MRCN: Throughput-Oriented Multicast Routing for Customized Network-on-Chips

Young Sik Lee, *Student Member, IEEE*, Yong Wook Kim, *Student Member, IEEE*, and Tae Hee Han, *Senior Member, IEEE*

Abstract—The relentless proliferation of Big Data and artificial intelligence has compelled computing platform architectures to evolve into heterogeneous multicores for greater energy efficiency. A customized network-on-chip (NoC) supporting interconnection diversity is pivotal for the asymmetric data-access traffic requirements of modern heterogeneous multicore system-on-chip (SoC). A significant portion of on-chip data access comprises single-source multi-destination (SSMD) traffic, which supports barrier synchronization, multi-threading, cache coherency protocols, and deep neural network (DNN) acceleration. By amortizing SSMD traffic, multicast routing is essential for effectively utilizing communication bandwidth. One of the primary concerns in supporting multicast routing in NoCs is to circumvent the additional deadlock conditions caused by branch operations among the active routers. However, it is challenging to implement the throughput-optimized multicast routing in irregular topology-based NoCs because the deadlock conditions become highly complicated, and the Hamiltonian path required to apply the labeling rule may not exist. Two important observations were identified regarding multicast routing in customized NoCs: 1) Even if the NoC lacks a Hamiltonian path, deadlock-freedom can be guaranteed by restricting branch operations to a specific destination. 2) A variable path diversity in a custom topology can be leveraged in routing path allocation and branch. Based on these properties, this study proposes a deadlock-free and throughput-enhanced multicast routing for customized NoC (MRCN). MRCN ensures deadlock freedom by utilizing extended routing and router labeling rules. Furthermore, destination router partitioning and traffic-aware adaptive branching are incorporated to reduce packet routing hops and disperse channel traffic. The effectiveness of MRCN was verified using Noxim, a well-known cycle-accurate NoC simulator, under various topologies and traffic patterns. The simulation revealed that MRCN improved the average latency by 13.98 % and the throughput by 12.16 % under the saturated traffic conditions over the previous multicast routings in customized NoCs.

Index Terms—Customized network-on-chip, multicast routing, irregular topology, path-based routing, packet branching

1 Introduction

Because the computational demands driven by Big Data and artificial intelligence are remarkably increasing, heterogeneous multicores comprised of devices via domain specialization are gaining popularity for higher energy efficiency [1], [2], [3]. The other central axis of the computing platform is the interconnect architecture; thus, a customized network-on-chip (NoC) to provide enriched communication diversity is required to cope with the massive traffic in heterogeneous multicore system-on-chip (SoC) [4], [5].

Intermediate data from handling messages for DNN acceleration [6], [7], [8], [9], configuration [10], synchronization [11], and cache coherence protocols [12], exhibit a single-source multiple-destination (SSMD) transmission pattern in multicore systems. These types of traffic account for more than 30 % of all communication patterns, which has the portion no longer be ignored [10], [11], [12].

Multicast routing, which enables data transmission to a group of destinations, has the potential to amortize SSMD traffic significantly [13], [14]. The branch operation that replicates and delivers packets from the input buffer to multiple output ports inside the router is crucial for multicast routing in NoCs. Multicast packets are merged and distributed at the source router; therefore, the channel load can be diminished.

However, multicast routing suffers from additional deadlock conditions not observed in unicast routing [15], [16]. In particular, the complexity of deadlock analysis considerably increases in customized NoCs caused by asymmetric connectivity. Moreover, finding Hamiltonian paths is becoming highly complicated in irregular topology-based NoCs, because the systematic labeling rule-based routings [13], [14], [15], [16], [17] cannot efficiently be utilized.

Numerous efforts have addressed this problem by providing more hardware resources, such as virtual channel (VC), deeper input buffer, and deadlock recovery logic within the router [18], [19], [20], [21], [22], [23], [24]; however, the throughput improvements were limited since the topological characteristics of customized NoCs were not adequately reflected.

- *Young Sik Lee and Yong Wook Kim are with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea. E-mail: {lty0107, coail}@skku.edu.*
- *Tae Hee Han is with the Department of Semiconductor System Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea. E-mail: than@skku.edu.*

Manuscript received 25 March 2022; revised 12 September 2022; accepted 22 October 2022. Date of publication 26 October 2022; date of current version 16 November 2022.
This work was supported in part by National R&D Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT under Grant 2020M3H2A1076786, in part by the Ministry of Trade, Industry and Energy (MOTIE) under Grant 20011074, and in part by the Institute of Information & communications Technology Planning & Evaluation(IITP) funded by the Korea government (MSIT) under Grant 2019-0-00421 (AI Graduate School Support Program, Sungkyunkwan University).
(Corresponding author: Tae Hee Han.)
Recommended for acceptance by M. Becchi.
Digital Object Identifier no. 10.1109/TPDS.2022.3217296

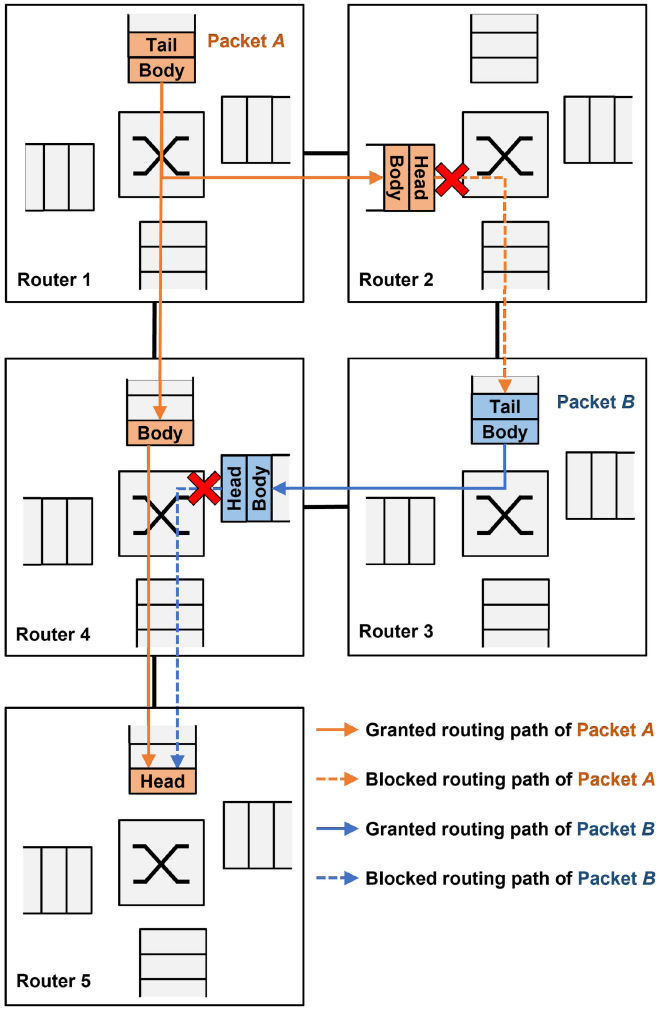


Fig. 1. A possible multicast deadlock scenario in an NoC.

We identified two findings to resolve the multicast problem in customized NoCs: 1) Even if a Hamiltonian path does not exist, restricting branch operations to a specific destination allows the labeling rule compliant routing to operate deadlock-free. 2) A variable path diversity in an irregular topology can be leveraged in routing path allocation and branch.

Inspired by these observations, this study proposes a deadlock-free and throughput-enhanced multicast routing in customized NoC (MRCN) with minimizing additional hardware overhead. MRCN assures deadlock freedom by defining the extended routing rules and labeling routers on the path involving the maximum number of routers. Furthermore, an additional router partitioning method and traffic-aware adaptive branching can reduce packet routing hops and amortize unbalanced network load.

The remainder of this paper is organized as follows. Section 2 introduces multicast-enabled NoC studies related to MRCN and emphasizes the contribution of this study in contrast to them. In Section 3, MRCN is described in the following order: dedicated router architecture, labeled path searching, destination router partitioning, and adaptive branching. Section 4 validates the effectiveness of MRCN through Noxim, a prevalent cycle-accurate NoC simulator, under various traffic patterns. Finally, Section 5 concludes the paper.

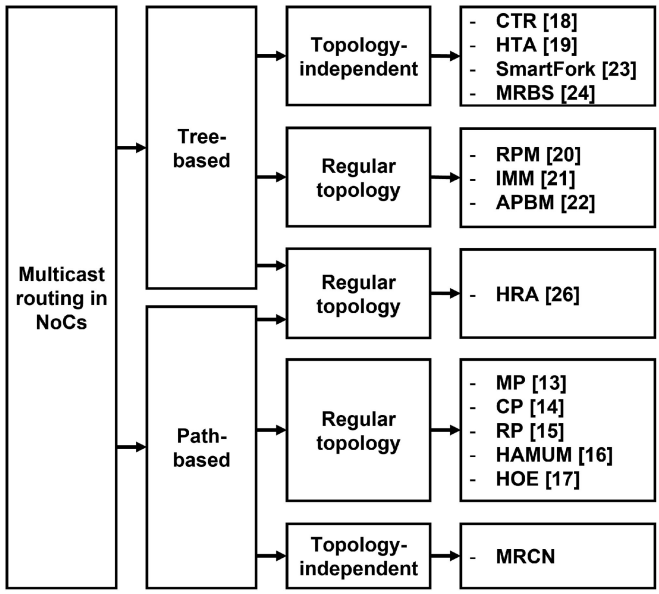


Fig. 2. A classification of multicast routing-related studies on NoCs.

2 Background and Related Work

Branch operation for multicast routing can cause deadlock conditions not observed in unicasting-only cases. [15], [16]. Coffman et al. proved that deadlock occurs if and only if four conditions, namely *mutual exclusion*, *hold-and-wait*, *no preemption*, and *circular wait*, hold simultaneously [25]. These four conditions can be adapted in NoCs as follows:

1) *Mutual exclusion:* Each output port is not shareable.
2) *Hold-and-wait:* A packet holds one output port and waits for another.
3) *No preemption:* Output ports cannot be preempted.
4) *Circular wait:* A circular chain of packets exists.

Because *mutual exclusion* and *no preemption* are common in packet-switched NoCs, deadlock freedom can be guaranteed only by preventing *hold-and-wait* or *circular wait*.

Fig. 1 illustrates an example of multicast deadlock in an NoC. The dashed line indicates a blocked port request, while a solid line indicates a granted and assigned port request. Packet *A* requests a packet transmission to the southern port of router 2, which is currently held by packet *B*. Likewise, packet *B* requests a packet transmission to the southern port of router 4, which is currently held by packet *A*. Each packet holds one output port and waits for another held by the other packet; thus, the packet cannot propagate, resulting in a multicast deadlock. This type of deadlock is caused by inevitable packet branching owing to multicast routing.

Various efforts have been devoted to avoiding multicast deadlock in NoCs, which can broadly be classified into two categories: 1) primarily through router architecture modifications referred to as tree-based routing [18], [19], [20], [21], [22], [23], [24], and 2) a labeling rule-based routing technique referred to as path-based routing [13], [14], [15], [16], [17], as shown in Fig. 2. Both methods can further be classified into either the ones for regular topologies such as mesh and torus or the others on topology-independent cases.

Tree-based approaches provide minimal routings with the source node as the root and all destination nodes as

leaves [24]. The source node establishes a minimal spanning tree and sends a single multicast message across the tree. The packet must be forwarded in all directions simultaneously to duplicate flits at the branching-point router. The contention of multicast packets increases the likelihood of multicast deadlock occurrence, thereby inducing *hold-and-wait*. Tree-based techniques eliminate *hold-and-wait* by deploying additional hardware resources.

Cut-through switched routers [18] avoided *hold-and-wait* by implementing a sufficiently large first-in-first-out (FIFO) buffer to store the largest packet. Hardware tree-based multicast routing algorithm (HTA) [19] implemented a deadlock queue in each router to prevent deadlocks. The recursive partitioning multicast (RPM) [20] and the improved minimal multicast routing (IMM) [21] employed two VCs to transmit up and down streams separately to avoid deadlocks. IMM further built a spanning tree with fully-utilized shared routes to offer a minimal multicast path. Adaptive partition-based multicast (APBM) [22] adopted more VCs and dedicated turn models, north-last and west-last. SmartFork [23] partitioned the output ports of the router into several groups. Intra-group ports were serviced serially, while inter-group ports were serviced in parallel. The VCs in SmartFork were embraced to target any desired point on the multicast performance spectrum. The multicast router using buffer sharing (MRBS) [24] achieved deadlock-free minimal path routing with low area overhead by exploiting the spatial diversity of the input buffer. MRBS managed all input buffers as a register file style to efficiently utilize the buffer space instead of dedicating a single buffer per input port.

Path-based routing assigns consecutive labels to all routers, allowing the packets to be routed in ascending or descending order depending on the labels. Because of the simplicity in avoiding *hold-and-wait*, label-based routing has been widely adopted in mesh and torus topology, where ordered labeling is possible.

Dual-path routing (DP) [13] classified destination routers according to their label significance than the source router: higher labels routed ascending and lower labels routed descending orders, respectively. Although DP achieved deadlock-free multicast routing by avoiding *circular wait*, it might cause the worst-case critical path visiting all routers.

The label-based routing rule applied to the grouped destinations with a proper partitioning satisfied deadlock-freedom in every partitioned group while avoiding the worst-case routing path passing through all routers [14], [15]. Accordingly, several path-based approaches concerning destination partitioning have been proposed.

The multi-path (MP) algorithm, also presented in [3], divided nodes into four disjoint rectangular subsets depending on the location of the source node. By allowing only routing paths within each subset, MP achieved a lower total hop count than DP, whereas critical paths were not reduced significantly due to unbalanced subset size. The column-path (CP) algorithm [14] partitioned nodes into up to $2\times k$ subsets, where k is the number of columns in the mesh. CP generated more balanced subsets and shorter routing paths than MP with a high degree of parallelism. The recursive partitioning (RP) [15] method balanced the number of nodes included in each subset to minimize hop counts. However, both CP and RP could cause congestion in some clusters of nodes because of ignoring the locality of traffic loads.

The Hamiltonian adaptive multicast unicast model (HAMUM) [16] permitted more routing paths by using adaptive MP (AMP) and adaptive CP (ACP) techniques, which resulted in less congested paths and thus enhanced throughput compared with MP and CP. The Hamiltonian-based odd-even (HOE) turn model [17] maximized the degree of adaptiveness and maintained deadlock freedom without additional VCs. The hybrid routing algorithm (HRA) [26] attempted to combine the benefits of both path- and tree-based routings by adaptively determining the branching policy according to the buffer usage.

Whereas HRA and the path-based approaches incorporated the regularity of NoC topologies to reduce the routing hop counts and alleviate congestions, they could not be directly applied in customized NoCs owing to the increased complexity in finding the Hamiltonian path required by label-based routing rules. Although previous topology-independent tree-based methods could be easily adapted to customized NoCs, the vulnerability to contention still remained owing to the ignorance of topological characteristics. MRCN enables path-based routing in customized NoCs through extended routing rules and reduces network congestion by embracing path diversity for the given topology.

3 MRCN

The primary goal of MRCN is to ensure deadlock-free and throughput-enhanced multicast routing in customized NoCs. In this regard, we create the labeled path searching, destination router partitioning, and adaptive branching algorithms to guarantee deadlock freedom and boost performance. Fig. 3 presents the schematic diagram of each process of MRCN and the corresponding improvements. The definitions of the topology graph and the notations utilized in MRCN are described in Table 1 to help explain each procedure.

3.1 MRCN Router Architecture

As shown in Fig. 4, the MRCN routers operate with a five-stage pipeline: buffer write (BW), route computation (RC), switch allocation (SA), switch traversal (ST), and link traversal (LT), which are successively run on the input buffer, RC logic, switch allocator, crossbar switch, and links, respectively. The switch allocator and crossbar switch are modified to support the adaptive branching, and the extended RC logic allows for sophisticated route computations in MRCN.

The input buffer stores the incoming packet in the BW stage and delivers the destination list extracted from the head flit to the RC logic. Owing to path-based routing, MRCN routers consume smaller input buffers than tree-based approaches that employ VC schemes and cut-through switching.

In the RC stage, the output ports of the packet and the destination set in each direction are determined based on the adaptive branching algorithm. The RC logic acquires

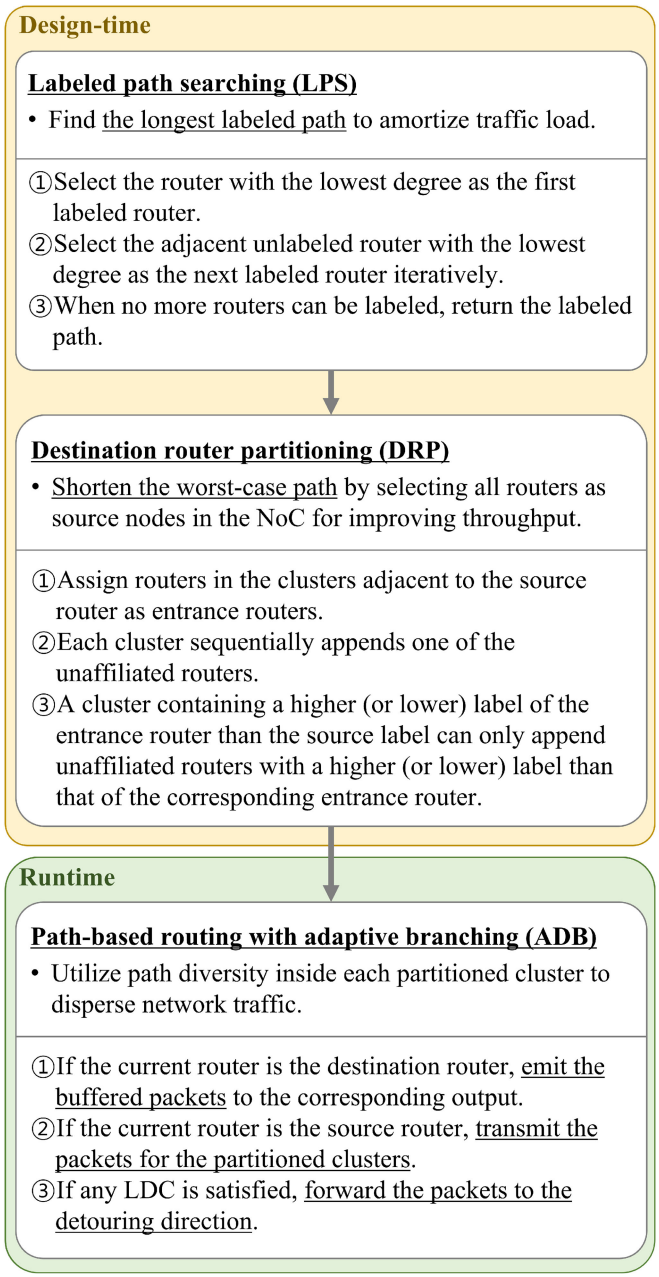


Fig. 3. Overall flow of MRCN.

the credit signal from the adjacent routers to reflect the traffic condition and decides whether to detour the labeled path depending on the credit conditions of adjacent routers.

In the SA stage, the switch allocator manages the packet requests to the desired output ports through the round-robin policy and continues until the request for the head flit to the desired output port is accepted. The arbitration function is extended to support parallel packet propagation to multiple output ports, required for adaptive branching.

After the SA, the granted packets enter the ST stage and are routed to the corresponding output ports via the crossbar switch. The input buffer invalidates the processed flit and sends a credit signal to the upstream router to indicate the availability. Next, the packet passing through the crossbar switch is traversed to the input port of the downstream router in the LT stage.

3.2 Labeled Path Searching

Labeled path-based routing is widely adopted for regular topology-based NoCs having Hamiltonian path owing to the benefit of deadlock-freedom without demanding additional hardware resources. For instance, in the 4×4 mesh depicted in Fig. 5a, the Hamiltonian path of seamlessly connected red arrows can be revealed, avoiding *circular wait*. Nonetheless, a customized NoC depicted in Fig. 5b, which has no Hamiltonian path, is incompatible with existing labeled path-based routings.

Algorithm 1. Labeled Path Searching Algorithm $LPS(T(U, L), \mathcal{R})$

Input: Topology graph $T(U, L)$
Labeled path list $\mathcal{R}$
Output: Updated labeled path list $\mathcal{R}'$

if $(N(\mathcal{R}) < 1)$ **then**
  initialize $\mathcal{R}$ as an empty list
  for $i = 1$ to $N(U)$ **do**
    find $u_i \in U$ such that $L(u_i)$ is minimal
    add u_i to U_C
  end for
  for $h = 0$, $N(U_C) < 2$, $h++$ **do**
    find $u_i \in U_C$ such that $\sum_{\forall u_j \in H(u_i,h)} N(L(u_j))$ is minimal
  end for
  label u_i as R_1
  add R_1 to $\mathcal{R}$
  $\mathcal{R}' = LPS(T(U, L),\mathcal{R})$
  return $\mathcal{R}'$
else
  for $j = 1$ to $N(U)$ **do**
    find $u_j \in U$ such that u_j is unlabeled && $l_{ij} \in L$ && $L_U(u_j)$ is minimal
    add u_j to U_C
  end for
  for $h = 0$, $N(U_C) < 2$, $h++$ **do**
    find $u_j \in U_C$ such that $\sum_{\forall u_k \in H(u_j,h)} N(L(u_k))$ is minimal
  end for
  if $N(U_C) == 0$ **then**
    return $\mathcal{R}$
  else
    label u_j as $R_{N(\mathcal{R})+1}$
    add $R_{N(\mathcal{R})+1}$ to $\mathcal{R}$
    $\mathcal{R}' = LPS(T(U, L),\mathcal{R})$
    return $\mathcal{R}'$
  end if
end if

Deadlock occurs if and only if all Coffman conditions are held concurrently; therefore, deadlock freedom is achieved if at least one of these conditions is permanently removed. *Circular wait* for packets passing through unlabeled routers can be prevented by supplementing the following two routing rules: 1) A packet destined for an unlabeled router is imposed as a unicast packet, and 2) A packet from the unlabeled router is transferred to the nearest labeled router with legacy path-based routing.

TABLE 1
Notations Used in MRCN

Notation	Description
$T(U, L)$	Topology graph where each vertex $u_i \in U$ denotes a router in the topology and each $l_{i,j} \in L$ denotes a link between u_i and u_j.
$\mathcal{R}$	Set of labeled router $R_1, R_2, \ldots, R_{N(\mathcal{R})}$
s	Label of the source router where packet is injected.
c	Label of the current router where packet is stored.
N_P	The number of router clusters in the destination router partitioning.
F	Set of unaffiliated routers in the destination router partitioning.
$L_U(u_i)$	The degree of a router which represents the number of links connected to the router u_i.
$L_R(R_i, R_j)$	The number of links between a router R_i and another router R_j.
$L_P(R_i, P_j)$	The number of links between a router R_i and a router cluster P_j.
N_{port}	The number of output ports in the router.
U_C	Set of first router candidates to be labeled.
$H(u_i, h)$	Set of routers h hop away from u_i.
D	Destinations of a packet.
O	Set of output ports in the current router $\{o_1, o_2, \ldots, o_{N_{port}}\}$.
B	Set of available depth of input buffer in downstream router to output ports $\{b_1, b_2, \ldots, b_{N_{port}}\}$.
DIR	Set of output directions for the packet in the adaptive branching $\{dir_1, dir_2, \ldots, dir_{N_{port}}\}$.
$\mathcal{D}$	Set of destinations for each output direction $\{D_1, D_2, \ldots, D_{N_{port}}\}$.
$R(o_i)$	The downstream router in the direction of output port o_i in current router.
$O_D(R_i, R_j)$	The output port in R_i connected to R_j
S_P	The number of flits in a single packet.
B_M	Maximum input buffer depth

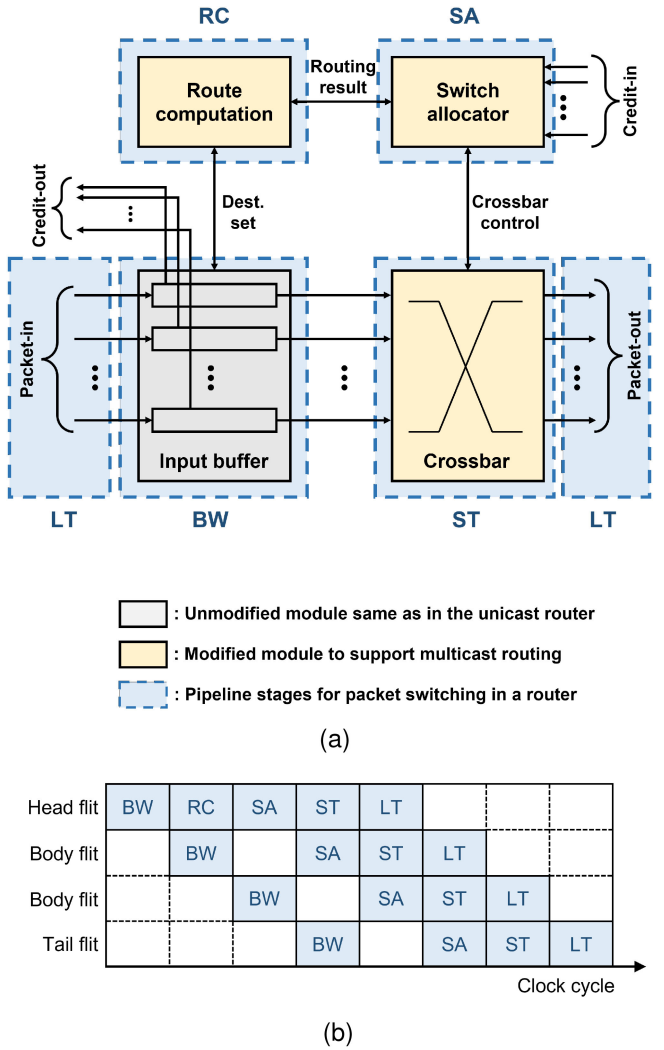


Fig. 4. Router architecture of MRCN. (a) Router structure and scope of the pipeline stage. (b) Packet propagation process consisting of four flits in non-blocking condition.

The extended routing rule allows unlabeled routers, and thus legacy path-based routing can be utilized by restricting packet branching. Extra traffic caused by inevitable unicast packets can be relieved by maximizing the labeled routers. Fig. 5c depicts one of the labeled paths with the maximum number of labeled routers in the customized NoC of Fig. 5b. In this case, the packets destined to only unlabeled router R_{u1} are transmitted in a unicast manner. Accordingly, the proposed labeled path searching (LPS) algorithm aims to find a path that includes as many routers as possible to minimize unlabeled routers, as illustrated in Fig. 5c.

The objective of the LPS algorithm is to label as many unlabeled routers as possible, which can be accomplished by maximizing link complexity among unlabeled routers. Generally, as the link complexity of the topology increases, the number of path labeling cases to be explored and the probability of determining a labeled path with the highest hop count also increases. When the router with the lowest degree is labeled preferentially, the link complexity of the remaining unlabeled routers in the sub-topology is maximized. Therefore, LPS attempts to maximize the labeled routers by labeling the routers with the lowest degree preference.

Algorithm 1 describes the LPS method for the maximal labeled path. First, the router with the lowest degree is selected as the first router R_1 of the labeled path (lines 1-7). Next, the router with the lowest degree among the unlabeled routers adjacent to R_1 is chosen as the next labeled router (lines 11-20). The router labeling process is recursive (lines 8-9), and by labeling the adjacent routers with the lowest degree first, the connection to the remaining unlabeled routers is maximized (line 12). If the labeled path encompasses all routers, it implies the existence of a Hamiltonian path, and thus no unicast packet to the unlabeled routers is generated.

Lines 5-9 describe the selection criteria for multiple routers having the same degree. In cases where several routers share the lowest degree, the one with the fewest total degree of directly connected routers is chosen. If multiple such routers exist, the one with the fewest total degree of routers at two hops is selected as the first router to be labeled. Similarly, if there is still more than one of a such the router with a minimum sum of degrees of routers at two hops, the first router selection criteria are extended by increasing the hop count by one. If multiple routers with

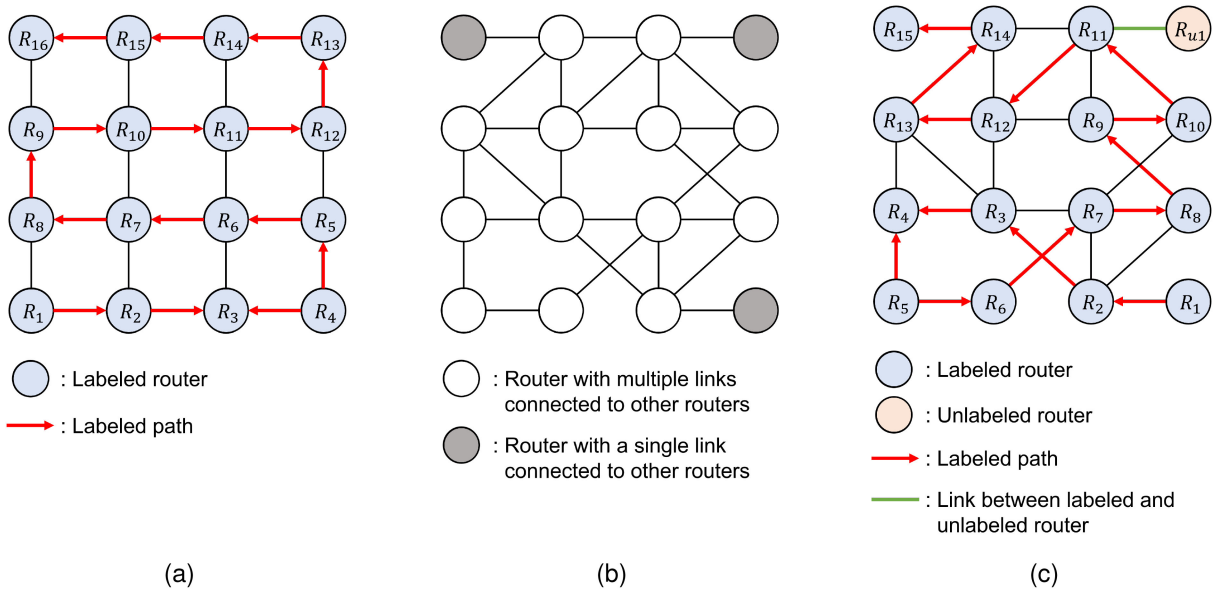


Fig. 5. Optimized labeled paths in mesh and customized NoCs. (a) Example of a Hamiltonian path with conventional labeling rule applied in 4×4 mesh. (b) Customized NoC case without Hamiltonian paths. (c) Example of a labeled path including the maximum number of routers in the NoC of Fig. 5(b).

the minimum degree sum remain after the searching algorithm reaches the maximum hop count, all these routers have the highest priority, and the first labeled router is arbitrarily selected among them

In the example of Fig. 5c, R_1, R_{15}, and R_{u1} have a minimum degree of one. The sum of degrees of routers one hop away from R_1, R_{15}, and R_{u1} are four, four, and five, respectively; therefore, the first router to be labeled is selected between R_1 and R_{15}. The total degree of routers two hops away from R_1 and R_{15} are 13 and 14, respectively, and R_1 is eventually selected as the first router of the labeled path.

In the LPS of MRCN, both ascending and descending are allowed in the path labeling order. Nevertheless, labeling in ascending order is more straightforward than in descending order due to the data structure utilized by LPS. Since the number of labeled routers was not determined prior to path labeling, LPS declares labeled routers by adopting the stack-based dynamic memory allocation. The elements of the stack structure and the push operation indicate labeled router information and path labeling, respectively. By reflecting that the top index increases as elements are pushed into the stack structure, LPS labels the routers in ascending order, where the stack indices match the labels of the routers.

While LPS ensures deadlock freedom, it must also overcome the additional challenges of preventing the critical path that passes through all routers and minimizing unicast packets. Therefore, destination router partitioning and adaptive branching are introduced to resolve these two problems.

3.3 Destination Router Partitioning

Destination router partitioning (DRP) aims to remove the worst-case path visiting all routers in customized NoCs, thereby improving overall throughput. As mentioned in Section 2, appropriate router clustering can shrink the critical path of the entire NoC to the critical path of the largest cluster [14], [15]. To reduce the size of the maximal cluster that determines the longest labeled path, DRP attempts to balance the cluster sizes. The number of clusters equals the number of routers adjacent to a given source router.

Algorithm 2. Destination Router Partitioning ($T(U, L)$, $\mathcal{R}$, s)

Input: Topology graph $T(U, L)$
Labeled path list $\mathcal{R}$
Label of the source router s
Output: Partitioned router cluster $\mathcal{P} = \{P_1, P_2, \ldots, P_{N_p}\}$

$j = 0$
add all routers except R_s into the unaffiliated router set F
for $i = 1$ to $N(\mathcal{R})$ **do**
  if $L_R(R_s, R_i) == 1$ **then**
    add R_i to P_j
    $j = j + 1$
  end if
end for
$N_P = j - 1$
while all routers except R_s are partitioned **do**
  $j = (j + 1)\%N(N_P + 1)$
  for $i = 1$ to $N(\mathcal{R})$ **do**
    find $R_i \in F$ such that $\sum_{0<j<N_P} L_P(P_j, R_i)$ is maximal
  end for
  add R_i to P_j
  exclude R_i from F
end while

Algorithm 2 describes the DRP method in detail. A router in each cluster adjacent to a given source router is uniquely assigned as an entrance router. Each cluster takes turns appending one of the unaffiliated routers to reduce the variance in the number of routers between clusters. The newly appended router in each cluster must keep up with the labeling order (either ascending or descending)

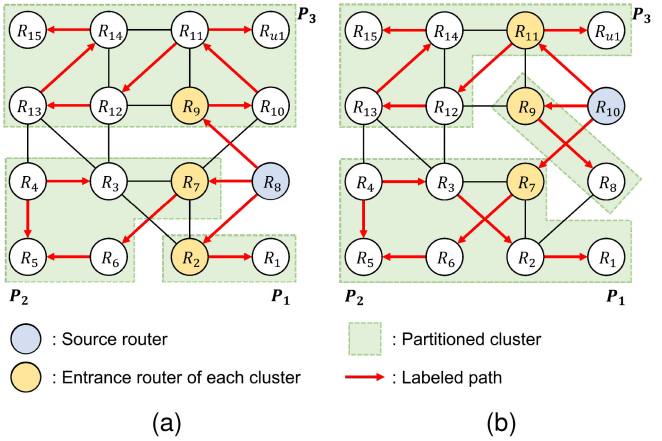


Fig. 6. Exemplary router partitioning results in the customized NoC of Fig. 5(c) where (a) R_8 is the source router and (b) R_{10} is the source router.

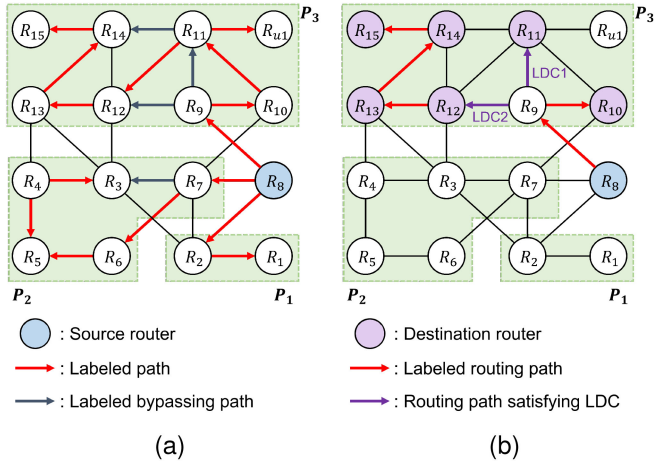


Fig. 7. (a) Label-detouring path in ADB-enabled NoC for Fig. 6a where R_8 is the source router. (b) ADB example of a packet whose source is R_8 and destinations are R_{10}, R_{11}, R_{12}, R_{13}, R_{14}, and R_{15}.

concerning the entrance router. A cluster containing an entrance router with a label higher (or lower) than the source can only contain unaffiliated routers with a label higher (or lower) than the source router. An unlabeled entrance router can include only adjacent unlabeled routers in a cluster.

To ensure maximum uniformity in the number of routers between clusters, it is desirable to affiliate the remaining routers in ascending order of the number of neighboring clusters. Therefore, each cluster prioritizes unaffiliated routers with the fewest connections to other clusters.

For routing all source and destination pairs, DRP is performed deterministically by sequentially selecting all routers as source nodes. Fig. 6 illustrates an example of DRP with R_8 and R_{10} as source routers, respectively. In Fig. 6a, R_2, R_7, and R_9 adjacent to R_8 are designated as entrance routers of clusters P_1, P_2, and P_3, respectively. Using the labeling rules, we place R_1, R_6, and R_{11} into clusters P_1, P_2, and P_3, respectively, based on their connectivity to other clusters. Since P_1 is entirely devoid of mergeable routers, the remaining unconnected routers must exist in either P_2 or P_3. Finally, the resulting clusters comprise $P_1 = \{R_1, R_2\}$, $P_2 = \{R_3, R_4, R_5, R_6, R_7\}$ and $P_3 = \{R_9, R_{10}, R_{11}, R_{12}, R_{13}, R_{14}, R_{15}, R_{u1}\}$.

In Fig. 6b, R_7, R_9, and R_{11} are designated entrance routers for clusters P_1, P_2, and P_3 adjacent to R_8. Using the labeling rules, we place R_6, R_8, and R_{12} into clusters P_1, P_2, and P_3, respectively, based on their connectivity to other clusters. Since P_1 is entirely devoid of mergeable routers, the remaining unconnected routers must exist in either P_2 or P_3. Finally, the resulting clusters comprise $P_1 = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$, $P_2 = \{R_8, R_9\}$ and $P_3 = \{R_{11}, R_{12}, R_{13}, R_{14}, R_{15}, R_{u1}\}$.

3.4 Adaptive Branching

Adaptive branching (ADB) utilizes path diversity inside each partitioned cluster to disperse network traffic. Alternative paths can be created by excluding labeled paths in the cluster. However, since the alternative paths are not labeled regularly, the Coffman condition cannot always be avoided. We define label-detouring conditions (LDCs) that prevent both *circular wait* and *hold-and-wait*, thus allowing alternative path routing only in these cases.

Label-Detouring Condition 1 (LDC 1)

- The packet is destined to the downstream router in the detouring direction.
- The input buffer in the downstream router in the detouring direction is empty.

Label-Detouring Condition 2 (LDC 2)

- There is a destination whose label is either higher or lower than the current router and the downstream router in the detouring direction.
- The entire packet can be stored in the available input buffer space on the downstream router in the detouring direction.

Satisfying LDCs 1 and 2 avoids *circular wait* and *hold-and-wait*, respectively, thus allowing detouring without deadlock. LDC 1 was intended to enable packets with destinations to be routed to downstream routers. Because branching packets are ejected directly from the destination router, they never experience *circular wait*. When there is enough input buffer space to hold the entire packet in the downstream router, LDC 2 was created to support branching. *Hold-and-wait* is not activated for packets that have been completely stored in the downstream router. In addition, the resulting alternative path can reduce packet latency because it has fewer hops than the labeled path.

If there exists any destination with a label valued between the current router and the downstream router of the detouring direction, the packet should also be propagated through the labeled path to avoid *circular wait*. Accordingly, the packet is branched to a labeled path and a label-detouring path, thus reducing total hop counts.

Fig. 7a depicts a label-detouring path in ADB-enabled NoC of Fig. 6, where R_8 is the source router. If either LDCs are satisfied, the packets can be routed via the label-detouring path. For example, for a packet whose source is R_8 and destinations are R_{10}, R_{11}, R_{12}, R_{13}, R_{14}, and R_{15} as shown in Fig. 7b, if R_9 satisfies LDC 1 and LDC 2, the packet is routed to R_{11} and R_{12} through label-detouring paths.

Algorithm 3. Adaptive Branching Algorithm $ADB(s, c, D, O, B, \mathcal{P}, S_P)$

Input: Label of source router s
Label of current router c
Destination list D
Output ports $O = \{o_1, o_2, \ldots, o_{N_{port}}\}$
Available downstream buffer depth $B = \{b_1, b_2, \ldots, b_{N_{port}}\}$
Partitioned router clusters $\mathcal{P}$
Packet size S_P
Output: Set of destinations for each output direction $\mathcal{D}$
Output direction of the packet $DIR = \{dir_1, dir_2, \ldots, dir_{N_{port}}\}$

1: **if** $R_C \in D$ **then**
2: request the destined ejection port of the packet
3: $D = D - R_C$
4: **end if**
5: **if** $S == C$ **then**
6: **for** $i = 1$ to N_{port} **do**
7: $D_i = D \cap P_i$
8: **if** $D_i \neq \phi$ **then**
9: $dir_i = 1$
10: **end if**
11: **end for**
12: **else if** $S < C$ **then**
13: **for** $i = 1$ to N_{port} **do**
14: **if** $R(o_i)! = R_{C+1} \&\& R(o_i) \in D \&\& b_i == B_M$ **then**
15: $dir_i = 1$
16: $D_i = D_i \cup R(o_i)$
17: $D = D - R(o_i)$
18: **end if**
19: **end for**
20: **for** $i = 1$ to N_{port} **do**
21: **for** $j = 1$ to $N(D)$ **do**
22: **if** $R(o_i)! = R_{C+1} \&\& S_P < b_i \&\& d_j > R(o_i)$ **then**
23: $dir_i = 1$
24: $D_i = D_i \cup R(o_i)$
25: $D = D - R(o_i)$
26: **end if**
27: **end for**
28: **end for**
29: **if** $D! = \phi$ **then**
30: $dir_{O_D(R_C, R_{C+1})} = 1$
31: $D_{O_D(R_C, R_{C+1})} = D$
32: **end if**
33: **else**
34: - similar to lines 12-32
35: **end if**

Algorithm 3 describes the route computation with ADB for multicast packets in the router. The packet is ejected to the corresponding output port if the current router is included in the packet destinations (lines 1-4). If the current router is the source router, packets are transmitted for each cluster partitioned by the DRP algorithm (lines 5-11). If LDCs with the detouring direction output ports are satisfied, the router forwards the packet to the corresponding ports (lines 12-48).

4 Evaluation

4.1 Simulation Setup

We evaluated the effectiveness of MRCN in terms of the average latency, throughput, and area overhead in various topologies. To this end, routers and network interfaces for HTA [19], CTR [18], ACP [16], HOECP [17], HRA [26], SmartFork [23], MRBS [24], and MRCN have been implemented in a SystemC [27] environment. Noxim [28], a well-known cycle-accurate NoC simulator, was utilized to generate and simulate NoCs with mesh and irregular topologies.

TABLE 2
Configuration of NoC and Injected Packets

NoC model	Mesh: 4×4 – 9×9 Custom: 120 distinct models with 16 – 81 nodes
Router degree	Mesh: 5 Custom : 3 – 6
Flit width	130 bits (2-bit flit type + 128-bit message data)
Packet length	Min. : 2 flits (1 head flit + 1 tail flit) Max.: 10 flits (1 head flit + 8 body flits + 1 tail flit)
Operation frequency	0.5 GHz
Traffic model	Uniform random, CTG-based, Rent's rule, Scenario-based

The detailed NoC configurations are presented in Table 2. All implemented routing approaches were simulated on an 8×8 structured mesh with 64 nodes. The prevalent Hamiltonian path of DP [13] was employed in the mesh without unlabeled router, and the LPS algorithm in Section 3.2 was applied only in customized NoC.

The connectivity of the customized NoC was generated through a task graph for free (TGFF) [29], which is commonly used to build a heterogeneous multicore communication task graph and a fault-tolerant topology generation (FTTG) [30] which optimizes the resulting irregular topology. As a result, 20 distinct customized NoC models with 16, 25, 36, 49, 64, and 81 communication nodes were constructed. Furthermore, the generated custom topology models were classified based on whether the average router degree was greater than five or not. Then, we conducted correlation analysis for each network size with the corresponding mesh topology model (average degree of five). Because ACP, HOECP, and HRA were dedicated to mesh-based NoCs, they were excluded in the customized NoC simulations.

The stimulus was generated using uniform random, CTG-based, and Rent's rule [31] patterns to reflect various traffic conditions. Uniform random traffic arbitrarily produces unicast and multicast packets depending on the injection ratio for all destinations. CTG-based traffic injects packets according to the traffic rate between each communication node in the task graphs generated by FTTG [30]. Rent's rule traffic is inspired by "communication nodes are mapped in a direction to reduce hop count [31]", creating a realistic pattern of traffic that enhances the traffic between nearby nodes.

A bimodal distribution of packets was assumed, with 70 % of the packets being minimal and the 10 % being maximal to reflect the realistic packet switching pattern of the

TABLE 3
Comparison of the Saturation Points for Simulation Traffic Models in the Mesh According to the Network Size

Uniform random	Saturation point (mPKTS/Cycle/Node)					
	4×4	5×5	6×6	7×7	8×8	9×9
HTA	16.55	22.03	27.02	30.46	32.73	38.85
MRBS	21.46	24.65	29.74	34.52	38.88	44.16
CTR	20.35	26.85	30.93	36.16	39.35	44.74
SmartFork	26.93	33.02	38.38	43.08	48.55	57.22
ACP	29.63	35.68	41.79	51.46	53.68	64.28
HOECP	32.66	39.97	47.95	52.34	55.43	69.83
HRA	34.53	42.55	48.72	58.33	64.12	72.98
MRCN	33.95	41.12	49.71	56.59	63.28	73.32
CTG -based	Saturation point (mPKTS/Cycle/Node)					
	4×4	5×5	6×6	7×7	8×8	9×9
HTA	14.09	18.75	24.59	26.30	30.84	34.32
MRBS	17.64	23.14	26.70	30.54	36.11	39.98
CTR	19.02	22.70	28.61	33.27	36.28	41.95
SmartFork	22.83	31.03	34.90	38.11	44.46	54.26
ACP	27.95	31.79	39.42	46.62	52.14	58.67
HOECP	28.65	35.60	43.05	50.54	56.14	64.35
HRA	29.95	38.84	45.58	53.15	59.11	67.14
MRCN	29.73	37.26	47.03	53.39	58.84	68.09
Rent's rule	Saturation point (mPKTS/Cycle/Node)					
	4×4	5×5	6×6	7×7	8×8	9×9
HTA	12.76	18.74	21.79	25.66	28.78	33.98
MRBS	16.78	21.96	26.18	28.26	34.34	39.62
CTR	15.69	23.20	26.89	30.72	34.30	38.39
SmartFork	21.43	28.21	32.68	38.66	42.83	48.97
ACP	26.21	30.01	35.24	42.71	47.70	56.96
HOECP	27.01	33.11	42.08	46.31	53.81	61.86
HRA	30.48	36.86	41.35	49.70	56.81	65.43
MRCN	28.68	36.67	41.87	48.52	55.80	64.17
SMT Scenario	Saturation point (mPKTS/Cycle/Node)					
	4×4	5×5	6×6	7×7	8×8	9×9
HTA	14.19	19.24	22.77	25.15	29.15	32.46
MRBS	17.60	19.70	25.41	29.07	34.08	38.20
CTR	16.08	23.70	27.45	32.33	33.53	40.07
SmartFork	22.55	28.46	34.30	38.67	43.14	48.71
ACP	25.59	31.39	36.45	44.88	47.22	56.71
HOECP	27.56	34.62	41.76	47.24	53.16	62.43
HRA	28.32	38.19	42.05	49.53	55.98	64.38
MRCN	28.16	34.96	43.12	49.87	54.51	64.15
DNN acc. Scenario	Saturation point (mPKTS/Cycle/Node)					
	4×4	5×5	6×6	7×7	8×8	9×9
HTA	11.37	16.87	22.96	24.58	25.28	31.39
MRBS	16.80	17.96	21.93	27.47	28.98	35.08
CTR	14.23	18.51	21.90	28.32	29.52	37.06
SmartFork	18.19	26.45	31.07	32.21	37.87	47.45
ACP	20.35	28.95	31.10	38.34	47.12	52.76
HOECP	23.99	30.89	39.10	45.37	50.50	57.87
HRA	28.53	35.99	39.59	47.74	49.77	62.00
MRCN	24.71	31.26	40.06	46.70	53.27	60.68

heterogeneous multicore platform, in accordance with [32]. In addition, guided by prior research [33] that reported real-world multicast percentages, we adopted two different scenarios: 5 % multicast traffic (low multicast intensity) and 30 % multicast traffic (high multicast intensity). In DNN acceleration operation, a scenario based on the traffic model of DNN+NeuroSim [34] was utilized in the evaluation. DNN+NeuroSim is a benchmark for training and inference behavior for the CIFAR-10 and ImageNet datasets, generating massive SSMD traffic from a single memory node to multiple computing units. Simultaneous multi-threading (SMT) scenario of the Gem5 [35] was used to generate traffic benchmarks, including configuration, synchronization, and cache coherence operations.

All router models under consideration were implemented using industry-standard Verilog HDL [36] and DesignCompiler [37] with the selected area electron diffraction (SAED) 32 nm standard cell library [38] to assess the area overhead.

4.2 Simulation Result

4.2.1 Average Latency

Fig. 8 depicts the average latency according to the injection rate in the 8×8 mesh. The saturation point is the packet

TABLE 4
Comparison of the Saturation Points in Customized NoCs With Average Degree Lower Than Five

Uniform random	Saturation point (mPKTS/Cycle/Node)					
Node size	16	25	36	49	64	81
HTA	16.96	20.45	22.92	27.15	29.36	34.22
MRBS	17.48	22.20	25.72	32.45	34.28	41.37
CTR	19.66	24.69	27.48	32.62	34.84	41.73
SmartFork	23.96	27.05	34.94	38.41	42.80	50.70
MRCN	30.66	36.32	43.59	51.63	53.24	63.63
CTG -based	Saturation point (mPKTS/Cycle/Node)					
Node size	16	25	36	49	64	81
HTA	13.48	16.45	19.55	22.95	26.10	32.00
MRBS	16.07	20.17	23.49	30.77	32.29	37.60
CTR	17.98	23.28	23.19	30.79	30.15	38.31
SmartFork	21.26	22.89	32.59	36.05	38.15	45.91
MRCN	27.07	33.87	40.77	48.67	53.15	59.11
Rent's rule	Saturation point (mPKTS/Cycle/Node)					
Node size	16	25	36	49	64	81
HTA	13.09	18.17	19.28	21.98	23.50	29.68
MRBS	14.43	17.19	21.29	27.87	30.44	34.47
CTR	15.40	20.93	24.60	28.10	29.47	36.26
SmartFork	20.97	23.31	28.90	31.83	35.59	43.52
MRCN	26.39	30.91	38.28	45.17	48.33	57.17
SMT Scenario	Saturation point (mPKTS/Cycle/Node)					
Node size	16	25	36	49	64	81
HTA	13.29	15.44	18.53	23.18	23.97	29.97
MRBS	13.51	18.40	21.69	27.37	30.81	35.96
CTR	16.55	20.46	23.17	26.87	30.72	36.02
SmartFork	21.47	22.81	28.75	32.32	37.76	43.78
MRCN	25.97	32.69	36.98	44.30	49.98	55.62
DNN acc. Scenario	Saturation point (mPKTS/Cycle/Node)					
Node size	16	25	36	49	64	81
HTA	9.87	12.51	17.07	19.46	22.35	27.40
MRBS	10.54	16.37	21.22	23.56	25.59	33.25
CTR	13.39	18.70	20.88	24.83	27.12	31.01
SmartFork	18.74	20.17	29.16	31.81	32.97	40.59
MRCN	21.76	28.47	36.64	43.15	45.00	50.43

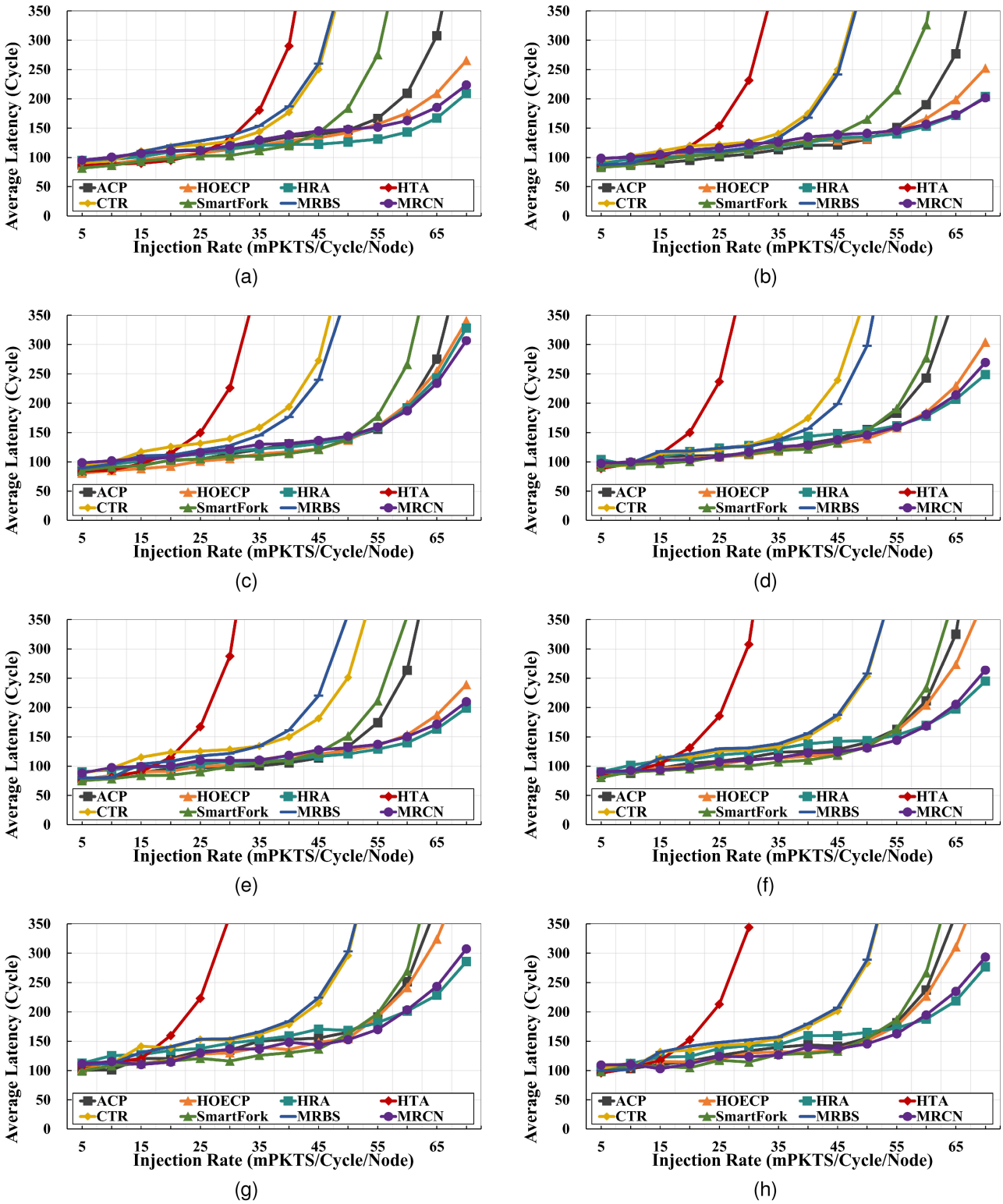


Fig. 8. Average latency simulation results in the 8×8 mesh under (a) Uniform random traffic with 5 % multicast intensity, (b) Uniform random traffic with 30 % multicast intensity, (c) CTG-based traffic with 5 % multicast intensity, (d) CTG-based traffic with 30 % multicast intensity, (e) Rent's rule traffic with 5 % multicast intensity, (f) Rent's rule traffic with 30 % multicast intensity, (g) DNN acceleration scenario, and (h) SMT scenario.

injection rate at which the latency is twice that at the zero-load [3]. As the injection rate increased, it was saturated in the order of HTA, MRBS, CTR, SmartFork, ACP, HOECP, MRCN, and HRA. Adaptive routing in MRCN reduced contentions by allocating alternative paths according to the traffic conditions, thus resulting in a 23.24 % greater saturation point than tree-based approaches. In addition, MRCN achieved a saturation point of 12.41 % greater than the other path-based approaches, which focus on the mesh topologies by allowing to avoid contentions efficiently. As an exception, the saturation point of MRCN was lower than that of the mesh-optimized HRA but showed a similar difference.

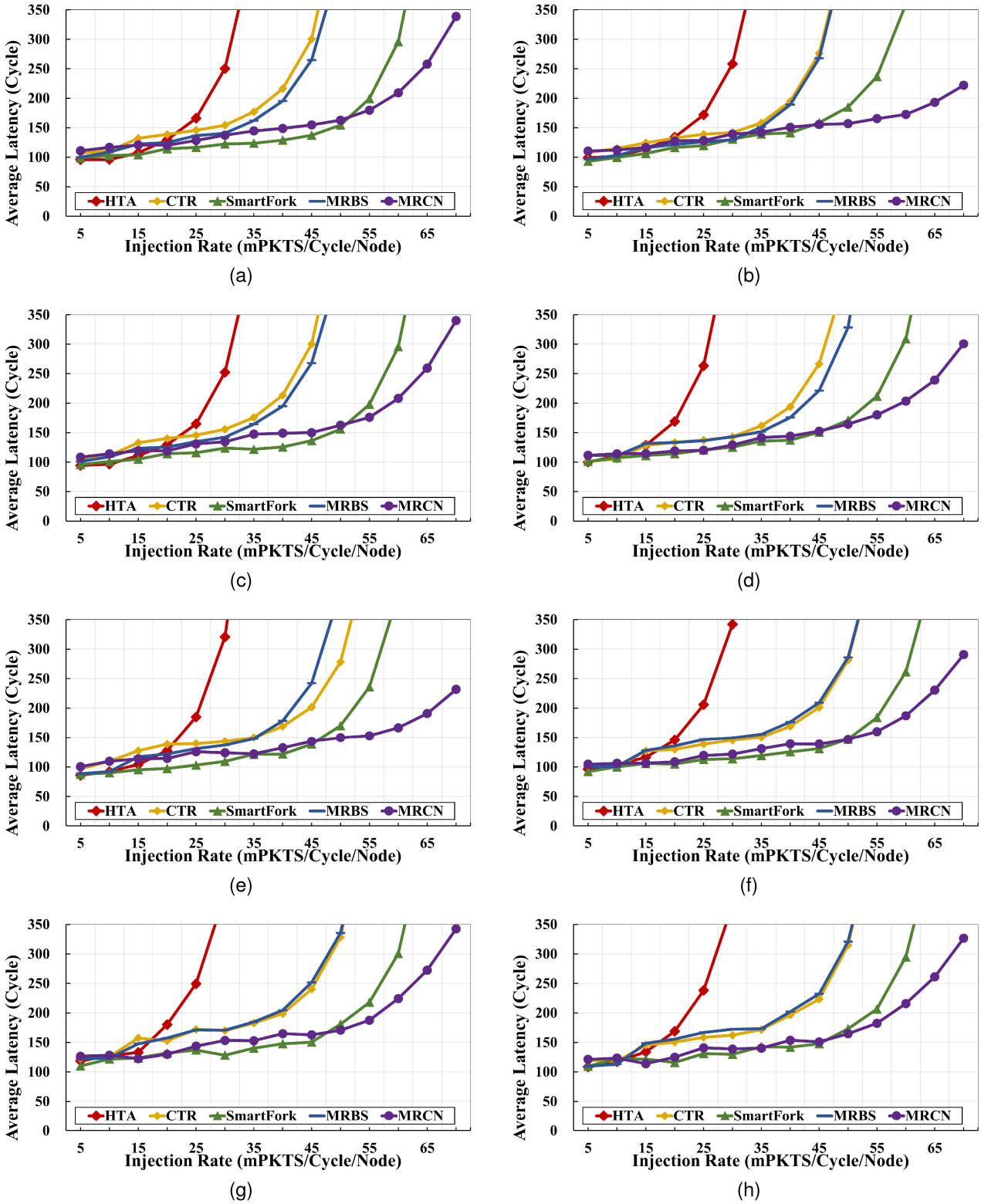


Fig. 9. Average latency simulation results in the 64-node customized NoCs with a degree lower than five under (a) Uniform random traffic with 5 % multicast intensity, (b) Uniform random traffic with 30 % multicast intensity, (c) CTG-based traffic with 5 % multicast intensity, (d) CTG-based traffic with 30 % multicast intensity, (e) Rent's rule traffic with 5 % multicast intensity, (f) Rent's rule traffic with 30 % multicast intensity, (g) DNN acceleration scenario, and (h) SMT scenario.

HRA gives slightly better saturation point compared to proposed MRCN in an 8×8 mesh-based NoCs for every injection rate. MRCN is a scalable multicast routing algorithm that can be applied to both regular and customized topologies, whereas HRA is optimized for mesh topology. As a result, in the 8×8 mesh network mentioned by the reviewer, HRA is likely to provide superior throughput. The hybrid path balancing method (HPBM) in HRA optimizes the path of branched packets by calculating the total transmission hop count from the branching point to the

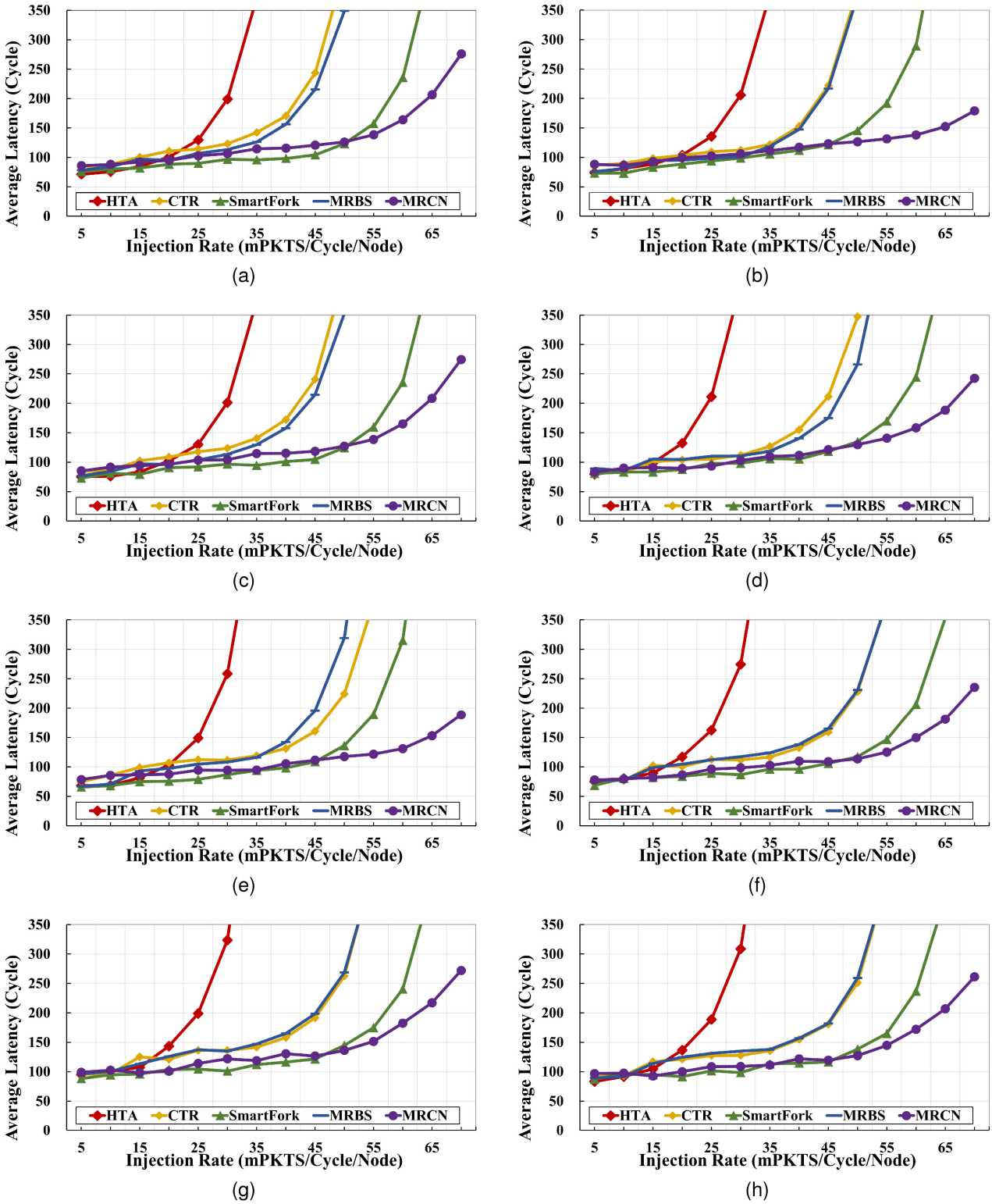


Fig. 10. Average latency simulation results in the 64-node customized NoCs with a degree higher than five under (a) Uniform random traffic with 5 % multicast intensity, (b) Uniform random traffic with 30 % multicast intensity, (c) CTG-based traffic with 5 % multicast intensity, (d) CTG-based traffic with 30 % multicast intensity, (e) Rent's rule traffic with 5 % multicast intensity, (f) Rent's rule traffic with 30 % multicast intensity, (g) DNN acceleration scenario, and (h) SMT scenario.

destinations based on the geometrical properties of the mesh with coordinate-based analysis. On the other hand, coordinate-based characteristics limited to the mesh cannot be applied to ADB of MRCN; instead, we defined labeled path detouring conditions that can be applied to all topologies. As a result, the difference in determining the multicast routing path in the mesh topology can be interpreted as the background demonstrating superior saturation point of HRA.

NoC traffic models that cause a high communication load on a specific path tend to be saturated when the packet

TABLE 5
Comparison of the Saturation Points in Customized NoCs With Average Degree Higher Than Five

Uniform random	Saturation point (mPKTS/Cycle/Node)					
Node Size	16	25	36	47	64	81
HTA	20.32	25.37	28.64	32.97	36.89	42.61
MRBS	21.55	28.74	32.11	40.01	42.81	50.00
CTR	23.41	30.54	34.14	41.30	44.15	50.50
SmartFork	29.11	35.52	42.78	49.59	54.16	63.73
MRCN	36.87	45.64	54.03	61.84	74.42	79.29
CTG -based	Saturation point (mPKTS/Cycle/Node)					
Node Size	16	25	36	47	64	81
HTA	17.33	23.87	25.69	28.91	22.83	39.79
MRBS	18.19	26.34	29.07	35.51	27.30	44.51
CTR	21.02	28.15	30.01	37.27	26.29	45.45
SmartFork	26.47	32.08	39.38	46.28	50.41	60.32
MRCN	34.71	40.37	51.18	55.92	71.73	75.12
Rent's rule	Saturation point (mPKTS/Cycle/Node)					
Node Size	16	25	36	47	64	81
HTA	18.14	22.09	23.66	28.75	19.71	36.36
MRBS	16.95	24.60	28.87	33.68	27.30	44.95
CTR	19.35	25.01	27.80	35.17	25.99	42.56
SmartFork	25.77	31.12	36.20	42.34	47.04	55.35
MRCN	31.33	39.87	48.53	52.70	65.69	68.95
SMT Scenario	Saturation point (mPKTS/Cycle/Node)					
Node Size	16	25	36	47	64	81
HTA	17.93	20.75	25.67	29.00	21.06	35.92
MRBS	19.01	24.26	28.24	34.16	27.45	42.87
CTR	20.58	25.08	29.11	36.82	27.46	44.66
SmartFork	23.55	30.37	37.71	42.19	46.58	54.39
MRCN	32.07	40.53	47.86	52.66	64.85	70.30
DNN acc. Scenario	Saturation point (mPKTS/Cycle/Node)					
Node Size	16	25	36	47	64	81
HTA	15.72	17.78	22.07	27.44	18.56	35.11
MRBS	17.10	21.95	23.63	29.73	21.55	39.38
CTR	17.12	22.73	28.85	34.40	21.87	42.57
SmartFork	24.49	28.76	35.13	39.86	43.94	50.24
MRCN	26.47	35.94	42.40	48.85	60.82	66.42

injection rate is relatively lower than other traffic models. Because these traffic models make links unavailable owing to the high communication load concentrated on a specific path, the NoC quickly reaches the acceptable limit that can cope with congestion. Uniform random traffic generates relatively even contentions across channels. In CTG-based traffic, packets are concentrated on specific paths, and channels included in these paths exhibit high loads. In CTG-based traffic, congestion frequently occurs on several links where communication paths corresponding to the task graph overlies. Rent's rule traffic generates traffic-intensive links among high-degree routers, creating wide traffic hotspots in the surrounding areas of these routers. Therefore, uniform random traffic with a distributed channel load has a lower average latency compared with other traffic models. Therefore, the packet injection rate at which each NoC is saturated in uniform random traffic is the highest among traffic models. On the other hand, CTG-based traffic, which concentrates the channel load on a specific path, shows a relatively low saturation point compared to other traffic models.

The saturation points for meshes of varying network sizes are presented in Table 3. In 9×9 mesh, the improvement over tree-based approaches of MRCN, which was 5.61 mPKTS/Cycle/Node in 4×4 mesh, increases to 13.23 mPKTS/Cycle/Node. In addition, the improvement over HOECP of MRCN increases from 0.61 mPKTS/Cycle/Node in 4×4 mesh to 1.72 mPKTS/Cycle/Node in 9×9 mesh. The results imply that ADB in MRCN significantly enhances NoC performance by preventing frequent contention as the network expands.

Fig. 9 depicts the average latency according to the injection rate in the 64-node customized NoCs with a degree lower than five. As the injection rate increased, it was saturated in the order of HTA, MRBS, CTR, SmartFork, and MRCN. Adaptive routing in MRCN reduced contentions by allocating alternative paths according to the traffic conditions, thus resulting in a 19.61 % greater saturation point than tree-based approaches.

CTR and MRBS showed similar average latency and saturation points. This is because CTR- and MRBS-based NoCs adopted tree-based minimal routing in common. When contention arises, CTR and MRBS store packets in the input buffer of a single channel and multiple channels, respectively. The different buffer management methods of MRBS and CTR result in a slight variation in the number of cycles required to transmit buffered data to the destination node. For this reason, even with the same simulation benchmarks, the CTR-based NoC shows a similar but not exactly the same average latency as the MRBS-based NoC.

Table 4 lists the saturation points for the customized NoCs with a degree lower than five by varying the network sizes. In 81-node networks, the improvement over tree-based approaches of MRCN, which was 3.02 mPKTS/Cycle/Node in 16-node networks, increases to 9.84 mPKTS/Cycle/Node. The results imply that ADB in MRCN significantly enhances NoC performance by preventing frequent contention as the network expands.

Fig. 10 depicts the average latency according to the injection rate in the 64-node customized NoCs with a degree higher than five. As the injection rate increased, it was saturated in the order of HTA, MRBS, CTR, SmartFork, and MRCN. Adaptive routing in MRCN reduced contentions by allocating alternative paths according to the traffic conditions, thus resulting in a 27.22 % greater saturation point than tree-based approaches.

Table 5 lists the saturation points for the customized NoCs with a degree higher than five by varying the network sizes. In 81-node networks, the improvement over tree-based approaches of MRCN, which was 1.98 mPKTS/Cycle/Node in 16-node networks, increases to 13.61 mPKTS/Cycle/Node. The results imply that ADB in MRCN significantly enhances NoC performance by preventing frequent contention as the network expands.

At the same injection rate, the average latency of the network and the latency improvement of MRCN increased when the average degree of the NoC was heightened. The result implies that MRCN avoids contention by fully utilizing path diversity in a topology with a high degree and link complexity.

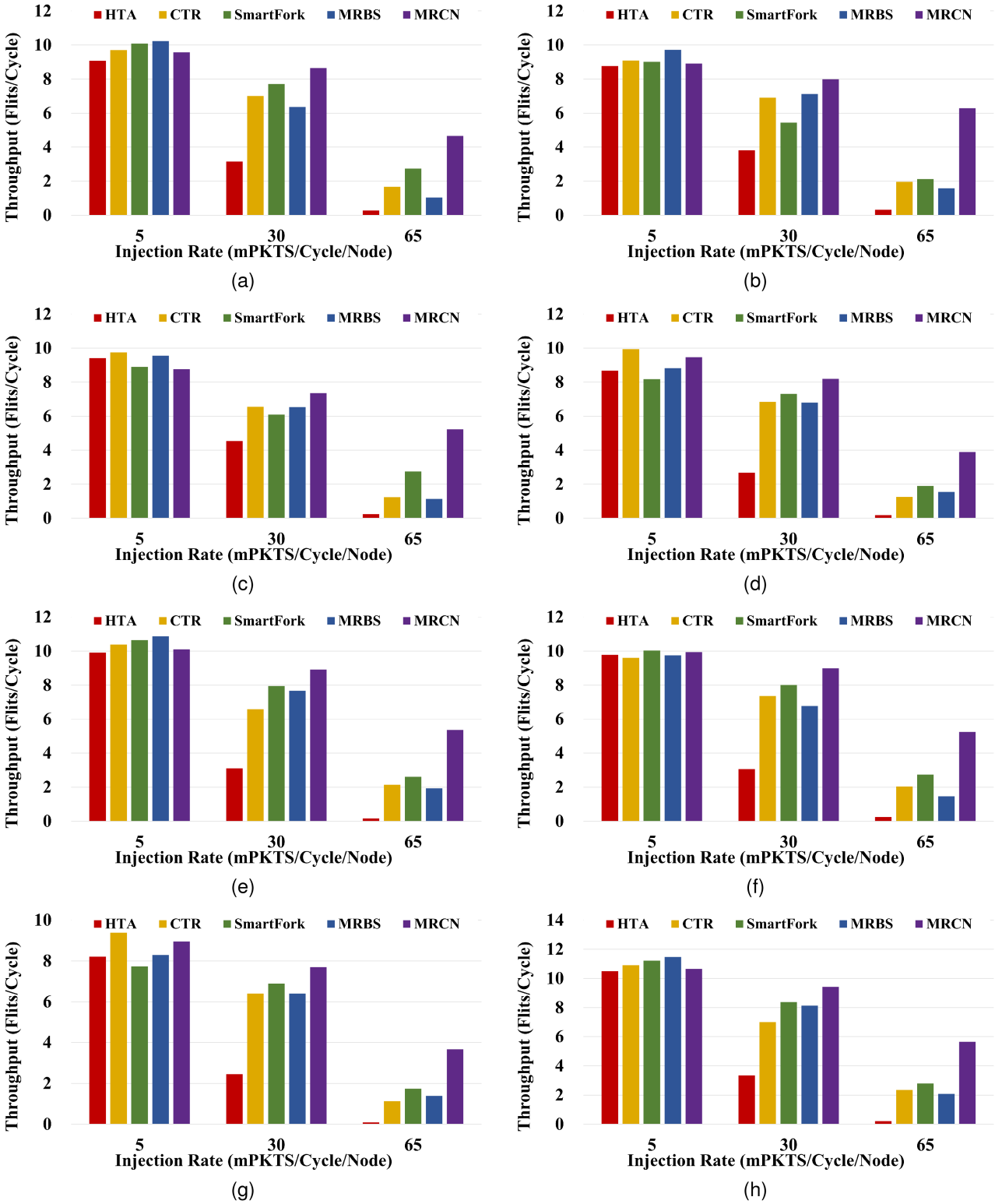


Fig. 11. Throughput simulation results in customized NoCs under (a) Uniform random traffic with 5 % multicast intensity, (b) Uniform random traffic with 30 % multicast intensity, (c) CTG-based traffic with 5 % multicast intensity, (d) CTG-based traffic with 30 % multicast intensity, (e) Rent's rule traffic with 5 % multicast intensity, (f) Rent's rule traffic with 30 % multicast intensity, (g) DNN acceleration scenario, and (h) SMT scenario.

4.2.2 *Throughput*

Fig. 11 illustrates the throughput for each traffic model in customized NoCs. The throughput is measured at 0.005, 0.03, and 0.065 PKTS/Cycle/Node, corresponding to near-zero loads, HTA saturation point, and MRCN saturation point. MRCN achieves 12.16 % higher throughput than best-case tree-based routing at 0.03 PKTS/Cycle/Node. This difference increases considerably at 0.065 PKTS/Cycle/Node, where all tree-based approaches are saturated. ADB of MRCN maximizes throughput by utilizing

TABLE 6
Area Comparison Among 5-Port Routers in ACP, HOECP, HRA, HTA, CTR, SmartFork, MRBS, and MRCN

AREA(um^2)	ACP	HOECP	HRA	HTA	CTR	SmartFork	MRBS	MRCN
RC	16,168	17,563	17,375	15,017	30,461	16,837	18,093	18,731
SA	9,847	9,847	9,847	8,154	10,445	17,471	10,445	10,445
X-bar	5,074	5,074	5,074	4,748	6,597	6,883	6,475	5,074
Input buffer	51,366	51,366	51,366	51,366	102,732	102,732	51,366	51,366
Deadlock recovery logic	-	-	-	-	-	5,375	1,374	-
Total Area	82,455	83,850	83,662	79,285	150,235	150,298	87,753	85,616

the path diversity of customized NoC to the fullest extent possible.

The result reveals that when the injection rate increases, all multicast routings in the customized NoCs reach their saturation points earlier than MRCN, indicating that MRCN has improved throughput and average latency compared to prior arts at high injection rates.

At 0.065 PKTS/Cycle/Node, the throughput gap between MRCN and tree-based approaches widened as the traffic distribution became asymmetric. The throughput gap between SmartFork and MRCN was 21.06 % and 49.92 % more significant in CTG-based and Rent's rule, respectively than in uniform random. Even when contention is concentrated on a single spot, ADB in MRCN minimizes throughput degradation.

MRBS provides higher throughput than MRCN with a low injection rate of 5 mPKTS/Cycle/Node. Minimal tree-based routings, including MRBS, provide maximized throughput in contention-free conditions. On the other hand, MRCN, which operates adaptively to traffic conditions, is not easily saturated in a situation where contention is frequent. Owing to this difference, MRCN shows higher throughput in Figs. 11d and 11f, which causes the high multicast intensity of 30 % and intensive traffic on a specific link, and MRBS has higher throughput in traffic conditions with little contention.

4.2.3 Area Overhead

Table 6 lists the area comparison of 5-port routers in CTR, HTA, ACP, HOECP, HRA, SmarFork, MRBS, and MRCN synthesized through SynopsysÂ® Design Compiler [37] under SAED 32 nm library [38]. CTR employs an input buffer large enough to store entire packets for cut-through routing, whereas SmartFork employs multiple VCs. CTR and SmartFork require additional input buffers, resulting in an area of 75.48 % and 75.55 % larger than MRCN, respectively. Other routers have the same size as the input buffer; however, the RC logic varies depending on the routing approach. The complexity of the route computation algorithm increases in the order of ACP, HOECP, HRA, and MRCN, correspondingly raising the RC logic area. Although MRCN is a multicast routing solution that is not limited to a mesh and exhibits a higher complexity than the mesh-optimized routing method, the overall router revealed area differences less than 4.36 %.

4.2.4 Energy Overhead

Additional simulation was conducted using the energy estimation function provided by Noxim. First, the power estimation function of Synopsys Design CompilerÂ® and the power model of the SAED 32 nm Design Kit were applied. In addition, the switching activity information (SAIF) file was extracted from the average delay simulation results, including the toggle counts and signal retention time. Using the SAIF file for the signal transition-aware simulation further improves the power estimation accuracy. The power model of multicast routers was converted into an energy model using Orion 3.0 [39] and integrated into Noxim. The simulation was conducted at an 8×8 mesh structure and an injection rate of 30 mPKTS/Cycle/Node for variable control.

The energy estimation results based on the simulation are presented in Table 7. MRCN showed the highest energy efficiency among multicast routings applicable to customized NoCs. The result means that the energy overhead of routing computation logic for path-based routing is lower than additional hardware resources such as VC and DRU to avoid multicast deadlock. The VC allocation and deadlock recovery schemes of tree-based approaches show high energy consumption because they must be activated in entire cycles while the chip operates. On the other hand, route computation of MRCN operates only when the packet header is stored, resulting in relatively low energy consumption compared with the tree-based approaches.

TABLE 7
Comparison of Energy Consumption Among ACP, HOECP, HRA, HTA, CTR, SmartFork, MRBS, and MRCN

	Uniform random traffic	CTG-based traffic	Rent's rule traffic	DNN acc. scenario	SMT scenario
ACP	135,973 nJ	163,243 nJ	156,413 nJ	183,566 nJ	169,998 nJ
HOECP	138,090 nJ	165,797 nJ	158,883 nJ	186,47 nJ	172,675 nJ
HRA	137,760 nJ	165,325 nJ	158,509 nJ	186,034 nJ	172,239 nJ
HTA	181,158 nJ	217,409 nJ	208,387 nJ	244,630 nJ	226,510 nJ
CTR	228,145 nJ	273,846 nJ	262,379 nJ	307,999 nJ	285,225 nJ
SmartFork	229,534 nJ	275,475 nJ	264,063 nJ	309,912 nJ	286,957 nJ
MRBS	194,640 nJ	213,583 nJ	206,379 nJ	245,335 nJ	230,852 nJ
MRCN	140,786 nJ	168,969 nJ	161,978 nJ	190,064 nJ	176,015 nJ

5 Conclusion

This study proposes deadlock-free and throughput-enhanced multicast routing in customized NoCs. MRCN prevents the multicast deadlock with negligible additional hardware overhead. Furthermore, by adapting the routing paths and the branched destinations to the network conditions, MRCN minimizes the congestion to enhance the throughput. Simulation results with various network conditions indicate that MRCN achieves 13.98 % lower average latency and 12.16 % higher average throughput than tree-based multicast routing approaches.

References

[1] A. Gamatie, G. Devic, G. Sassatelli, S. Bernabovi, P. Naudin, and M. Chapman, "Towards energy-efficient heterogeneous multicore architectures for edge computing," *IEEE Access*, vol. 7, pp. 49474–49491, 2019.

[2] D. Shin, J. Lee, J. Lee, J. Lee, and H. -J. Yoo, "DNPU: An energy-efficient deep-learning processor with heterogeneous multi-core architecture," *IEEE Micro*, vol. 38, no. 5, pp. 85–93, Sep./Oct. 2018.

[3] H.-E. Zahaf, G. Lipari, M. Bertogna, and P. Boulet, "The parallel multi-mode digraph task model for energy-aware real-time heterogeneous multi-core systems," *IEEE Trans. Comput.*, vol. 68, no. 10, pp. 1511–1524, Oct. 2019.

[4] B. K. Joardar, R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu, "Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems," *IEEE Trans. Comput.*, vol. 68, no. 6, pp. 852–866, Jun. 2019.

[5] H. Zheng, K. Wang, and A. Louri, "Adapt-NoC: A flexible network-on-chip design for heterogeneous manycore architectures," in *Proc. IEEE Int. Symp. High- Perform. Comput. Archit.*, 2021, pp. 723–735.

[6] S. M. Nabavinejad, M. Baharloo, K. -C. Chen, M. Palesi, T. Kogel, and M. Ebrahimi, "An overview of efficient interconnection networks for deep neural network accelerators," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no, 3, pp. 268–282, Sep. 2020.

[7] S. Y. H. Mirmahaleh and A. M. Rahmani, "DNN pruning and mapping on NoC-based communication infrastructure," *Microelectron J.*, vol. 94, Dec. 2019, Art. no. 104655.

[8] K.-C. Chen, M. Ebrahimi, T.-Y. Wang, and Y.-C. Yang, "NoC-based DNN accelerator: A future design paradigm," in *Proc. IEEE/ACM 13th Int. Symp. Netw.-Chip*, 2019, pp. 1–8.

[9] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[10] N. E. Jerger, L. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Proc. Int. Symp. Comput. Archit.*, 2008, pp. 229–240.

[11] S. Ma, N. E. Jerger, and Z. Wang, "Efficient design of fully adaptive routing algorithms for networks-on-chip," in *Proc. IEEE Int. Symp. High-Perform. Comp Archit.*, 2012, pp. 1–12.

[12] S. Abadal, R. Martínez, J. Solé-Pareta, E. Alarcón, and A. Cabellos-Aparicio, "Characterization and modeling of multicast communication in cache-coherent manycore processors," *Comput. Electr. Eng.*, vol. 51, pp. 168–183, Apr. 2016.

[13] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2-D mesh multicomputers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 8, pp. 793–804, Aug. 1994.

[14] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "Resource deadlocks and performance of wormhole multicast routing algorithms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 6, pp. 535–549, Jun. 1998.

[15] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, J. Flich, and H. Tenhunen, "Path-based partitioning methods for 3D networks-on-chip with minimal adaptive routing," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 718–733, Mar. 2014.

[16] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "HAMUM – A novel routing protocol for unicast and multicast traffic in MPSoCs," in *Proc. 18th Euromicro Int. Conf. Parallel Distrib. Netw.-Based Process.*, 2010, pp. 525–532.

[17] P. Bahrebar and D. Stroobandt, "The hamiltonian-based odd-even turn model for maximally adaptive routing in 2D mesh networks-on-chip," *J. Comput. Elect. Eng.*, vol. 45, pp. 386–401, Jul. 2015.

[18] M. P. Malumbres, J. Duato, and J. Torrellas, "An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors," in *Proc. IEEE 8th Symp. Parallel Distrib. Process.*, 1996, pp. 186–189.

[19] D. R. Kumar, W. A. Najjar, and P. K. Srimani, "A new adaptive hardware tree-based multicast routing in k-ary n-cubes," *IEEE Trans. Comput.*, vol. 50, no. 7, pp. 647–659, Jul. 2001.

[20] L. Wang, Y. Jin, H. Kim, and E. J. Kim, "Recursive partitioning multicast: A bandwidth-efficient routing for Networks-on-Chip," in *Proc. IEEE Symp. Netw.-on-Chip*, 2009, pp. 64–73.

[21] M. Zhong et al., "An improved minimal multicast routing algorithm for mesh-based networks-on-chip," in *Proc. IEEE Int. Conf. Signal Process. Commun. Comput.*, 2014, pp. 755–779.

[22] Z. Wang, H. Gu, Y. Yang, H. Zhang, and Y. Chen, "An adaptive partition-based multicast routing scheme for mesh-based networks-on-chip," *J. Comput. Elect. Eng.*, vol. 51, pp. 235–251, Apr. 2016.

[23] D. Konstantinou, C. Nicopoulos, J. Lee, G. C. Sirakoulis, and G. Dimitrakopoulos, "SmartFork: Partitioned multicast allocation and switching in network-on-chip routers," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2020, pp. 1–5.

[24] M. C. Yang, Y. S. Lee, and T. H. Han, "MRBS: An area-efficient multicast router for network-on-chip using buffer sharing," *IEEE Access*, vol. 9, pp. 168783–168793, 2021.

[25] E. G. Coffman, M. Elphick, and A. Shoshani, "System deadlocks," *J. ACM Comput. Surv.*, vol. 3, no. 2, pp. 67–78, Jun. 1971.

[26] C. -W. Wu, K. -J. Lee, and A. P. Su, "A hybrid multicast routing approach with enhanced methods for mesh-based networks-on-chip," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1231–1245, Sep. 2018.

[27] P. R. Panda, "SystemC: A modeling platform supporting multiple design abstractions," in *Proc. 14th Int. Symp. Syst. Synth.*, 2001, pp. 75–80.

[28] V. Catania et al., "Cycle-accurate network on chip simulation with noxim," *ACM Trans. Model. Comput. Simul.*, vol. 27, no. 1, pp. 1–25, Aug. 2016.

[29] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. 6th Int. Workshop Hardware/Softw. Codesign*, 1998, pp. 97–101.

[30] S. Tosun, V. B. Ajabshir, O. Mercanoglu, and O. Ozturk, "Fault-tolerant topology generation method for application-specific network-on-chips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 9, pp. 1495–1508, Sep. 2015.

[31] M. Y. Lanzerotti, G. Fiorenza, and R. A. Rand, "Microminiature packaging and integrated circuitry: The work of E. F. Rent, with an application to on-chip interconnection requirements," *J. IBM Res. Develop.*, vol. 46, no. 4.5, pp. 777–803, Jul. 2005.

[32] D. Konstantinou, C. Nicopoulos, J. Lee, G. C. Sirakoulis, and G. Dimitrakopoulos, "SmartFork: Partitioned multicast allocation and switching in network-on-chip routers," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2020, pp. 1–5.

[33] S. Abadal, R. Martínez, J. Solé-Pareta, E. Alarcón, and A. Cabellos-Aparicio, "Characterization and modeling of multicast communication in cache-coherent manycore processors," *Comput. Electr. Eng.*, vol. 51, pp. 168–183, Apr. 2016.

[34] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *Proc. IEEE Int. Electron Devices Meeting*, 2019, pp. 32.5.1–32.5.4.

[35] N. Binkert et al., "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

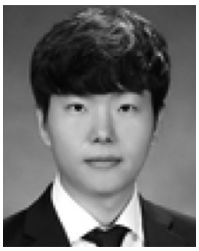
[36] A. A. K. Nielsen et al., "Genetic circuit design automation," *Science*, vol. 352, no. 6281, Apr. 2016, Art. no. aac7341.

[37] S. Gayathri and T. C. Taranath, "RTL synthesis of case study using design compiler," in *Proc. Int. Conf. Electr. Electron. Commun. Comput. Optim. Techn.*, 2017, pp. 1–7.

[38] R. Goldman, K. Bartleson, T. Wood, V. Melikyan, and E. Babayan, "Synopsys' low power design educational platform," in *Proc. 9th Eur. Workshop Microelectronics Educ.*, 2012. pp. 23–26.

[39] A. B. Kahng, B. Lin, and S. Nath, "ORION3.0: A comprehensive NoC router estimation tool," *IEEE Embedded Syst. Lett.*, vol. 7, no. 2, pp. 41–45, Jun. 2015.

Young Sik Lee (Student Member, IEEE) received the BS, MS, and PhD degrees in electronic and electrical engineering from Sungkyunkwan University, Suwon, South Korea, in 2016, 2018, and 2022, respectively. Since 2022, he has been with the Memory Division, Samsung Electronics, South Korea, where he worked on HBM design. His research interests include network-on-chip for SoC/chiplet and advanced memory architecture.



Yong Wook Kim (Student Member, IEEE) received the BS degree in electronic and electrical engineering from Sungkyunkwan University, Suwon, South Korea, in 2018, where he is currently working toward the MS and PhD degrees in electrical and computer engineering. His research interests include NoC, machine learning, and computer architecture.

Tae Hee Han (Senior Member, IEEE) received the BS, MS, and PhD degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2006, he was with the Telecom R&D center of Samsung Electronics, where he developed 3G wireless, mobile TV, and mobile WiMax handset chipsets. Since March 2008, he has been with Sungkyunkwan University, Suwon, Korea, as a professor. From 2011 to 2013, he had served as a full-time advisor on System ICs for the Korean Government. His current research interests include SoC/Chiplet architectures for AI, advanced memory architecture, network-on-chip, and system-level design methodologies.