MRBS: An Area-Efficient Multicast Router for Network-on-Chip using Buffer Sharing

MIN CHAE YANG1, (Student Member, IEEE), YOUNG SIK LEE2, (Student Member, IEEE) TAE HEE HAN3, (Member, IEEE)
1Department of Artificial Intelligence, Sungkyunkwan University, Suwon 16419, South Korea
2Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea
3Department of Semiconductor Systems Engineering, Sungkyunkwan University, Suwon 16419, South Korea
Corresponding author: Tae Hee Han (e-mail: than@skku.edu).

This research was supported in part by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2019-0-00421, AI Graduate School Support Program (Sungkyunkwan University)), in part by National R&D Program through the National Research Foundation of Korea (NRF) funded by Ministry of Science and ICT (No. 2020M3H2A1076786), in part by the Ministry of Trade, Industry and Energy (MOTIE) under Grant 20011074 and in part by the MOTIE and the Korea Evaluation Institute of Industrial Technology (KEIT) under Grant 20010560.

ABSTRACT Network-on-chip (NoC) has become the mainstream fabric architecture for chip multiprocessor (CMP) design. Owing to the market-driven advancement of modern applications in CMP, multicast traffic is aggressively increasing to support barrier synchronization, multithreading, and cache coherence protocols. Although multicast by branching of packets in the NoC router facilitates shortest path routing, additional branching-induced deadlocks must be circumvented. Existing NoC studies on deadlock-free minimal path routing in multicast traffic have typically deployed additional virtual channels or large buffers to hold entire packets, thereby significantly increasing the router area. Focusing on the area-efficient solution while sustaining the performance, we propose a novel multicast router using buffer sharing (MRBS) to guarantee deadlock-free multicast routing by exploiting the spatial diversity of the input buffer. MRBS ensures minimal path routing without requiring additional virtual channels or large buffers to hold entire packets. Extensive experiments were conducted by varying the buffer, packet, and network sizes, as well as the number of destinations per packet, under random multicast traffic with diverse injection rates. Simulation results show that MRBS achieves a 39.3% improvement in the area-delay product on average for various network sizes compared to the conventional tree-based router.

INDEX TERMS Area-efficient design, buffer sharing, deadlock recovery, multicast communication, network-on-chip, router architecture, tree-based routing

I. INTRODUCTION With advancing semiconductor manufacturing technology, large-scale chip multiprocessor (CMP)-based system-on-chips (SoCs) have become commonplace. The SoC interconnection architecture, which is responsible for the communication among the massive processing elements, plays a key role in the overall system performance. Accordingly, network-on-chips (NoCs) have rapidly pervaded state-of-the-art SoCs [1].

Multicast messages are responsible for the efficient execution of several parallel applications, such as search algorithms, graph algorithms, and matrix operations [2]. In addition, multicast messages are commonly employed in the management and network configuration, synchronization, and cache coherence protocols. The lack of multicast support has impeded the expansion of the capabilities of NoCs because multicast traffic has a significant impact on modern multicore computing environments [3]. The main challenge of multicast communication involves additional deadlock cases caused by the branching of packets.

The root cause of a deadlock in multicast systems is a branch operation, which is essential for minimal path routing. Most previous studies have addressed this problem by offering large buffers to hold entire packets [4]-[5], or virtual channels [6] to implement deadlock-free multicast routers without harming minimal path routing. However, these ap-
proaches have a shortcoming in that the input buffer size in all routers must be guaranteed to be equal to or greater than the maximum multicast packet length, which leads to significant area overhead. A typical NoC router consumes a notable portion of its area on its buffer; thus, the area overhead will be more severe as the required communication bandwidth increases [7]. Therefore, conventional deadlock-free multicast routers are inefficient in terms of buffer space utilization. We leverage this observation to propose a low-cost multicast router using buffer sharing (MRBS) that guarantees deadlock freedom without employing additional virtual channels or large buffers to hold entire packets.

This study explores opportunities to find a deadlock-free multicast router architecture that enhances area efficiency while sustaining performance. To efficiently utilize the buffer space, especially for multicast traffic patterns, the proposed router architecture manages all input buffers in a router as a register file style instead of dedicating a single buffer per input port. Sharing of buffer resources facilitates the following. (1) Designing a buffer-sharing router architecture to maximize resource utilization. (2) Resolving a multicast deadlock in a tree-based router sustaining a minimal path. (3) Compromising the trade-offs in the design space, focusing on the area-delay product of the MRBS compared to a conventional tree-based router.

The remainder of this paper is organized as follows. Section 2 presents the background and related work. Section 3 describes the proposed technique in terms of the router architecture and deadlock recovery procedure. The simulation results and analysis under various conditions are provided in Section 4. Finally, concluding remarks are drawn in Section 5.

II. BACKGROUND & RELATED WORK

Various efforts have been made to improve the multicast performance of NoCs, which can largely be categorized into tree-based methods that support branch operations inside a router and path-based ones without branch operations.

A. MULTICAST ROUTING

Path-based multicast routes packets using node-labeling. The XY coordinates of each node determine the label. The path-based method divides all nodes into groups based on labels and sends duplicate packets to each group, which includes one or more destination nodes, as shown in Fig. 1 (a).

The dual path routing method [8] divides destinations into two groups, one with labels smaller than the source node and the other with larger labels. Depending on the node labeling, packets are sent to the higher or lower label groups. Several existing path-based algorithms use different partitioning techniques to reduce path lengths. Multipath routing improves the dual path method by segmenting four rectangular subset groups based on source node location. Nodes in the same subgroup are assigned to the same path, preventing long routing paths. The Hamiltonian adaptive multicast unicast model by Ebrahimi et al. [9] can encompass more routing paths, increasing the probability that the packet will choose a less congested path.

In tree-based routing, the source node builds a spanning tree and sends a single multicast message down the tree. Fig. 1 (b) shows a simple tree-based routing example. When routing multi-destination packets, the source node is the root and the destination nodes are leaves. Using shortest-path routing, the packets are branched into different links at the intermediate node. In order to duplicate flits at intermediate nodes, the packet must be forwarded simultaneously in all directions. A blocked path prevents the flit from being duplicated, increasing the risk of a multicast deadlock. In other words, all tree-based methods are vulnerable to multicast deadlocks.

For each router in the hardware-tree based multicast routing algorithm [10], a deadlock queue was implemented. Dual virtual channels were used in recursive partitioning multicast [11] and improved minimal multicast routing [12] to avoid deadlock. According to [13], the authors solved the cyclic dependency deadlock by adaptively selecting a routing path based on the downstream router’s remaining buffer size.

Path-based routing without branch operations is advantageous for avoiding multicast deadlocks. However, the unbalanced partitioning of path-based routings caused by boundary-located source nodes suffers from non-minimal routing paths. The tree-based multicast routing has low latency but needs to solve multicast deadlocks through additional hardware resources [14]. Conventional approaches have eliminated deadlock by employing deep FIFO buffers to hold the largest packet or virtual channels to prevent the input buffer preemption. However, these methods dramatically increase the input buffer size, which occupies most of the router area. MRBS reduces the input buffer size by buffer sharing scheme while preserving the low latency that is the strength of the tree-based approach.

B. DEADLOCK IN MULTICAST ROUTING

In wormhole-switched NoCs, contention between packets causes head-of-line (HoL) blocking, leading to performance

![Figure 1. Examples of (a) path- and (b) tree-based approaches](image-url)
deterioration or communication failure; thus, deadlock-free is a vital design objective. A deadlock in unicast is mainly caused by a cyclic dependency between packets and is easily rescued by restricting the turn models at the routing stage [15]. However, in multicast routing, different types of deadlocks owing to the branching of packets must also be entirely handled.

Fig. 2 shows an example of a multicast deadlock. The dashed line indicates a port request that is not yet granted, and the solid line represents a port request. Packet A requests the southern port of Router 2, which is blocked by packet B, whereas packet B demands the southern port of Router 3 held by packet A. Because of branching of packet A in Router 1, one path blocks packet B, and the other is blocked by packet B. In this situation, a request to the southern output port of packet A can be granted, whereas a request to the eastern output port cannot, resulting in a multicast deadlock.

Coffman et al. [16] showed that a deadlock occurs if and only if four situations, namely mutual exclusion, hold and wait, no preemption, and circular wait, fulfil simultaneously. These four situations can be rephrased as:

- Mutual exclusion: each output port is not sharable
- Hold and wait: a packet is holding one output port and waiting for another
- No preemption: output ports cannot be preempted
- Circular wait: a circular chain of packet exists

Conventional tree-based routers dismissed hold and wait situation by enlarging buffer size, which results area overhead. In contrast, MRBS aims to release hold and wait situation without requiring large buffers to hold entire packets.

A long packet transmission distance increases the probability of deadlock. The longer the packet transmission distance, the more frequently HoL blocking occurs [17]-[18]. Because HoL blocking generates a hold-and-wait condition, one of Coffman’s conditions, it can result in deadlock.

C. BUFFER SHARING

There have been a number of studies on the NoC router with a centralized shared input buffer [19]–[21] and distributed input buffer sharing scheme [22]–[23]. Centralized shared input buffer structure is a single buffer shared by all input ports. Distributed input buffer sharing is a scheme that allows other input ports to share a distributed input buffer dedicated to each input port through multiplexing. Li et al. designed a router with the centralized shared buffer and proposed a packet merging mechanism to improve the utilization of buffer space and network throughput for dataflow architecture [19]. Bhardwaj et al. introduced a continuous-time replication strategy in asynchronous NoC and improved throughput while minimizing power and area overhead using a centralized shared buffer [20]. Shared-queue router architecture was designed to maximize buffer utilization for boosting network throughput [21]. The novel flexible buffering mechanisms were proposed to improve average throughput for 3D NoCs [22]. The first approach increases the utilization of underutilized buffers based on the number of free slots available in each port, prioritizing minimum occupied buffers. The second mechanism is the flexible buffering technique employing a simple priority order for each buffer.

A Flexible router provided a way to handle the requests to a busy buffer by other buffers to increase saturation rate while sustaining area [23].

MRBS targets a different buffer size spectrum compared to the routers used in [19]–[23] while using distributed input buffer sharing scheme. When each input buffer size is smaller than the maximum packet size, we guarantee area-efficient deadlock freedom. On the other hand, other studies sought to improve performance when the buffer size was larger than the maximum packet length. Since they used a cut-through routing method that does not cause deadlock, the deadlock was not considered, and it occupies a considerably larger area than MRBS. Overall, our work differs from previous studies in that it dramatically reduces buffer size using a deadlock recovery scheme and adopts distributed input buffer sharing that enables faster packet access than a centralized shared buffer.

III. MRBS

The main concept behind MRBS is leveraging the spatial diversity of the input buffer for multicast deadlock recovery while maintaining the minimal path routing. A router architecture (Section III.A) and deadlock recovery scheme (Section III.B) are presented in realizing the idea.

A. OVERALL ROUTER ARCHITECTURE

Fig. 3 illustrates the proposed multicast router architecture. Instead of using dedicated deadlock queues [24]-[25], we utilize the input buffer as a deadlock queue only when multicast deadlock is detected. The versatile input buffer incorporates...
an input port selector and a deadlock recovery control unit (DRU) that enable multicast deadlock recovery to facilitate buffering sharing. The input port selector expedites the data to a specific input port to be forwarded to another available input buffer. The DRU controls the recovery operation and coordinates the handshaking signals (i.e., \( D_{req} \), \( D_{gnt} \), \( Rdy \), and \( Done \)) between the adjacent routers and the control signals for the input port selector. All handshaking signals were one-hot encoded to reduce the decoding delay. The associated notations are described in Table 1. To explicitly explain the multicast deadlock and recovery operation of the MRBS, trunk router \( R_{tr} \), twig router \( R_{tw} \), and greenleaf buffer \( B_{gl} \) are defined according to the branching flow of multicast packet.

![FIGURE 3. Overall router architecture: changes to support buffer sharing are shown in red](image)

**Trunk router** (\( R_{tr} \)): A router behaves as the trunk router \( R_{tr} \) where multicast packets are branched and corresponds to **Router 1** in Fig. 2. When a multicast deadlock occurs, some of the output ports of \( R_{tr} \) are blocked, and this becomes the *hold and wait* situation where branched packets cannot be sent to other output ports. DRU in \( R_{tr} \) performs deadlock detection and blocked packet forwarding to dismiss *hold and wait* situation.

**Twig router** (\( R_{tw} \)): A router acts as twig router \( R_{tw} \) where multicast packets are branched and corresponds to **Router 2** in Fig. 2. When a multicast deadlock occurs, branched packets cannot be propagated due to the preemption of other packets in the output port of \( R_{tw} \). DRU in \( R_{tw} \) performs buffer sharing and flit reordering by using input port selector.

**Greenleaf buffer** (\( B_{gl} \)): A Greenleaf buffer \( B_{gl} \) is an available input buffer in \( R_{tw} \) selected to keep the part of the multicast packet sent from \( R_{tr} \). By forwarding and storing the rest of the packet from \( R_{tr} \) to \( B_{gl} \), *hold and wait* situation in \( R_{tr} \) can be released. \( B_{gl} \) is determined through arbitration among the input buffers, excluding the buffer that is the cause of the deadlock. As shown in Fig. 2, there are four candidates of \( B_{gl} \) in a five-port router, excluding the west input buffer of \( R_{tw} \) (**Router 2**).

### TABLE 1. Notations of MRBS: \( r \) and \( t \) represent the row and column associated with the input and output port number, respectively

| Notation | Description |
|----------|-------------|
| \( N_p \) | The number of input and output ports |
| \( T_{thres} \) | Deadlock threshold time-out parameter |
| \( P_{L_{max}} \) | Maximum packet length |
| \( D_{buff} \) | FIFO depth of the input buffer |
| \( L_{prop} \) | Latency cycle required to propagate one hop |
| \( IB[i] \) | Input buffer corresponding to the \( i^{th} \) input port |
| \( CF[i] \) | Credit full signal of adjacent router connected with the \( i^{th} \) output port |
| \( TY[i] \) | The type of flit departing from the \( i^{th} \) input buffer |
| \( R[r][m] \) | Routing request matrix that describes the requests of all incoming flits to access the output ports |
| \( A[n][m] \) | Allocation matrix that describes the allocation state of all incoming flits to access the output ports |
| \( DR[r][m] \) | Deadlock recovery request matrix that describes the recovery requests of all incoming flits to access the output ports |
| \( D_{req}[i] \) | Deadlock recovery request signal from the \( i^{th} \) output port of \( R_{tr} \) |
| \( D_{gnt}[i] \) | Deadlock recovery grant signal to the \( i^{th} \) input port of \( R_{tw} \) |
| \( Rdy \) | Ready signal to upstream router of \( B_{gl} \) |
| \( Done \) | Done signal from upstream router of \( B_{gl} \) |

### B. MULTICAST DEADLOCK RECOVERY PROCEDURE

Since MRBS does not modify the pipeline operations and an additional operation is involved only when a multicast deadlock occurs, the extra cycles can be ignorable considering the rate of deadlock occurrence. In addition, we devote our efforts to minimizing the additional cycles caused by input port selector and DRU logic. A multicast deadlock recovery scheme using buffer sharing was applied simultaneously to multiple adjacent routers. This section describes the deadlock detection (step 1) and flit forwarding (step 2) on \( R_{tr} \) and input buffer sharing (step 3) and flit reordering (step 4) on \( R_{tw} \) for multicast deadlock recovery. Fig. 4 depicts the overall deadlock detection and recovery procedure.

1) Multicast Deadlock Detection

The multicast deadlock detection in MRBS is a heuristic timer-based, which implies that all input buffers are equipped with timers, and a deadlock is detected when the timer reaches a certain threshold. Thus, not only deadlocks but also contention by HoL blockings can be repaired immediately.

---

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
We set this threshold as the capability that the $R_{tw}$ can resolve worst-case contention by itself. When packets are stored in all input buffers of the $R_{tw}$, and all output ports are available, the maximum cycle $T_{thres}$ at which the contention in $R_{tw}$ is removed is as follows:

$$T_{thres} = PL_{max} \times (N_p - 1) + D_{buff} + L_{prop}$$

(1)

If HoL blocking is not eliminated during this time span, the DRU determines that there is a fault in routing the packet by the $R_{tw}$ and initiates the deadlock recovery procedure. Since we conducted the simulation in a mesh network with a maximum packet length of 10 flits, $PL_{max}$, $N_p$, $D_{buff}$, and $L_{prop}$ were set to 10, 5, 5, and 5, respectively.

Therefore, the MRBS effectively handles all possible traffic congestion caused by multicast packets. Because MRBS detects all time-out flits that occur during branch operations, it encompasses all multicast deadlock cases. We can identify whether time-out flit causes the multicast deadlock or not by checking the case which simultaneously satisfies the following two conditions.

- **Condition 1.** The time-out flit must request more than one output port:

  $$\exists_n : \sum_{m=0}^{N_p-1} R[n][m] \geq 2, \ n, m \in \{0, \cdots, N_p - 1\}$$

  (2)

- **Condition 2.** The time-out flit must be of a body or tail type:

  $$\exists_i : TS[i] = 1, TY[i] = BODY \text{ or TAIL, } \ i \in \{0, \cdots, N_p - 1\}$$

  (3)

**Condition 1** indicates the requirement for multicast packets. Because MRBS detects only multicast deadlocks, unbranched packets are treated as congestion. **Condition 2** indicates that a multicast deadlock always occurs after the packet header is branched. Therefore, a time-out flit conforming both **Conditions 1** and 2 causes multicast deadlock, and a router including a time-out flit becomes a $R_{tr}$.

2) Flit Forwarding

MRBS minimizes additional congestion during buffer sharing by diminishing the number of $R_{tw}$. This is accomplished by exploiting the buffer state of the downstream router. When a multicast deadlock is detected, the DRU sends $DR_{req}$ only to the output port whose credit status is full among the output ports requested by a multicast packet. A request to a port whose credit status is not full can be granted, and thus, it is not the source of a deadlock. The routers that receive $DR_{req}$ become $R_{tw}$. Here, $DR_{req}$ is created by the DRU and maintained until the tail flit of the multicast packet is removed from the buffer.

$R_{tw}$ sends $DR_{gnt}$ to $R_{tr}$ through the setup procedure described in step 3. $R_{tr}$ that has received $DR_{gnt}$ forwards a multicast flit, which is the flit of multicast packet (previously, the credit status was full, and thus it cannot be forwarded). This process conducted by sending a forward signal to the input buffer, which holds multicast flits when $DR_{gnt}$ corresponding to $DR_{req}$ is detected. The request of the multicast flit in the input buffer that receives the forward signal is always granted without considering credit status. The forward signal is maintained until the tail flit of the multicast packet is forwarded. **Algorithm 1** presents the above process of $R_{tr}$ operation of the MRBS.

We set this threshold as the capability that the $R_{tw}$ can resolve worst-case contention by itself. When packets are stored in all input buffers of the $R_{tw}$, and all output ports are available, the maximum cycle $T_{thres}$ at which the contention in $R_{tw}$ is removed is as follows:

$$T_{thres} = PL_{max} \times (N_p - 1) + D_{buff} + L_{prop}$$

(1)

If HoL blocking is not eliminated during this time span, the DRU determines that there is a fault in routing the packet by the $R_{tw}$ and initiates the deadlock recovery procedure. Since we conducted the simulation in a mesh network with a maximum packet length of 10 flits, $PL_{max}$, $N_p$, $D_{buff}$, and $L_{prop}$ were set to 10, 5, 5, and 5, respectively.

Therefore, the MRBS effectively handles all possible traffic congestion caused by multicast packets. Because MRBS detects all time-out flits that occur during branch operations, it encompasses all multicast deadlock cases. We can identify whether time-out flit causes the multicast deadlock or not by checking the case which simultaneously satisfies the following two conditions.

- **Condition 1.** The time-out flit must request more than one output port:

$$\exists_n : \sum_{m=0}^{N_p-1} R[n][m] \geq 2, \ n, m \in \{0, \cdots, N_p - 1\}$$

(2)

- **Condition 2.** The time-out flit must be of a body or tail type:

$$\exists_i : TS[i] = 1, TY[i] = BODY \text{ or TAIL, } \ i \in \{0, \cdots, N_p - 1\}$$

(3)

**Condition 1** indicates the requirement for multicast packets. Because MRBS detects only multicast deadlocks, unbranched packets are treated as congestion. **Condition 2** indicates that a multicast deadlock always occurs after the packet header is branched. Therefore, a time-out flit conforming both **Conditions 1** and 2 causes multicast deadlock, and a router including a time-out flit becomes a $R_{tr}$.

2) Flit Forwarding

MRBS minimizes additional congestion during buffer sharing by diminishing the number of $R_{tw}$. This is accomplished by exploiting the buffer state of the downstream router. When a multicast deadlock is detected, the DRU sends $DR_{req}$ only to the output port whose credit status is full among the output ports requested by a multicast packet. A request to a port whose credit status is not full can be granted, and thus, it is not the source of a deadlock. The routers that receive $DR_{req}$ become $R_{tw}$. Here, $DR_{req}$ is created by the DRU and maintained until the tail flit of the multicast packet is removed from the buffer.

$R_{tw}$ sends $DR_{gnt}$ to $R_{tr}$ through the setup procedure described in step 3. $R_{tr}$ that has received $DR_{gnt}$ forwards a multicast flit, which is the flit of multicast packet (previously, the credit status was full, and thus it cannot be forwarded). This process conducted by sending a forward signal to the input buffer, which holds multicast flits when $DR_{gnt}$ corresponding to $DR_{req}$ is detected. The request of the multicast flit in the input buffer that receives the forward signal is always granted without considering credit status. The forward signal is maintained until the tail flit of the multicast packet is forwarded. **Algorithm 1** presents the above process of $R_{tr}$ operation of the MRBS.
3) Input Buffer Sharing

\( R_{tw} \) that has received \( DR_{req} \) needs to select \( B_{gl} \) and send \( DR_{gnt} \) to \( R_{tw} \) through the setup procedure. The setup procedure of \( B_{gl} \) is essential to preventing the worst-case of losing packets. This setup procedure is controlled by the DRI and proceeds in the following order: (1) When \( DR_{req} \) is detected, \( R_{tw} \) selects \( B_{gl} \) through arbitration (in a round-robin manner) among the remaining buffers, except for the input buffer that causes a deadlock. In addition, an empty buffer (by detecting the flag from the input buffer) has the highest priority to minimize the deadlock recovery cycle. (2) Here, \( Rdg \) is sent to the router or network interface (when \( B_{gl} \) is a local input buffer) connected to \( B_{gl} \) to avoid receiving any more packets. (3) When \( Rdg \) is detected in the router or network interface, the switch allocator is controlled such that the corresponding output port does not grant any packet and Done is sent to \( R_{tw} \). The delay added by the setup procedure is approximately two cycles (depending on packet length, which is pre-allocated to \( B_{gl} \)).

The input port selector of \( R_{tw} \) must change the input port so that multicast flits forwarded from \( R_{tw} \) can be stored in \( B_{gl} \). Therefore, when \( DR_{gnt} \) is sent to \( R_{tw} \), information on the initially allocated buffer and \( B_{gl} \) is updated in the DRI in the form of a table (buffer sharing table), and the input port selector changes from an idle state (in which all flits are resident in the correct input buffer) to the recovery state. The input port selector works like a crossbar switch controlled by a buffer sharing table in DRI. When the head flit of the initially allocated buffer is removed, the input port selector returns to the idle state. Fig. 5 shows an example of an input port selector operation in recovery state. The multicast flit that was to be stored in the initially allocated buffer is migrated into \( B_{gl} \) by input port selector operating in recovery state.

4) Flit Reordering

The branch operation is completed by forwarding multicast flits from \( R_{tw} \) to \( B_{gl} \). Finally, it is necessary to reorder the flits divided into the initially allocated buffer and \( B_{gl} \) in \( R_{tw} \). After the multicast flits in the initially allocated buffer are granted access to the output port, flit reordering is started if empty or new header flits are detected. The row in the allocation table corresponding to the initially allocated buffer can be migrated to the row corresponding to \( B_{gl} \) with the buffer sharing table. Therefore, the deadlock recovery procedure is completed when the divided multicast flit is reordered, then the tail flit of the multicast packet is removed from \( B_{gl} \). The overall recovery time of MRBS does not exceed 15 cycles, even in the worst-case setup procedure of \( B_{gl} \). When considering the scarce frequency of the multicast deadlocks, such latency increase can be ignored. Algorithm 2 presents the above process of \( R_{tw} \) operation of the MRBS.

**Algorithm 1. Deadlock recovery procedure in \( R_{tw} \)**

| Input: \( TS, CF, TY, A \) |
|---------------------------|
| Output: \( DR_{req} \) |

1: \( \text{for } i = 0 \text{ to } N_p - 1 \) do
2: \( \text{initialize } DR_{req}[i] \) as 0
3: \( \text{if } ((TS[i] == 1) \&\& \text{condition 1} \&\& \text{condition 2}) \) then
4: \( \text{for } j = 0 \text{ to } N_p - 1 \) do
5: \( DR[i][j] = A[i][j] \& CF[i] \)
6: end for
7: \( \text{if } (TY[i] == \text{TAIL}) \) then
8: \( \text{reset } i^{th} \text{ row of } DR \)
9: else maintain \( i^{th} \text{ row of } DR \)
10: end if
11: end if
12: \( \text{for } k = 0 \text{ to } N_p - 1 \) do
13: \( DR_{req}[i] = DR_{req}[i] \| DR_T[i][k] \)
14: end for
15: end for

**Algorithm 2. Deadlock recovery procedure in \( R_{tw} \)**

| Input: \( DR_{req}, \text{Done}, IB, TY, A \) |
|---------------------------|
| Output: \( DR_{gnt} \) |

1: \( \text{for } i = 0 \text{ to } N_p - 1 \) do
2: \( \text{if } (DR_{req}[i] == 1) \) then
3: \( \text{Choose one of the input buffers } IB[k] \) except \( IB[i] \)
4: \( \text{Send } Rdg \text{ to upstream router of } IB[k] \)
5: \( \text{if } (\text{Done}) \) then
6: \( DR_{gnt}[i] = DR_{req}[i] \)
7: \( \text{Update } IB[i], IB[k] \text{ to buffer sharing table} \)
8: end if
9: end if
10: \( \text{while } (DR_{gnt}[i] == 1) \) do
11: \( \text{if } (TY[i] \text{ changes from BODY to HEAD or EMPTY}) \) then
12: \( \text{Change } i^{th} \text{ and } k^{th} \text{ row of matrix } A \)
13: end if
14: end while
15: end for

**Fig. 5. Input port selector operation in recovery state**

**Fig. 6.** An example of a multicast deadlock configuration that occurs in XY multicast routing. \( R_{tw} \), where the branch operation of packet \( B \) occurs, becomes \( R_{tw} \), and \( R_{tw} \) becomes \( R_{tw} \). \( B_{gl} \) of \( R_{tw} \) is determined as the local input buffer, and multicast flits are then forwarded from \( R_{tw} \). Because the branch operation of packet \( B \) is completed, the branch operation of packet \( A \) becomes possible, and thus, the multicast deadlock is broken. Finally, the multicast deadlock recovery procedure is completed by reordering of packet \( B \) in \( R_{tw} \).
IV. EVALUATION

A. SIMULATION SETUP

To validate the effectiveness of MRBS, the increase in the average latency relative to the decreased area of the MRBS is analyzed. A baseline conventional tree-based router (CTR) and MRBS were modeled using SystemVerilog [26] at register-transfer-level. We used Synopsys VCS, a commonly used tool for SoC design in industry and academics, to perform cycle-accurate simulation. CTR was modeled by exploiting stalled process, branch operation, and bit-string encoding packet format of state-of-the-art tree-based multicast routers [6], [11], [12], [13], [14].

With the widely accepted 2D mesh NoC topologies, we compare the latency performance of the CTR- and MRBS-based NoCs. The simulation was conducted by increasing the network size from $4 \times 4$ to $10 \times 10$ to analyze the latency and area tendencies according to node size. A widely known deterministic XY routing is commonly applied to exclude the effect of the routing algorithm.

The configurations of NoC and traffic are listed in Table 2. In terms of synthetic traffic patterns, we evaluated uniform random traffic. The injected traffic consists of two packets to mimic realistic system scenarios: two-flit short packets (such as request packets in a CMP) and longer 10-flit packets (such as response packets carrying a cache line). We assume a bimodal distribution of packets, with 80% of the packets being short and the rest being long, according to [27]. In addition, guided by prior research [28] that reported real-world multicast percentages, we investigated two different scenarios: (a) 5% multicast traffic (low multicast intensity) and (b) 30% multicast traffic (heavy multicast intensity). For both scenarios, we assume that each multicast packet is sent to a maximum of 25% of the network nodes, in line with what has been observed in real applications.

For the experiments with cache-coherence traffic, we injected traffic that mimics the behavior of (c) HyperTransport [29] (a directory-based protocol), and (d) Token Coherence [30] (a snooping-based), in a 64-node ($8 \times 8$) CMP. Broadcast traffic makes for a large portion of the injected traffic in the HyperTransport and Token Coherence protocols, accounting for 6% and 15%, respectively [27].

Because the CTR must guarantee to accommodate all 10 flits, which is the maximum packet length, each input channel has a buffer size of 10. On the other hand, since the MRBS embraces a buffer sharing scheme, each input channel has a buffer size of 5 (half of the CTR) and includes a DRU and input port selector to support the proposed deadlock recovery process. As mentioned in section III.B, $T_{\text{thres}}$ (50 cycles) is determined in consideration of the worst-case contention in which all the maximum lengths of packet simultaneously request one output port.

Each router was synthesized with Synopsys Design Compiler using the SAED 32nm standard cell to extract the timing and area information. We validated the average latency according to the traffic and area overhead of the router in the NoC configurations using each router. We validated
TABLE 2. Configuration of NoC and injected packets

| Characteristic             | CTR   | MRBS |
|----------------------------|-------|------|
| Network size               | 4 × 4 – 10 × 10 |      |
| Buffer depth (flits)       | 10    | 5    |
| Flit width (bits)          | 128   |      |
| Packet length              | 1 head flit (Body: Long: 8 flits, Short: 0 flit) | 1 tail flit |
| Short packet ratio         | 80%   |      |
| Multicast ratio            | 5 % (low intensity) | 30 % (high intensity) |
| Broadcast ratio            | 6 % (Hypertransport) | 15 % (Token Coherence) |
| Multicast destination count| Maximum 25 % of all communication nodes |      |
| Operation frequency        | 0.5GHz|      |

the improvement in the MRBS by deriving the area-delay product (ADP), considering the trade-off between them.

B. LATENCY ANALYSIS

Table 3 shows the saturation point of the CTR-based NoC at different network sizes and average latencies of the CTR- and MRBS-based NoCs at the corresponding injection rate. Saturation point indicates the point at which the latency is twice the zero-load latency [3], where the zero-load latency refers to the latency required to send a packet from its source node to all of its destination nodes with no other packets in the network.

The saturation point of the MRBS-based NoC was approximately 3 % smaller than that of the CTR-based NoC on average. The fact that there is a negligible difference in the saturation point considering the design with half of the buffer size indicates that the actual multicast traffic was not efficiently utilized in the existing CTR-based NoC. For a fair comparison, the average latency at the same injection rate was measured, and the injection rate was set as the saturation point of the CTR-based NoC.

Fig. 7 depicts the average latency in synthetic uniform random traffic. At the multicast intensities of 5 % and 30 %, the average latency of MRBS-based NoC had 5.0 % and 5.5 % overhead compared to CTR-based NoC, respectively. Under cache-coherence traffic, the average latency of MRBS-based NoC increased by 6.2 % and 7.1 % than that of CTR-based NoC, respectively, as shown in Fig. 8. Owing to the permissible difference in latency, the router area reduction of MRBS-based NoC significantly affects the ADP.

Since the deadlock recovery of MRBS is activated based on the time-out of the blocked multicast packet, it is also triggered by long-time contention. Flit forwarding (Step 2 in Section III.B) makes the input buffers occupied by the multicast packet in $R_{tr}$ available and resolves the contention of packets. This contention resolution mostly compensates for the average latency increase due to the reduced input buffer size of MRBS.

FIGURE 7. Network performance results for CTR- and MRBS-based NoCs under synthetic uniform random traffic with (a) 5 % and (b) 30 % multicast traffic intensity
FIGURE 8. Network performance results for CTR- and MRBS-based NoCs under traffic that mimics the behavior of (c) HyperTransport, and (d) Token coherence traffic in an 8x8 2D mesh.

C. ROUTER AREA AND POWER ANALYSIS

Table 4 lists the MRBS and CTR areas. Compared to the CTR, which can hold 10 flits in each input buffer, MRBS, whose capacity is up to 5 flits, has an input buffer with a 51366 $\text{um}^2$ smaller area. It also uses route computation (RC) logic, a switch allocator (SA), and a crossbar switch (X-bar) with a 12368 $\text{um}^2$ smaller area. This is mainly due to SA for credit control of the downstream router. The added DRU and input port selector consume an area of 1374 $\text{um}^2$. As a result, the total area of the MRBS is 42.1 $\%$ smaller than that of the CTR. This result shows that the area overhead owing to the DRU and input port selector is negligible because the buffer size occupies a significant portion of the total area.

| Area (um$^2$)                  | CTR    | MRBS    |
|--------------------------------|--------|---------|
| RC / SA / X-bar                | 45,461 | 33,093  |
| Input buffer                   | 102,732| 51,366  |
| DRU / Input port selector      | -      | 1,374   |
| Total area                     | 148,193| 85,833  |

The power consumption of CTR and MRBS is shown in Table 5. MRBS shows 41.0 $\%$ less power consumption than CTR. The input buffer in MRBS is half the size of that in CTR, which results in a 40.6 $\%$ reduction in register power. Minor credit control logic due to SA in MRBS reduces the power dissipated by 42.3 $\%$. On the other hand, additional power consumption by the DRU is only 0.6 $\%$.

| Power (uW)                  | CTR    | MRBS    |
|-----------------------------|--------|---------|
| RC / SA / X-bar             | 75,455 | 43,511  |
| Input buffer                | 152,690| 90,671  |
| DRU / Input port selector   | -      | 458     |
| Total area                  | 228,145| 134,640 |

D. AREA-DELAY PRODUCT

ADP is a performance index for evaluating an NoC when considering the trade-off between the average latency of data transmission and area overhead [31]. Fig. 9 shows the normalized ADP of the MRBS-based and CTR-based NoCs for different multicast traffic ratios. The MRBS-based NoC shows a 4.5 $\%$ (low multicast intensity) and 6.1 $\%$ (high multicast intensity) higher average latency than CTR-based NoC. In contrast, the area of the MRBS-based NoC is 42.1 $\%$ lower than that of the CTR-based NoC, regardless of network size. As a result, MRBS-based NoC has 39.3 $\%$ (a) and 38.3 $\%$ (b) lower ADP than a CTR-based NoC.

E. ENERGY-DELAY PRODUCT

We performed additional simulations to find the energy-delay product (EDP) of networks based on CTR and MRBS at various network sizes. The switching activity information (SAIF) file was extracted from the average delay simulation result containing toggle counts and retention time about the signal. The use of SAIF file for signal transition-aware simulation enables more precise power estimation. The EDP comparison between MRBS-based and CTR-based NoC is depicted in Fig. 10. The EDPs of the MRBS-based approach are (a) 28.5 $\%$ and (b) 26.6 $\%$ lower than that of the CTR-based approach in 8x8 2D mesh with uniform-random multicast traffic, respectively.
proved that MRBS-based NoC is more scalable than CTR-tree-based router. Simulation with various network sizes lower ADP while sustaining performance than a conventional multicast routers. MRBS achieves a 39.3 % area-efficient and scalable.

**REFERENCES**

[1] T. Bjerregaard and S. Mahadevan, “A survey of research and practices of Network-on-chip”. ACM Computing Surveys, vol. 38, no. 1, pp. 1–51, Jun. 2006.

[2] S. Ma, N. E. Jerger, and Z. Wang, “Supporting efficient collective communication in NoCs,” in IEEE International Symposium on High-Performance Comp Architecture, Feb. 2012, pp. 1–12.

[3] N. E. Jerger, L.-S. Peh, and M. Lipasti, “Virtual circuit tree multicasting: A case for on-chip hardware multicast support”, in International Symposium on Computer Architecture, Jun. 2008, pp. 229–240.

[4] W. Hu, Z. Lu, A. Jantsch, and H. Liu, “Power-efficient tree-based multicast support for networks-on-chip,” in Asia and South Pacific Design Automation Conference (ASPDAC), Jan 2011, pp.363–368.

[5] K. Bhardwaj, W. Jiang, and S. M. Nowick, “Achieving lightweight multicast in asynchronous nos using a continuous-time multi-way read buffer,” in IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Oct. 2017, pp. 1–8.

[6] C. Xiao, H. Lou, C. Li, and K. Jin, “DBM: A Dimension-Bubble-Based Multicast Routing Algorithm for 2D Mesh Network-on-Chips,” in Conference on Advanced Computer Architecture, Sep. 2020, pp. 43–55.

[7] A. T. Tran and B. M. Baas, “DLABS: A dual-lane buffer-sharing router architecture for networks on chip,” IEEE Workshop On Signal Processing Systems, Oct. 2010, pp. 327–332.

[8] S. Velayudham, S. Rajagopal, S. Kathirvel, and B. Alhadidi, “An Overview of Multicast Routing Algorithms in Network on Chip,” in International Conference on Innovative Computing and Cutting-edge Technologies, Apr. 2021, pp. 163–178.

[9] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, “HAMUM – A novel routing protocol for unicast and multicast traffic in MPSoCs,” in Euromicro Conference on Parallel, Distributed and Network-based Processing, Feb. 2010, pp. 525–532.

[10] D. R. Kumar, W. A. Najjar and P. K. Srimani, “A new adaptive hardware tree-based multicast routing in k-ary n-cubes,” IEEE Transactions on Computers, vol. 50, no. 7, pp. 647–659, Jul. 2001.

[11] L. Wang, Y. Jin, H. Kim and E. J. Kim, “Recursive partitioning multicast: A bandwidth-efficient routing for Networks-on-Chip,” in ACM/IEEE International Symposium on Networks-on-Chip, May. 2009, pp. 64–73.

[12] Z. Ming, et al., “An improved minimal multicast routing algorithm for mesh-based networks-on-chip,” in IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Aug. 2014, pp. 775–779.

[13] Z. Wang, H. Gu, Y. Yang, H. Zhang, and Y. Chen, “An adaptive partition-based multicast routing scheme for mesh-based networks-on-chip,” Computers & Electrical Engineering, vol. 51, no. 1, pp. 235–251, Apr. 2016.

[14] C. Wu, K. Lee and A. P. Su, “A Hybrid Multicast Routing Approach with Enhanced Methods for Mesh-Based Networks-on-Chip,” IEEE Transactions on Computers, vol.67, no.9, pp. 1231–1245, Mar. 2018.

[15] S. Gaur, V. Laxmi, M. Zolotinskii, M. Kumar, N. Gupta and Ashish, “Network-on-chip: Current issues and challenges,” in International Symposium on VLSI Design and Test, Jun. 2015, pp. 1–3.

[16] E. G. Coffman, M. Elphick, and A. Shoshani, “System deadlocks,” ACM Computing Surveys, vol. 3, no. 2, pp. 67–78, Jun. 1971.

[17] M. Karol, M. Hluchyj, and S. Morgan, “Input Versus Output Queueing on Networks-on-Chip,” in ACM/IEEE International Symposium on Networks-on-Chip (NOCS), Feb. 2010, pp. 1–8.

[18] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, “HAMUM – A novel routing protocol for unicast and multicast traffic in MPSoCs,” in Euromicro Conference on Parallel, Distributed and Network-based Processing, Feb. 2010, pp. 525–532.

[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 Transactions on Computers, 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 ACM/IEEE International Symposium on Networks-on-Chip, May. 2009, pp. 64–73.

[21] Z. Ming, et al., “An improved minimal multicast routing algorithm for mesh-based networks-on-chip,” in IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Aug. 2014, pp. 775–779.

[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,” Computers & Electrical Engineering, vol. 51, no. 1, pp. 235–251, Apr. 2016.

[23] C. Wu, K. Lee and A. P. Su, “A Hybrid Multicast Routing Approach with Enhanced Methods for Mesh-Based Networks-on-Chip,” IEEE Transactions on Computers, vol.67, no.9, pp. 1231–1245, Mar. 2018.

[24] S. Gaur, V. Laxmi, M. Zolotinskii, M. Kumar, N. Gupta and Ashish, “Network-on-chip: Current issues and challenges,” in International Symposium on VLSI Design and Test, Jun. 2015, pp. 1–3

[25] E. G. Coffman, M. Elphick, and A. Shoshani, “System deadlocks,” ACM Computing Surveys, vol. 3, no. 2, pp. 67–78, Jun. 1971.

[26] M. Karol, M. Hluchyj, and S. Morgan, “Input Versus Output Queueing on Networks-on-Chip,” in ACM/IEEE International Symposium on Networks-on-Chip (NOCS), Feb. 2010, pp. 1–8.

[27] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, “HAMUM – A novel routing protocol for unicast and multicast traffic in MPSoCs,” in Euromicro Conference on Parallel, Distributed and Network-based Processing, Feb. 2010, pp. 525–532.

[28] D. R. Kumar, W. A. Najjar and P. K. Srimani, “A new adaptive hardware tree-based multicast routing in k-ary n-cubes,” IEEE Transactions on Computers, vol. 50, no. 7, pp. 647–659, Jul. 2001.

[29] L. Wang, Y. Jin, H. Kim and E. J. Kim, “Recursive partitioning multicast: A bandwidth-efficient routing for Networks-on-Chip,” in ACM/IEEE International Symposium on Networks-on-Chip, May. 2009, pp. 64–73.

[30] Z. Ming, et al., “An improved minimal multicast routing algorithm for mesh-based networks-on-chip,” in IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Aug. 2014, pp. 775–779.

[31] Z. Wang, H. Gu, Y. Yang, H. Zhang, and Y. Chen, “An adaptive partition-based multicast routing scheme for mesh-based networks-on-chip,” Computers & Electrical Engineering, vol. 51, no. 1, pp. 235–251, Apr. 2016.

[32] C. Wu, K. Lee and A. P. Su, “A Hybrid Multicast Routing Approach with Enhanced Methods for Mesh-Based Networks-on-Chip,” IEEE Transactions on Computers, vol.67, no.9, pp. 1231–1245, Mar. 2018.

[33] S. Gaur, V. Laxmi, M. Zolotinskii, M. Kumar, N. Gupta and Ashish, “Network-on-chip: Current issues and challenges,” in International Symposium on VLSI Design and Test, Jun. 2015, pp. 1–3
MIN CHAE YANG (S’20) received the B.S. degree in electronic and engineering from Hongik University, Seoul, South Korea, in 2020. He is currently pursuing the M.S. degrees in artificial intelligence from Sungkyunkwan University, Suwon, South Korea. His research interests include NoC, machine learning, and computer architecture.

YOUNG SIK LEE (S’16) received the B.S. degree in electronic and electrical engineering from Sungkyunkwan University, Suwon, South Korea, in 2016, where he is currently pursuing the M.S. and Ph.D. degrees in electrical and computer engineering. His research interests include NoC, machine learning, and computer architecture.

TAE HEE HAN (M’08) received the B.S., M.S., and Ph.D. 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 architecture for artificial intelligence, emerging memory systems, and processing in memory.

VOLUME 4, 2016