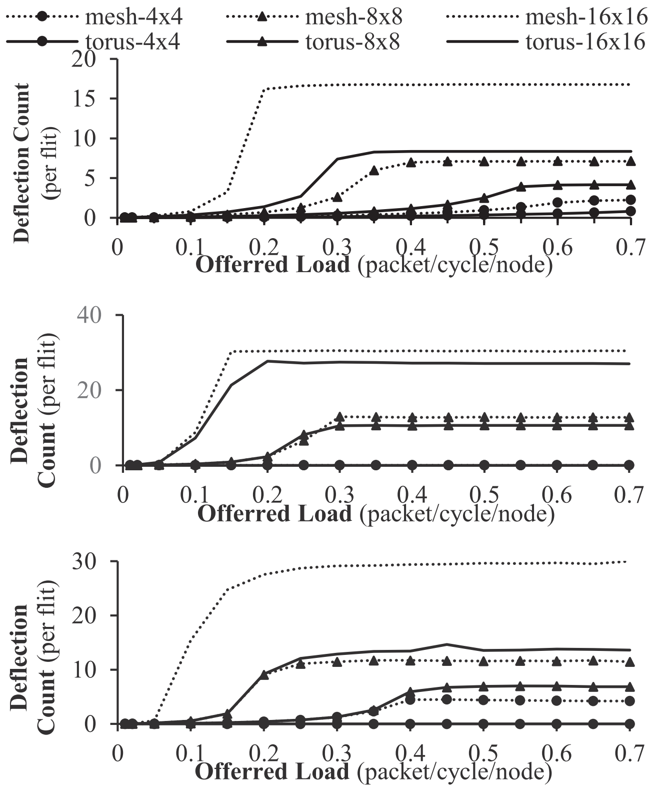

Fig. 12. Deflection count per flit versus offered load (packet/cycle/node) for synthetic traffic governed by three different models: UR (top), TR (mid), and BC (bottom), under mesh-based and torus-based DeC2-NoCs sized 4×4, 8×8, and 16×16.

the maximum capacity of DeC2-NoCs, which is close to 0.67 packet/cycle/node, irrespective of the NoC size. NOCulator [17], [47], [57], an open-source cycle-accurate network-on-chip simulator, is employed to evaluate NoCs sized 4×4, 8×8, and 16×16, with the results illustrated in Figs. 11, 12, and 13.

The latency curves in Fig. 11 reveal that under a low offered load (denoted as $\boldsymbol{\beta}$), say $\boldsymbol{\beta} < 0.05$, all injected packets in both torus-based and mesh-based DeC2-NoCs likely take the shortest path due to then almost deflection-freeness for any size and traffic pattern (see Fig. 12), giving rise to the minimum latency in a respective NoC. Naturally, the minimum latency NoC is noticeably smaller in a torus-based NoC than in a compatible mesh-based NoC especially when its size is large. If $\boldsymbol{\beta}$ rises, deflection counts under mesh-based NoCs increase quickly as uncovered in Fig. 12, whereas torus-based NoCs only encounter moderate deflection count hikes. According to Fig. 11, the 16×16 torus-based DeC2-NoC under BC, TR, and UR traffic patterns saturates respectively on $\boldsymbol{\beta} > 0.15$, $\boldsymbol{\beta} > 0.20$, and $\boldsymbol{\beta} > 0.30$, whereas its mesh-based counterpart saturates respectively on $\boldsymbol{\beta} > 0.05$, $\boldsymbol{\beta} > 0.10$, and $\boldsymbol{\beta} > 0.15$. As a result, the 16×16 torus-based NoC can sustain up to $2.33 \times (= (\frac{0.15}{0.05} + \frac{0.2}{0.1} + \frac{0.3}{0.15})/3)$ loads than a mesh-based counterpart over the three synthetic traffic types. Likewise, 8×8 and 4×4 torus-based NoCs can sustain far higher loads than their compatible mesh-based NoCs. Interestingly, the 4×4 torus-based DeC2-NoC never saturates, able to sustain the maximum network load. Given that every packet injected into a bufferless NoC always reaches its destination, the maximum throughput of such a NoC equals its sustained injection rate.

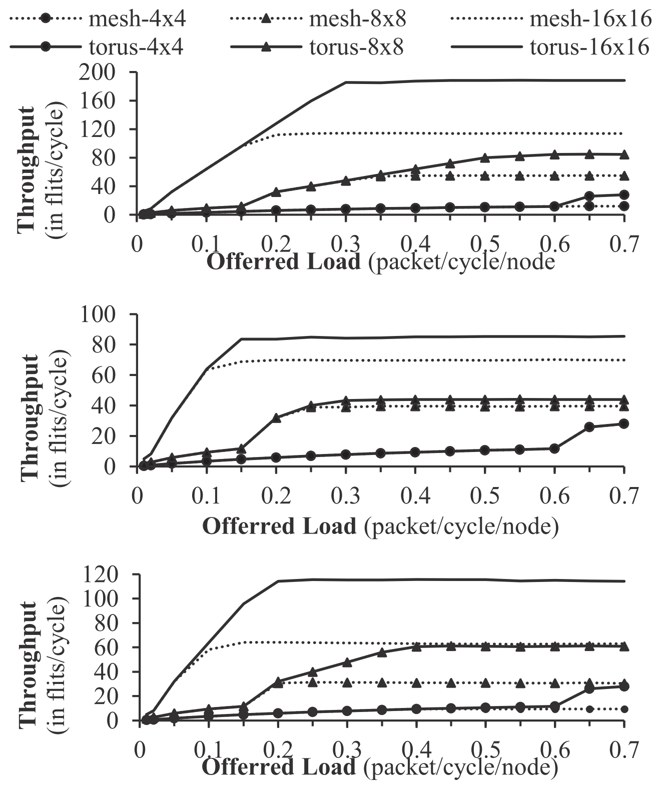

Fig. 13. Throughput vs. offered load (packet/cycle/node) for synthetic traffic governed by three different models: UR (top), TR (mid), and BC (bottom), under mesh-based and torus-based DeC2-NoCs sized 4×4, 8×8, and 16×16.

According to Fig. 11, the torus-based DeC2-NoCs markedly reduce the latency amounts under all three traffic types, especially under UR and BC and for the size of 16×16, when compared with their compatible mesh-based NoCs. Fig. 13 plots the throughput results of torus- and mesh-based NoCs for the three synthetic traffic types. The figure reveals that saturated throughputs of 16×16 DeC2-NoC under UR and BC are substantially larger for the torus-based NoC than for its mesh-based counterpart.

Overall, torus-based NoCs exhibit prominent gains on all performance metrics of interest in comparison to their mesh-based counterparts. They are so achieved without compromising the clock rate, in contrast to common intuitions and making them especially advantageous to bufferless NoCs.

6.6 Put It All Together

DeC reduces contention occurrences by employing multiple independent subnetworks to increase path diversity. As a result, the average hop count for each packet delivery is reduced. Further, when contention cannot be avoided, its adverse impact can be mitigated due to the existence of a bypass ring. The flit, which is otherwise deflected, can be sent to the corresponding router in another subnet and then contend for the destined port with only a single extra cycle delay (instead of multiple extra cycles upon deflection). Finally, critical path reduction allows routers to operate at a higher frequency, boosting network bandwidth. Hence, DeC can attain superior power efficiency and performance improvement, rendering the network to saturate at much higher loads, as a result of substantial deflection containment.

7 Discussion

7.1 Deadlocks and Livelocks

For a fair comparison with BLESS, we did not use any flow control in DeC. Like earlier work [4], infinite reassembly buffers at the destination end are assumed to avoid potential network level deadlocks for both BLESS and DeC. However, the Retransmit-Once protocol [4] can be easily applied to both designs to arrive at more realistic reassembly buffer sizing. As the oldest flit always takes the desired port, DeC can also ensure livelock freeness (i.e., end-to-end delivery guarantee) and protocol level deadlock-freeness (i.e., replies always to reach their destinations).

7.2 Wormhole Routing

Since flits are routed independently across the NoC, BLESS requires each flit to carry a header. WORM-BLESS [5] uses wormhole routing in a bufferless NoC so that it requires only the first flit of a packet to carry a header for setting up the path. All subsequent flits simply follow the path created by the head flit. However, WORM-BLESS is less appealing than BLESS in that it incurs more deflection and thus lower performance. The reason is that if the head flit of a packet is deflected, all its subsequent flits will be deflected as well since they follow the head flit. We have developed the DeC2 router that operates under wormhole routing. Each link of the developed router completely drops its additional wires that are to carry full header information (including flit sequence number, the source and the destination node IDs, time stamp, etc.), so that its link width is reduced to exactly the flit size. Packet transmission over the NoC built with such DeC2 routers uses the very first flit to carry full header information for the wormhole control path setup, and the payload flits of a packet follow the head flit.

Under wormhole routing, a deflected header flit leads to all its payload flits to be deflected in a router over subsequent cycles, *no matter whether the preferred port is available or not* during those cycles. Hence, wormhole routing is subject to lower flexibility in port selection at a router, reducing the NoC offered load. In addition, the wormhole router has to "truncated" on-going packet transmission at a port that is required by a newly arrived high priority packet (to avoid the livelocks, as stated earlier [5]). Our developed wormhole router for DeC2 is provisioned with packet truncation. Each packet truncation, however, adds one extra header flit (which heightens the load) to the NoC, further degrading its performance. We have evaluated DeC2 built by our developed router with wormhole support, confirming its inferior performance. For the packet payload of 64 bytes and the flit size of 16 bytes under uniform random traffic, 4×4 (or 8×8) mesh-based wormhole DeC2-NoC is found to reach the maximum traffic load that approaches saturation at 0.2 (or 0.07) packet/cycle/node (not shown). In comparison, the near saturation traffic loads of compatible mesh-based DeC2-NoCs without wormhole routing stay far higher at 0.55 and 0.34 (from Figs. 8 and 9). Similarly, 4×4 (or 8×8) torus-based wormhole DeC2-NoC has the maximum traffic load equal to 0.3 (or 0.14) packet/cycle/node according to our evaluation (not shown), markedly lower than those achieved without wormhole routing (see Fig. 11).

7.3 Power Gating

The network may be underutilized if its load is low. Low network utilization wastes significant static energy. Fortunately, in DeC, the router in a lightly loaded subnetwork can be power-gated to reduce power consumption. Since subnetworks are bridged to get a bypass ring in each node, turning off one subnetwork still retains partial deflection containment ability without compromising full connectivity. However, there are two challenges to be addressed. First, since the deflection rate of bufferless NoC is closely coupled with path diversity and bypass links, disabling a subnetwork may lead to performance reduction if the network load rises. Second, unlike buffered NoCs, in which flits can stay in local buffers when a connected router is inactive, a bufferless NoC cannot hold flits. In such a scenario, the deflection rate may increase. Using techniques to wake up a router early, such as look-ahead routing [16], or waking up the downstream router as long as the destination of L1 miss is known [15], may be necessary to hide the wakeup latency and reduce deflection. This is left for future exploration.

8 Conclusion

While bufferless NoC is a promising technique to reduce power consumption and the hardware cost, its high deflection leads to severe performance degradation and power waste. We have proposed Deflection Containment for bufferless NoCs to improve performance and save energy by employing multiple subnetworks. DeC incorporates a bypass channel in each router of one subnet-work to bridge to the corresponding router in another subnetwork. A flit can be “bypassed” from the router where contention occurs to the bridged router of another subnetwork, instead of being deflected away in order to curb deflection. This permits a contending flit to compete for the desired port again with only a single cycle delay, rather than multiple cycles upon deflection. With a relaxed prioritization constraint, port allocation is performed in parallel for all incoming flits, significantly shortening the critical path and markedly improving network bandwidth. Our proposed DeC design is readily applicable to both mesh-based NoCs and torus-based NoCs. We have evaluated resultant mesh- and torus-based DeC2-NoCs extensively using both (1) multi-threaded PARSEC benchmark traffic traces gathered using gem5 for the sizes of 4×4 and 8×8 and (2) various synthetic traffic loads for sizes equal to 4×4, 8×8, and 16×16. Evaluation outcomes confirm that DeC2 leads to substantial deflection containment (by some 90 percent) under real benchmark traffic trace loads for 8×8 mesh-based NoCs, reducing mean energy consumption by up to 51 percent, when compared with BLESS. In addition, the evaluation results under various synthetic traffic loads reveal that a 16×16 torus-based DeC2-NoC can sustain up to $2.33\times$ loads in comparison to its mesh-based counterpart, taking only 2.8 percent more power and 2.6 percent more area based on their full layouts under the 15 *nm* process.

Acknowledgments

This work was supported in part by U.S. National Science Foundation under Grant Numbers: CCF-1423302 and CNS-1527051. Xiyue Xiang and Purushottam Sigdel are contributed equally to this work.

References

[1] M. B. Taylor, “A landscape of the new dark silicon design regime,” in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2013, pp. 8–19.
[2] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz mesh interconnect for a teraflops processor,” in *Proc. 40th IEEE/ACM Int. Symp. Microarchitecture*, 2007, pp. 51–61.
[3] M. B. Taylor *et al.*, “Evaluation of the raw microprocessor: An exposed-wire-delay architecture for ILP and streams,” in *Proc. 31st Int. Symp. Comput. Architecture*, 2004, pp. 2–13.
[4] C. Fallin, C. Craik, and O. Mutlu, “CHIPPER: A low-complexity bufferless deflection router,” in *Proc. 17th Int. Symp. High Perform. Comput. Architecture*, 2011, pp. 144–155.
[5] T. Moscibroda and O. Mutlu, “A case for bufferless routing in on-chip networks,” in *Proc. 36th Int. Symp. Comput. Architecture*, 2009, pp. 196–207.
[6] C. Fallin *et al.*, G “MinBD: Minimally-buffered deflection routing for energy-efficient interconnect,” in *Proc. 6th IEEE/ACM Int. Symp. Netw. Chip*, 2012, pp. 1–10.
[7] W. J. Dally, “Virtual-channel flow control,” in *Proc. 17th Int. Symp. Comput. Architecture*, 1990, pp. 60–68.
[8] M. Hayenga, N. E. Jerger, and M. Lipasti, “SCARAB: A single cycle adaptive routing and bufferless network,” in *Proc. 42nd IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 244–254.
[9] C. Fallin, C. Craik, and O. Mutlu, “CHIPPER: A low-complexity bufferless deflection router,” CALCM, Carnegie Mellon University, SAFARI Technical Report: TR-2010-001, Dec. 29, 2010.
[10] J. Kim, J. Balfour, and W. J. Dally, “Flattened butterfly topology for on-chip networks,” in *Proc. 40th IEEE/ACM Int. Symp. Microarchitecture*, 2007, pp. 172–182.
[11] J. Kim, W. J. Dally, S. Scott, and D. Abts, “Technology-driven, highly-scalable dragonfly topology,” in *Proc. 35th Int. Symp. Comput. Architecture*, 2008, pp. 77–88.
[12] A. Kumar, L. S. Peh, P. Kundu, and N. K. Jha, “Express virtual channels: Towards the ideal interconnection fabric,” in *Proc. 34th Int. Symp. Comput. Architecture*, 2007, pp. 150–161.
[13] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
[14] K. Batcher, “Sorting networks and their applications,” in *Proc. AFIPS Spring Joint Comput. Conf.*, 1968, pp. 307–314.
[15] L. Z. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, “Power punch: Towards non-blocking power-gating of NoC routers,” in *Proc. 21st Int. Symp. High Perform. Comput. Architecture*, 2015, pp. 378–389.
[16] H. Matsutani, M. Koibuchi, H. Amano, and D. Wang, “Run-time power gating of on-chip routers using look-ahead routing,” in *Proc. 13th Asia South Pacific Des. Autom. Conf.*, 2008, pp. 55–60.
[17] R. Ausavarungnirun *et al.*, “Design and evaluation of hierachical rings with deflection routing,” in *Proc. 26th Int. Symp. Comput. Architecture High Perform. Comput.*, 2014, pp. 230–237.
[18] M. Martins *et al.*, “Open cell library in 15nm FreePDK technology,” in *Proc. Int. Symp. Physical Des.*, 2015, pp. 171–178.
[19] H. Kim, C. Kim, M. Kim, K. Won, and J. Kim. “Extending bufferless on-chip networks to high-throughput workloads,” in *Proc. 8th IEEE/ACM Int. Symp. Netw. Chip*, 2014, pp. 9–16.
[20] H. Kim, Y. Kim, and J. Kim, “Clumsy flow control for high-throughput bufferless on-chip networks,” *IEEE Comput. Archit. Lett.*, vol. 12, no. 2, pp. 47–50, Jul. 2013.
[21] R. Ausavarungirun, K. Chang, C. Fallin, and O. Mutlu, “Adaptive cluster throttling: Improving high-load performance in bufferless on-chip networks,” CALCM, Carnegie Mellon University, SAFARI TR-2011-006, Dec. 6, 2011.
[22] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis, “Evaluating bufferless flow control for on-chip networks,” in *Proc. 4th IEEE/ACM Int. Symp. Netw. Chip*, 2010, pp. 9–16.
[23] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, “Catnap: Energy propotional multiple network-on-chip,” in *Proc. 40th Int. Symp. Comput. Architecture*, 2013, pp. 320–331.
[24] D. Wentzlaff *et al.*, “On-chip interconnection architecture of the tile processor,” in *Proc. 40th IEEE/ACM Int. Symp. Microarchitecture*, 2007, pp. 15–31.
[25] M. B. Taylor *et al.*, “The raw microprocessor: A computational fabric for software circuits and general-purpose programs,” in *Proc. 35th IEEE/ACM Int. Symp. Microarchitecture*, 2002, vol. 22, pp. 25–35.
[26] K. Sankaralingam *et al.*, “Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture,” in *Proc. 30th Int. Symp. Comput. Architecture*, 2003, pp. 422–433.

[27] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, “Express cube topologies for on-chip interconnects,” in *Proc. 15th Int. Symp. High Perform. Comput. Architecture*, 2009, pp. 163–174.

[28] A. K. Mishra, O. Mutlu, and C. R. Das, “A heterogeneous multiple network-on-chip design: An application-aware approach,” in *Proc. 50th Des. Autom. Conf.*, 2013, pp. 1–10.

[29] C. Kim, D. Burger, and S. Keckler, “An adaptive, non-uniform cache structure for wire-dominated on-chip caches,” in *Proc. 10th Int. Conf. Architectural Support Program. Languages Operating Syst.*, 2002, pp. 211–222.

[30] M. Koibuchi, H. Matsutani, H. Amano, and T. M. Pinkston, “A lightweight fault-tolerant mechanism for network-on-chip,” in *Proc. 2nd ACM/IEEE Int. Symp. Netw.-Chip*, Apr. 2008, pp. 13–22.

[31] A. Snavely and D. M. Tullsen, “Symbiotic job scheduling for a simultaneous multithreaded processor,” in *Proc. 9th Int. Conf. Architectural Support Program. Languages Operating Syst.*, 2000, pp. 234–244.

[32] L.-S. Peh and W. Dally, “Flit-reservation flow control,” in *Proc. 6th Int. Symp. High Perform. Comput. Architecture*, 2000, pp. 73–84.

[33] A. Abousamra, R. G. Melhem, and A. K. Jones, “Deja vu switching for multiplane NoCs,” in *Proc. 6th IEEE/ACM Int. Symp. Netw. Chip*, 2012, pp. 11–18.

[34] A. Flores, J. L. Aragon, and M. E. Acacio, “Heterogeneous interconnects for energy-efficient message management in CMPs,” *IEEE Trans. Comput.*, vol. 59, no. 1, pp. 16–28, Jan. 2010.

[35] S. Volos C. Seiculescu, and B. Grot, “CCNoC: Specializing on-chip interconnects for energy efficiency in cache-coherent servers,” in *Proc. 6th IEEE/ACM Int. Symp. Netw. Chip*, 2012, pp. 67–74.

[36] J. Balfour and W. J. Dally, “Design tradeoffs for tiled CMP on-chip networks,” in *Proc. 20th Annu. Int. Conf. Supercomputing*, 2006, pp. 187–198.

[37] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, “NoC Architectures for silicon interposer systems,” in *Proc. 47th IEEE/ACM Int. Symp. Microarchitecture*, 2014, pp. 458–470.

[38] J. S. Miguel and N. E. Jerger, “Data criticality in network-on-chip design,” in *Proc. 9th IEEE/ACM Int. Symp. Netw. Chip*, 2015, Art. no. 22.

[39] W. C. Kwon and L. S. Peh, “A universal ordered NoC design platform for shared memory MPSoC,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2015, pp. 697–704.

[40] Z. Li, J. S. Miguel, and N. E. Jerger, “The runahead network-on-chip,” in *Proc. 22nd Int. Symp. High Perform. Comput. Architecture*, 2016, pp. 333–344.

[41] B. K. Daya *et al.*, “SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering,” in *Proc. 41st Int. Symp. Comput. Architecture*, 2014, pp. 25–36.

[42] Arm Limited, “AMBA: The standard for on-chip communication,” 1995-2019, Accessed: Dec. 2019. [Online]. Available: http://www.arm.com/products/system-ip/amba

[43] Arm Limited, “AMBA AXI and ACE Protocol Specification,” 2013, Accessed: Dec. 2019. [Online]. Available: http://infocenter.arm.com

[44] P. Conway and B. Hughes, “The AMD opteron northbridge architecture,” in *Proc. 40th IEEE/ACM Int. Symp. Microarchitecture*, 2007, pp. 10–21.

[45] Intel Corporation, “An Introduction to the intel quickpath Interconnect,” 2009, Accessed: Dec. 2019. [Online]. Available: https://www.intel.com/content/www/us/en/io/quickpath-technology/quick-path-interconnect-introduction-paper.html

[46] R. Ausavarungnirun *et al.*, “Design and evaluation of hierarchical rings with deflection routing,” in *Proc. 26th Int. Symp. Comput. Architecture High Perform. Comput.*, 2014, pp. 230–237.

[47] R. Ausavarungnirun *et al.*, “A case for hierarchical rings with deflection routing: An energy-efficient on-chip communication substrate,” *J. Parallel Comput.*, vol. 54, pp. 29–45, 2016.

[48] K. Chang, R. Ausavarungnirun, C. Fallin, and O. Mutlu, “HAT: Heterogeneous adaptive throttling for on-chip networks,” in *Proc. 24th Int. Symp. Comput. Architecture High Perform. Comput.*, 2014, pp. 9–18.

[49] B. K. Daya, L. S. Peh, and A. P. Chandrakasan, “Quest for high-performance bufferless NoCs with single-cycle express paths and self-learning throttling,” in *Proc. 53rd ACM/EDAC/IEEE Des. Autom. Conf.*, 2016, pp. 1–6.

[50] G. P. Nychis, C. Fallin, T. Moscibroda, O. Mutlu, and S. Seshan, “On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects,” in *Proc. ACM Conf. Appl. Technol. Architectures Protocols Comput. Commun.*, 2012, pp. 407–418.

[51] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu, “Next generation on-chip networks: What kind of congestion control do we need?” in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, Art. No. 12.

[52] M. Thottethodi, A. R. Lebeck, and S. S. Mukherjee, “Self-tuned congestion control for multiprocessor networks,” in *Proc. 7th Int. Symp. High Perform. Comput. Architecture*, 2001, pp. 107–118.

[53] X. Xiang, W. Shi, S. Ghose, L. Peng, O. Mutlu, and N.-F. Tzeng, “Carpool: A bufferless on-chip network supporting adaptive multicast and hotspot alleviation,” in *Proc. Int. Conf. Supercomputing*, 2017, Art. no. 19.

[54] J. Balkind *et al.*, “OpenPiton: An open source manycore research framework,” in *Proc. 21st Int. Conf. Architectural Support Program. Languages Operating Syst.*, 2016, pp. 217–232.

[55] A. Sodani *et al.*, “Knights landing: Second-generation intel xeon phi product,” in *Proc. 49th IEEE/ACM Int. Symp. Microarchitecture*, 2016, pp. 34–46.

[56] F. Alazemi, A. Azizimazreah, B. Bose, and L. Chen, “Routerless networks-on-chip,” in *Proc. 24th Int. Symp. High Perform. Comput. Architecture*, 2018, pp. 492–503.

[57] “NOCulator” [Accessed: 08-April 2019], [Online]. Available: https://github.com/CMU-SAFARI/NOCulator/.

[58] N. R. Adiga *et al.*. “Blue gene/L torus interconnection network,” *IBM J. Res. Dev.*, vol. 49, pp. 265–276, Mar. 2005.

[59] P. V. Gratz and S. W. Keckler, “Realistic workload characterization and analysis for networks-on-chip design,” in *Proc. Chip Multiprocessor Memory Syst. Interconnects*, 2010, pp. 1–10.

[60] A. Singh *et al.*. “Locality-preserving randomized oblivious routing on torus networks,” in *Proc. 14th Annu. ACM Symp. Parallel Algorithms Architectures*, 2002, pp. 9–13.

[61] C. Bienia, S. Kumar, J. Singh, and K. Li, “The PARSEC benchmark suite: Characterization and architectural implications,” in *Proc. 17th Int. Conf. Parallel Architectures Compilation Tech.*, Oct. 2008, pp. 72–81.

[62] N. Binkert *et al.*, “The gem5 simulator,” *ACM SIGARCH Comput. Architecture News*, vol. 39, pp. 1–7, 2011.

Xiyue Xiang received the BE degree in electrical engineering from Xi’an Shiyou University, Xi’an, China, in 2008, and the MS degree in telecommunication, second MS degree in computer engineering, and PhD degree in computer engineering from the University of Louisiana at Lafayette, Louisiana, in 2012, 2014, and 2017, respectively. He is currently a senior design engineer with Xilinx Inc. Prior to join Xilinx, he worked at Carnegie Mellon University as a visiting student researcher, in 2015. His research interests include network-on-chips, application-aware architectural optimization, FPGAs, and data center networks.

Purushottam Sigdel received the BE degree in electronics and communication engineering, in 2000, and MS degree in information and communication from Trubhuvan University, Kathmandu, Nepal, in 2006, and second MS degree in computer science from the University of Louisiana at Lafayette, in 2014. Currently, he is working toward the PhD degree in the Center for Advanced Computer Studies, the University of Louisiana at Lafayette. From 2000 to 2012, he worked as an instructor in the department of electronics and computer engineering in Pulchowk campus, Lalitpur, Nepal. His research interests include data compression, distributed computing, storage systems, and computer system architecture.

Nian-Feng Tzeng (M’86-SM’92-F’10) has been with Center for Advanced Computer Studies, School of Computing and Informatics, the University of Louisiana at Lafayette, since 1987. His current research interest include the areas of high-performance computer systems, and parallel and distributed processing. He was on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems*, 1998 – 2001, and *IEEE Transactions on Computers*, 1994 – 1998. He was the chair of Technical Committee on Distributed Processing, the IEEE Computer Society, from 1999 – 2002. He is the recipient of the Outstanding Paper Award of the 10th IEEE International Conference on Distributed Computing Systems, May 1990, and received the University Foundation Distinguished Professor Award in 1997.