Achieving Flexible Global Reconfiguration in NoCs using Reconfigurable Rings

Liang Wang, Leibo Liu, Jie Han, Xiaohang Wang, Shouyi Yin, Shaojun Wei

Abstract—The communication behaviors in NoCs of chip-multiprocessors exhibit great spatial and temporal variations, which introduce significant challenges for the reconfiguration in NoCs. Existing reconfigurable NoCs are still far from ideal reconfiguration scenarios, in which globally reconfigurable interconnects can be immediately reconfigured to provide bandwidths on demand for varying traffic flows. In this paper, we propose a hybrid NoC architecture that globally reconfigures the ring-based interconnect to adapt to the varying traffic flows with a high flexibility. The ring-based interconnect has the following advantages. First, it includes horizontal rings and vertical rings, which can be dynamically combined or split to provide low-latency channels for heavy traffic flows. Second, each combined ring connects a number of nodes, thereby improving both the utilization of each ring and the probability to reuse previous reconfigurable interconnects. Finally, the reconfiguration algorithm has a linear-time complexity and can be implemented using a low-overhead hardware design, making it possible to achieve a fast reconfiguration in NoCs. The experimental results show that compared to recent reconfigurable NoCs, the proposed NoC architecture can greatly improve the saturation throughput for synthetic traffic patterns, and reduce the packet latency over 40 percent for realistic benchmarks without incurring significant area and power overhead.

Index Terms—reconfiguration, NoCs, rings, traffic flows

1 Introduction

With an increasing number of cores integrated in chips, on-chip communication is becoming critically important to the system performance [1], [2]. Network-on-chip (NoC) is a widely used on-chip communication subsystem, providing higher bandwidth and lower latency than bus-based communication [3]. However, traditional router-based NoCs are statically provisioned without knowledge of application demands, leading to both poor communication performance and low utilization. In addition, the complex pipelines (e.g. 2$\sim$5 cycles per hop) of routers also limit the scalability of NoCs. Therefore, future NoCs are expected to be dynamically reconfigured according to communication demands [4].

The time-varying traffic behaviors in chip-multiprocessors (CMPs) bring new challenges for design of reconfigurable NoCs. Modern CMPs adopt NoCs as communication fabrics for cache-coherent messages (e.g., requests, invalidations, response data, etc.) in the shared last-level cache (LLC) [1]. Without sophisticated traffic characterization techniques [5], [6], the traffic behaviors in CMPs exhibit less-predictable pattern to the NoC. In other words, the traffic pattern is almost impossible to be predicted in a fine time granularity, although the traffic distribution can possibly be characterized on a long-term scale using machine learning algorithms and the distribution varies from phase to phase [5], [7], [8]. The less-predictable and quickly varying traffic patterns necessitate flexible reconfigurable NoC architectures that can provide fast and fine-grained adaptivity [9].

Prior reconfigurable NoCs are either locally reconfigured or globally reconfigured. Local reconfiguration approaches perform the reconfiguration on each router from microarchitecture level [2], [10], [11]. Router designs such as EVC [10] and SMART [2] set up multi-hop bypassing paths with cycle-by-cycle reconfiguration. However, they are agnostic of global traffic information, thereby achieving limited performance improvement. Global reconfiguration reconfigures the network topology by analyzing traffic behaviors, and allocates additional channels to heavy traffic flows [6], [12]–[14]. These approaches either assume constant traffic flows within each application [12], [13] or capture long-term communication distribution of chip-multiprocessors using machine learning algorithms [6], [14]. Because most of the reconfigurable architectures rely on software-based sophisticated algorithms for reconfiguration, they are not flexible enough to provide fast global reconfiguration for fine-grained adaptivity. In other words, they are not applicable for fine time granularity, in which the time-varying traffic patterns are less-predictable for reconfiguration. Therefore, both local reconfigurable NoCs and global reconfigurable NoCs are still far from ideal reconfiguration scenarios, in which globally reconfigurable interconnects can be flexibly reconfigured to provide bandwidths on demand for varying traffic flows in the network.

This paper presents a novel ring-based reconfigurable NoC that augments a router-based mesh network with a set of reconfigurable rings. A reconfiguration mechanism is proposed that dynamically combines or splits the rings such that any two nodes with high communication demands can be allocated a low-latency channel. Compared to traditionally augmented point-to-point channels, because

- *Liang Wang, Leibo Liu, Shouyi Yin and Shaojun Wei are with the Institute of Microelectronics, Tsinghua University, Beijing, China. Email: {lwang_cuhk, liulb, yinsy, wsj}@tsinghua.edu.cn*
- *Jie Han is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. Email: jhan8@ualberta.ca*
- *Xiaohang Wang is with the School of Software Engineering, South China University of Technology, and Guangzhou Institute of Advanced Technology, Guangzhou, China. Email: xiaohangwang@scut.edu.cn*
- **Leibo Liu is the corresponding author.*

each reconfigured ring connects a number of nodes, both the utilization of each ring and the probability to reuse previous interconnects are improved. Thus, it has a higher reconfiguration efficiency than point-to-point interconnects. Due to the simplicity of the proposed architecture, the reconfiguration algorithm has a linear-time complexity and is implemented using a low-overhead allocator in a distributed manner. For a 8×8 NoC, the reconfiguration algorithm takes at most 32 cycles. Compared to a router-based mesh NoC, the reconfigurable NoC greatly improves both the saturation throughput and zero-load latency with an additional 6.5% area overhead.

The novel contributions of this paper include:

(1) A reconfigurable NoC architecture is proposed that augments a router-based mesh network with a set of reconfigurable rings. The NoC architecture can be dynamically reconfigured to adapt traffic variation.

(2) A novel ring-based reconfigurable architecture is designed to utilize a set of horizontal rings and vertical rings as the interconnect, which can be dynamically combined or split according to current traffic demands. Since the reconfiguration architecture can both allocate additional bandwidth for heavy traffic flows on demand and improve the probability to reuse previous interconnects, the efficiency of the reconfiguration is maximized.

(3) A linear-time complexity reconfiguration algorithm and a low-overhead hardware implementation for the reconfiguration algorithm are proposed. The fast and flexible reconfiguration makes fine-time-granularity reconfiguration possible.

The remainder of this paper is organized as follows. Section 2 briefly introduces the related work. Section 3 discusses the characterization of realistic traffic. Section 4 presents the proposed hybrid architecture. Section 5 describes the reconfiguration algorithms and implementations. Section 6 presents the integration with the router-based network and deadlock avoidance. Section 7 analyzes the experimental results, and Section 8 concludes this paper.

2 Related Work

2.1 Local Reconfiguration

Previous studies have explored various microarchitecture approaches for reconfigurable NoCs. Circuit-switched fabrics have been proposed in [15], [16]. Compared to packet-switched networks, circuit-switched networks can greatly lower the packet latency. Hybrid-switched NoCs that intermingle packet-switched flits with circuit-switched flits are proposed in [11], [17]. However, circuit-switched reconfiguration techniques face high setup and tear-down latency, thereby they are more effective for those high-predictable traffic. In [18], Hesse et al. propose to replace traditional unidirectional channels with narrow bidirectional channels, which can dynamically change direction according to current bandwidth demands. Other researchers have proposed microarchitecture techniques to virtually bypass the router pipelines [2], [10]. In [2], the researchers present a single-cycle multi-hop asynchronous repeated traversal (SMART) flow-control mechanism that can reconfigure the multi-hop paths according to current requests from packets. Once the multi-hop path is set up, the packets can traverse the multi-hop path within one cycle. Although the zero-load latency can be effectively reduced, the length of the bypassing paths is usually limited at high traffic load because the bypassing paths are locally reconfigured without global knowledge of traffic. Runahead [19] is a multi-noc design that augments a buffered NoC with a bufferless NoC. The bufferless NoC is intrinsically locally reconfigurable and achieves a very low zero-load latency.

2.2 Global Reconfiguration

To insert global channels to bypass the intermediate routers, small-world NoCs have been proposed to insert long-range wired or wireless links to provide low-latency communication channels for distant nodes in mesh NoCs [20], [21]. The long-range insertion is formulated as optimization problems to minimize packet latency. However, those long-range links are statically inserted.

To dynamically adapt to the traffic variations, global reconfiguration approaches have been extensively studied. In [12], Modarressi et al. propose an application-aware NoC that achieves the reconfiguration by inserting several simple switches. A branch and bound algorithm is developed to choose bypassing paths for heavy traffic flows with the aim to reduce the communication latency. In [13], Xue et al. propose a mathematical framework for reconfiguration of NoC. To solve the problem, an optimization problem is formulated based on reconfiguration using circuit-switching, and a greedy algorithm is proposed to solve this problem. In [6], [14], the researchers discover that the communications are dominated by some traffic flows. By analyzing the communication behaviors using machine learning algorithms, the researchers propose to augment the mesh network with different reconfigurable interconnects, which provide globally reconfigurable channels for a few dominant traffic flows. Three reconfiguration algorithms named Flow-Greedy (FG), No-Split (NS) and Latency-Optimized (LO) in [6] are presented. Although No-Split (NS) has the lowest complexity, it achieves the poorest reconfiguration efficiency. The other two reconfiguration algorithms have much higher complexities, thereby they are not feasible for fine-time-granularity reconfiguration. Moreover, the augmented channels in [6] only connect nodes in the same row or the same column. Therefore, it cannot provide global channels for any source-destination pairs on demand. The reconfiguration algorithm in [14] is more complicated, which first sort all traffic flows and then allocate the channels according to a search tree-based heuristic. It is reported that the running time of the reconfiguration algorithm in a 8×8 NoC is in the order of 10^5 cycles. Moreover, the reconfigured channels are point-to-point, i.e., they are only dedicated for a few pairs of nodes, achieving limited reconfiguration efficiency for less-predictable traffic patterns.

2.3 Ring Interconnects

Ring interconnects [22]–[25] that only adopt rings to connect all processing nodes suffer from scalability issues. In [22], [25], researchers have proposed hierarchical ring NoCs, in which locals rings are connected by global rings. However, the switching between the local rings and the global rings limits the scalability. In [23], [24], researchers have proposed multiple overlapped rings. The basic idea is to select a set of smartly placed loops such that any pairs of cores are connected via at least one isolated ring, and each packet can reach the destination without transferring from one ring to another. Although the multi-ring NoCs show better performance and scalability than hierarchical rings [22], there would be a large number of overlapped rings for a large network. The maximum size of the network is also limited by the overlap cap (which is 16 in [24]). Therefore, the multi-ring NoCs still have limited scalability.

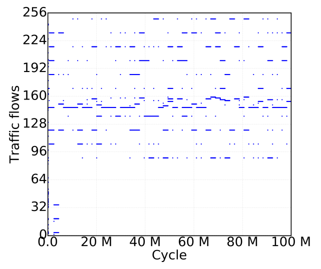
The proposed hybrid NoC architecture in this paper is based on two observations. First, since a ring interface has a much lower complexity than a router, it is easier to reconfigure the ring-based interconnects to accommodate to traffic variations. Second, ring-based interconnects can provide low latency (e.g., 1 cycle per hop) and they have very low area and power overhead. However, the disadvantages of ring-based architectures are that (1) it still has poor scalability if only rings are used and that (2) it has lower bandwidth than the router-based NoCs with many virtual channels. Therefore, a hybrid NoC that combines rings and a router-based mesh network is presented.

3 Characterization of Realistic Traffic

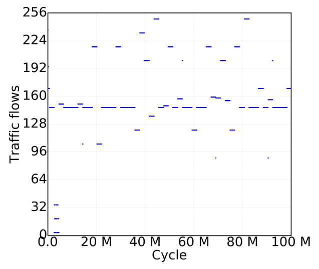
To design a reconfigurable interconnect for CMPs, we first conduct a case study to analyze the communication behaviors of realistic benchmarks *blackscholes* and *bodytrack* in a 4×4 CMP. The communication traces are extracted from gem5 using *simdev* set by simulating the Region-of-Interest (ROI) for $100M$ cycles ($1M$=1 million). Figure 1 presents scatter plots for the distributions of dominant traffic flows for benchmarks *blackscholes* and *bodytrack*. The most dominant traffic flow within each time interval is extracted. The time intervals are set 1000 cycles and 100000 cycles respectively. There are 256 source-destination pairs, i.e., 256 traffic flows in total. In the figure, the numbers for the traffic flow represent source-destination pairs. For example, the numbers 16-31 represent all traffic flows originate from node 1, and the numbers 240-255 represent all traffic follows originate from node 15.

Figure 1 shows that the communication behaviors exhibit great spatial and temporal variations. From Figure 1(a) and Figure 1(c), it can be seen that the dominant traffic flow varies significantly when the time interval is 1000 cycles. In contrast, when the time interval is 100000 cycles, Figure 1(b) and Figure 1(d) show that the dominant flows are relatively stable, which also proves that the communication behaviors can be characterized on a long-term scale. However, the fine-grained traffic variation cannot be captured. In a finer time granularity, the traffic behaviors show less-predictable, bringing significant challenges for the design of reconfigurable NoCs.

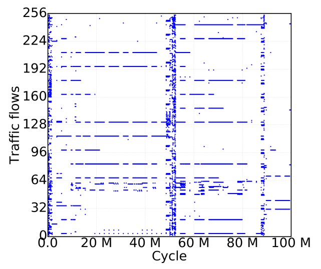
Motivated by the less-predictable communication behaviors, we design a globally reconfigurable NoC, which can achieve quick reconfiguration for the less-predictable traffic patterns. Our design methodology is orthogonal to those methods with complicated traffic prediction techniques. Instead, we try to design a reconfigurable architecture that is more appropriate for the less-predictable traffic.



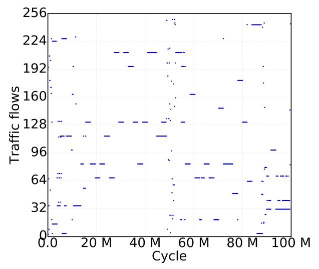
(a) blackscholes, 1000 cycles each interval



(b) blackscholes, 100000 cycles each interval



(c) bodytrack, 1000 cycles each interval



(d) bodytrack, 100000 cycles each interval

Fig. 1. Temporal distributions of dominant traffic flows with each time interval for benchmarks *blackscholes* and *bodytrack*. The time intervals are chosen 1000 and 100000 cycles respectively.

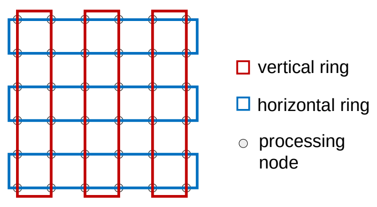
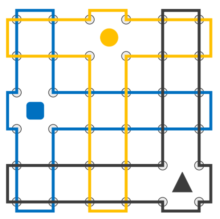


Fig. 2. The ring network includes two set of rings: horizontal rings and vertical rings.

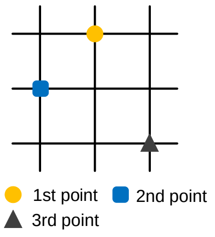
4 Ring-Based Reconfigurable Architecture

4.1 Basic Architecture of the Rings

In this paper, a hybrid network-on-chip is proposed, which augments a router-based mesh network with a ring-based network. The architecture of the ring-based network is presented in Figure 2. The ring-based network, integrated with a $N \times N$ router-based mesh network, is composed of two sets of rings, which include horizontal rings and vertical rings. The nodes located at the $2i$-th and the $(2i+1)$-th columns (or rows) are connected by a single ring ($i \in [0, N/2-1]$, $i \in \mathbb{Z}$). Therefore, for 6×6 nodes as the example in Figure 2, there are 3 horizontal rings and 3 vertical rings. Each node can use either the vertical ring or the horizontal ring to transmit packets. Because there is no buffer for packet stalling in the rings, the ring-based network provides single-cycle per hop transmission due to the simplicity of the ring interface. The nodes can still use the router-based mesh network for packet transmission. The integration with router-based network is detailed in Section 6. It is assumed that only NoCs with even and equal numbers of rows and columns are supported.



(a) The rings are combined by reconfiguration. Each horizontal ring and each vertical ring are exclusively combined.



(b) An example of the reconfiguration points that determine how the rings are combined.

Fig. 3. Reconfiguration of the rings

TABLE 1
Descriptions of the feature vectors

Feature vector	# of features	Description
PFlow	$N^2 \times N^2$	Packet injected between each source-destination pair
RCFlow	$N \times N$	Packet injected between rows and columns of the network
DRCFlow	$\frac{N}{2} \times \frac{N}{2}$	Packet injected between the $2i$-th, $(2i+1)$-th rows and the $2j$-th, $(2j+1)$-th columns

For a globally reconfigurable NoC, an important feature is that any two nodes can be provided with additional low-latency channels when needed. To support this feature, the proposed NoC can perform reconfiguration by combining or splitting the rings according to current communication demands of the on-chip traffic. As shown in Figure 3(a), when two rings are combined into a single ring, the nodes on the combined ring can directly communicate through the low-latency ring. If there is no available ring between two nodes, the packets can still be transmitted through the router-based NoC. To determine how the rings are combined, **reconfiguration points** are defined as shown in Figure 3(b). In the example, the second point indicates that the first vertical ring and the second horizontal ring are combined into a single ring. Through reconfiguration, the reconfiguration points can dynamically change according to the runtime communication requirements. For each set of reconfiguration points, each horizontal ring and each vertical ring are exclusively combined. This proposed ring-based architecture mainly has the following 3 advantages,

- The ring-based reconfigurable network can accommodate to the communication demands because any two processing nodes with heavy traffic flows can be possibly allocated a combined ring by determining a specific set of reconfiguration points.
- The allocation of the reconfiguration points is similar as an allocation problem in a network switch (e.g., switch allocation in a router), which results in a low-overhead hardware-based design for the reconfiguration algorithm as presented in Section 5. The low-complexity reconfiguration algorithm makes it possible to achieve flexible reconfigurations.
- In contrast to point-to-point channels, each combined ring connects a number of nodes (e.g. 20 nodes in a 6×6 network) instead of a pair, improving the probability to reuse previous reconfigurable interconnects, which is detailed in Section 4.2.

4.2 Reconfiguration Efficiency Analysis of the Proposed Architecture

To analyze the reconfiguration efficiency, we first adopt 3 feature vectors named Per-node Flow (PFlow), Row-column Flow (RCFlow) and Double Row-column Flow (DRCFlow), which are presented in Table 1. PFlow [5], widely exploited by the point-to-point reconfigured channels in existing reconfiguration mechanisms, denotes the distribution of the flows between each source-destination pair. RCFlow [5], aggregating nodes into rows and columns, denotes distribution of the flows between rows and columns of the network. DRCFlow, extracted based on the proposed ring architecture, denotes the distribution of the flows that originate from the $2i$-th and the $(2i+1)$-th rows to the $2j$-th and the $(2j+1)$-th columns.

Generally, a reconfiguration mechanism collects traffic flow information from previous time intervals and reconfigures the network in the following interval accordingly. If the reconfigured interconnect in the previous time interval can be reused in the following time interval, the performance can be effectively improved. The proposed reconfigurable architecture, corresponding to DRCFlow, has a higher probability to reuse reconfigurable interconnects in the previous time interval because there are fewer possible features of DRCFlow. More specifically, if the destination of a traffic flow varies randomly from interval to interval, the source node has a probability of $2/N$ to reuse previously reconfigured combined rings as there are $N/2$ rings to combine. In contrast, for point-to-point interconnects, the source node only has a probability of $1/N^2$ to reuse previous interconnects. Since the proposed reconfigurable architecture has more chances to reuse the previous interconnects, it reduces the probability to reconfigure the network.

Moreover, each ring connects $2N$ nodes, i.e., any pairs of the $2N$ nodes can communicate through the ring. Thus, the utilization of each ring is also improved. As a result, a larger ring connects a larger set of nodes and has a higher utilization, but for each node the probability to reuse the previous interconnects is lower.

On the other hand, the proposed architecture narrows down the search space of the reconfiguration problem. The maximum search space of the reconfiguration problem for the proposed reconfigurable rings is $N^2/4$, while the maximum search search space for point-to-point interconnects is N^4. The much smaller search space makes it easier to design a low-complexity reconfiguration algorithm.

In summary, since each ring connects a number of nodes, the proposed reconfigurable rings improve the reconfiguration efficiency by (1) improving the probability to reuse previous interconnects, (2) improving the utilization of each ring, and (3) narrowing down the search space of the reconfiguration problem.

5 Reconfiguration

5.1 Problem Definition

The reconfiguration points can be used as knobs for the reconfiguration. Therefore, the problem is to determine the reconfiguration points according to the traffic demands at runtime with the objective to minimize the average packet latency. $f_{i,j}$ is used to denote the volume of the traffic flows that originate from the nodes in the i-th horizontal ring to the nodes in the j-th vertical ring. Obviously, the number of horizontal rings or vertical rings augmented on a $N \times N$ mesh network is $N/2$, which is denoted as R. The problem can be formulated as follows,

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{R}\sum_{j=1}^{R} \alpha_{i,j} f_{i,j} \\ &\text{subject to} && \sum_{i=1}^{R} \alpha_{i,j} = 1, \sum_{j=1}^{R} \alpha_{i,j} = 1, \alpha_{i,j} \in \{0,1\} \end{aligned} \tag{1}$$

where $\alpha_{i,j}$ indicates whether the reconfiguration point (i,j) is granted. If the i-th horizontal ring and the j-th vertical ring are combined, $\alpha_{i,j} = 1$. Otherwise, $\alpha_{i,j} = 0$. The constraints guarantee that each horizontal ring and each vertical ring are exclusively combined. The basic idea of the formulation is to improve the utilization of the rings by allocating more low-latency channels to the heavier traffic flows, which have higher $f_{i,j}$.

5.2 Reconfiguration Algorithm

As seen in Figure 3(b), the allocation of the reconfiguration points is similar to the allocation problems of switch allocation in a router or a network switch. We propose to solve the reconfiguration problem using a modified allocation algorithm with two differences from [26]: (1) each point[1] has a fixed priority instead of a random priority, and the priority depends on $f_{i,j}$; (2) multiple iterations of a separable allocation are performed, which is also similar as Parallel iterative matching (PIM) [3]. The proposed allocation algorithm is presented in Algorithm 1. The algorithm takes $f_{i,j}$ as the input, and outputs the allocation of the reconfiguration points. In each iteration, the allocation includes two-stage arbitrations. The first stage selects the unmatched reconfiguration point with the maximum $f_{i,j}$ in each row. The arbitration results of the first stage are sent to the second stage as requests. The second stage performs the arbitration in each column, where the grant is set to the reconfiguration point that wins the arbitration. Finally, the rows and columns of the granted reconfiguration points are set matched and will not be considered in the following iterations. After R iterations, all rows and all columns can guarantee to be matched. Figure 4 presents an example of the allocation for the reconfiguration points. In each iteration, the arbitrations are first performed in each row, and then performed in each column. After 2 iterations, all the rows and columns are matched. The allocation of the reconfiguration points is like a traditional allocation problem that multiple agents are contending for multiple resources simultaneously. The main difference is that the allocation is determined by $f_{i,j}$.

1. The point in this section indicates the candidate reconfiguration point.

Algorithm 1 Iterative allocation for the reconfiguration points

```
Definitions  $R$: number of horizontal rings or vertical rings.
Input  $f_{i,j}$: the number of communication requests from row
    $i$ to column $j$;
Output  $g_{i,j}$: the allocation of the reconfiguration points.
 1: for $R$ iterations do
 2:     for each unmatched row $i$ do  ▷ first stage, in parallel
 3:         Select the unmatched column $q$ with the maximum $f_{i,j}$
 4:         Set $r_{i,q} = 1$
 5:     end for
 6:     for each column $j$ do  ▷ second stage, in parallel
 7:         Select the row $p$ that is with the maximum $f_{i,j}$ and with $r_{i,j} = 1$
 8:         Set $g_{p,j} = 1$
 9:         Set row $i$ and column $j$ matched
10:     end for
11: end for
```

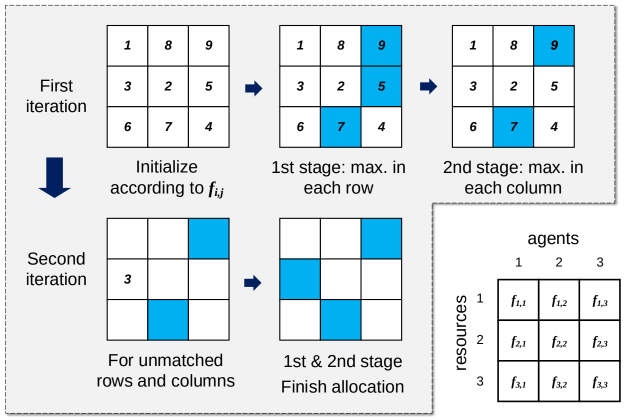

First iteration
Initialize according to $f_{i,j}$
1st stage: max. in each row
2nd stage: max. in each column
Second iteration
For unmatched rows and columns
1st & 2nd stage Finish allocation
agents
resources

Fig. 4. An example of allocation of reconfiguration points.

Analytically, the time complexity of Algorithm 1 is $O(2R^3)$. However, the arbitrations in each column or each row can be performed in parallel as annotated in Algorithm 1. Therefore, the time complexity of the algorithm can be reduced to $O(2R^2)$, which is a linear function with the total number of processing nodes $4R^2$. Because the algorithm is a greedy heuristic, the result of the algorithm is a suboptimal solution of the problem defined in Equation 1.

5.3 Distributed Reconfiguration Implementation

The algorithm can be implemented using a modified two-stage allocator that supports multi-iteration allocations. Figure 5 presents the architecture of the two-stage allocator. Each stage has R **R:1** arbiters. In contrast to traditional round-robin arbiters, a modified arbiter named **F arbiter** is designed, which selects the point with the maximum $f_{i,j}$ in each row or column. The design of the **F arbiter** is shown in Figure 6. In each arbiter, to select the point with the maximum f_i, **F arbiter** integrates multiple comparators. The point with a higher f_i sets $d_i = 1$ and unsets g_i of the previous points. en_i is used to enable this comparator, i.e., if this row or column has been matched in the previous iterations, the corresponding comparators are disabled.

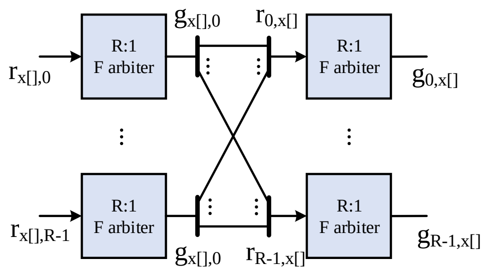


Fig. 5. The architecture of the allocator, which is composed of two-stage **F arbiters**. The first stage perform arbitrations among rows of the rings. The second stage perform arbitrations among columns of the rings.

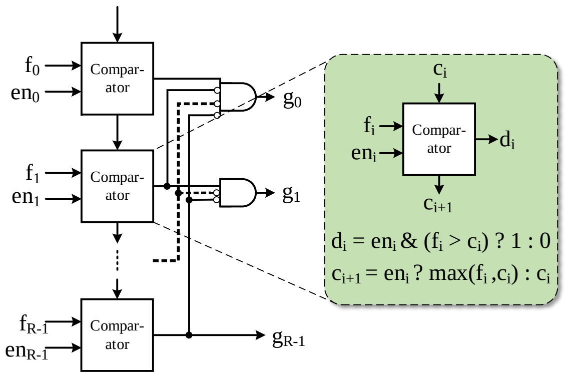


Fig. 6. The architecture of the proposed **F arbiter**. Each **F arbiter** is composed a set of comparators, which are distributed in each row or each column of the rings.

The comparators of each arbiter can be distributed to each point in this row or column. Therefore, the allocator can be implemented in a distributed manner. Assuming that the propagation delays of the signals within the arbiters are 1 cycle per row or per column, the delay per iteration is $2R$. Therefore, the maximum delay to obtain the solution is $2R^2$ or $N^2/2$, achieving a linear time complexity with the total number of processing nodes.

5.4 Routing Table Update

Each ring interface integrates a routing table that determines if a node can be reached by a ring and which path is the shortest. Each index of the routing table includes 3 bits (1) if a node can be reached by the reconfigured rings; (2) for which ring (horizontal or vertical) to select to reach a node; and (3) for which direction to use (clockwise or anticlockwise). Therefore, for a network with M processing nodes, the size of each routing table is $3M$ bits.

After allocating the reconfiguration points, each ring interface must update the routing table. To update the routing table using hardware, the routing table is initially set to 000 for all indices. Each node sends probe messages to 2 available directions (clockwise or anticlockwise direction in horizontal or vertical rings) via the connected rings. The probe message contains its sender and the hops it has traversed. If the probe message is received by its sender, then it is dropped. Otherwise, it continues to traverse along the same direction in the current ring. If node $\boldsymbol{a}$ and node $\boldsymbol{b}$ are connected by a combined ring, $\boldsymbol{b}$ can receive at least two probe messages from $\boldsymbol{a}$, and updates the routing table using the message whose hop count is smaller than half the length of the combined ring. Due to traversal of the probe messages, the routing table update requires an additional $4(N-1)$ cycles because the length of the combined ring is $4(N-1)$.

5.5 Analysis of Reconfiguration Time

The reconfiguration process includes 2 steps: reconfiguration algorithm and network reconfiguration.

Step 1: reconfiguration algorithm. This step takes at most $N^2/2$ cycles as detailed in Section 5.3. The running of the reconfiguration algorithm can be performed in parallel with the packet transmission in the rings.

Step 2: network reconfiguration. During this period, packets are not allowed to be injected through the rings. This step includes packet draining, routing table update and reconfiguration of ring interface. In the packet draining, all the packets in ring interconnects are drained to avoid the cases that some packets would never reach their destinations after reconfiguration. The packet draining takes $4(N-1)$ cycles. Since the packet draining is related to the ejection process, the packet draining time is further analyzed in Section 6.2. The routing table update also takes $4(N-1)$ cycles as presented in Section 5.4. Including 1 cycle for reconfiguration of ring interfaces, the total number of cycles for the network reconfiguration requires at most $(8N-7)$ cycles.

6 Integration with Router-Based Network

Each node is connected to a horizontal ring, a vertical ring and a router. A packet can either be injected to a ring or a router. The ring interface determines if the destination can be directly reached by a ring based on its routing table. If there is no direct ring to the destination or the specified output port of the ring is not available, then the packet chooses to use the router-based interconnect. Once a packet is injected to a ring or the router-based interconnect, it will reach the destination without switching between the two interconnects or switching between rings. This avoids both the bottleneck and high overhead of switching between the interconnects as the hierarchical rings [22].

6.1 Packet Injection

To minimize the average packet latency in the NoC, the packet injection process needs to select an appropriate interconnect for injection of each packet such that the packets can arrive at their destinations with a low latency. We propose a simple scheme for packet injection, which is presented as follows,

- If (1) the destination can be reached by a combined ring according to the routing table; and (2) in current cycle the designated port of the ring is available, i.e., there is no other packet passing by, then the packet is sent through the ring-based network.
- Otherwise, the packet is sent by the router-based mesh network.

For injection of multi-flit packets through the ring interface, a possible case is that a long packet is taking several cycles for injection while another packet arrives, and they are contending for the same output port. Because there is no flow control in the ring interface, we adopt a packet-size buffer (maximal packet size) to store the arriving packet temporarily which is also similar as the Extension Buffer Technique in [24].

6.2 Arbitration for Ejection

When a packet from the router-based interconnect or a ring arrives at its destination node, it is ejected by a router or a ring interface. However, a potential problem is that there are multiple flits from different interconnects to be ejected simultaneously but only one flit is allowed to be ejected at each cycle. Because multiple ejection links incur a high hardware overhead and the possibility of actual conflict is very low, we only adopt one ejection link and design an arbitration scheme for the ejection.

For the ejection, each node has 3 packet-size buffers to store the arrived packets. One is for the router, and the other 2 are for the 2 rings. Similar as that in [23], we adopt the Oldest-First mechanism [27] for the arbitration of the packets in the ejection buffers. In the Oldest-First mechanism, a time stamp is stored in the header of a packet to record the packet generation time. Thus, a packet with an older generation time has a higher priority. If the packets from the routers cannot be proceeded to their ejection buffers, they will be stalled in the input buffers of routers. If the packets from the rings cannot proceed to their ejection buffers, they will be deflected to the next node. The deflected packets will continue on their current rings and circle back to the destination later. The Oldest-First mechanism guarantees that a packet can finally arrive its destination, i.e., there is no livelock concern if there is no reconfiguration.

Packet draining: As previously mentioned, before reconfiguring the rings all the packets in ring interconnects are drained to avoid the cases that some packets would never reach their destinations after reconfiguration. To avoid possibly too long packet draining time in the reconfiguration due to the deflected packets, the arbitration scheme switches to a fixed-priority arbitration during packet draining. The packets from the rings are always prioritized over the packets from the router-based interconnect. This can minimize the number of deflections during packet draining. Moreover, the reconfiguration points are not allowed to be reconfigured if packet deflections happen within the packet draining time. Therefore, the maximum packet draining time is $4(N-1)$ cycles. The reason is that the maximum possible length of queue to be ejected is equal to the length of the combined ring.

Another advantage of the arbitration scheme is that the router-based network can still adopt 5-port routers instead of 6-port routers in [6], [14]. And a 6-port router would incur around 20% more area than a 5-port router. Therefore, by the proposed arbitration scheme for ejection, the additional rings does not incur too much area overhead for the router-based interconnect.

6.3 Deadlock Avoidance

Routing-induced deadlock will not happen because the router-based network adopts XY routing algorithm and the packets will not be stalled in the ring-based network.

Another possible deadlock is protocol-level deadlock in which the reply packets cannot be injected because the request packets are waiting to be ejected. Traditional router-based mesh network adopts separate virtual networks to avoid protocol deadlock. Since the proposed NoC architecture integrates rings with a router-based network, the protocol deadlock can be avoided by using 2 separate virtual networks in the router-based network. And the ring-based network can be used by all packets. Therefore, the cycle dependency of the protocol deadlock can be broken.

6.4 Hardware Implementations

The packets are injected to rings or ejected from rings through the ring interfaces. The design of the ring interface is shown in Figure 7. Most parts of the rings interface are similar as the rings interfaces in [23], [24] except the routing table, the ejection arbitration and the 2×2 switch.

To support reconfiguration, a 2×2 switch in each ring interface is integrated. As shown in Figure 7, each ring interface is connected to a bidirectional vertical ring and a horizontal ring. For simplicity, only clockwise rings are shown. If the current node is in a granted reconfiguration point, then the reconfiguration switch is enabled, i.e., the output ports of the horizontal ring and the vertical ring are exchanged. Each reconfiguration point corresponds to 4 ring interfaces with enabled switches.

Figure 8 shows that the pathways of the combined rings can be correctly set up by simply enabling the 4 reconfiguration switches in the reconfiguration point. If the reconfiguration point is granted, the reconfiguration switch is enabled. Then the two rings can be combined into one by exchanging the output ports of the vertical ring and the horizontal ring. The figure gives examples for different locations of the reconfiguration points. It can be seen from the figure that even the reconfiguration point located at the corner or the border of the network, the pathway can be correctly reconfigured just by simply enabling the switches of the corresponding 4 ring interfaces. Similarly, if the reconfiguration point located at the central part of the network, the pathway of the combined ring can also be correctly set up, which is not depicted in the figure. This nice feature also indicates that ring interconnects can be easily reconfigured without introducing complicated hardware logics for reconfiguration of the pathways.

7 Experimental Results

7.1 Experimental Setup

The evaluations are performed in the cycle-accurate GARNET NoC simulator [28]. The proposed NoC architecture is evaluated against the following 4 architectures: Baseline, SMART [2], Runahead [19], and S-channel [6].

Baseline is the basic 2D mesh NoC without any reconfigurations. The configurations of Baseline are presented in Table 2.

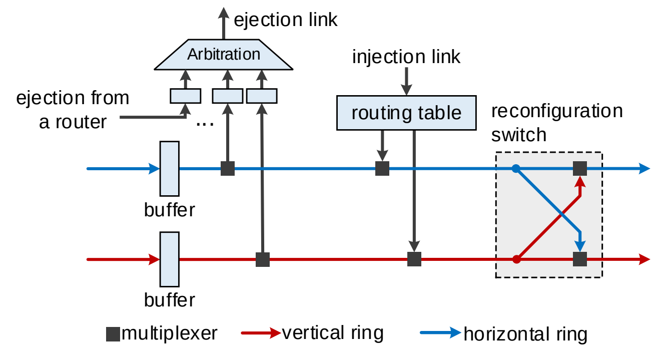


Fig. 7. Architecture of the ring interface. If the reconfiguration switch is enabled, the output ports of the horizontal ring and the vertical ring are exchanged.

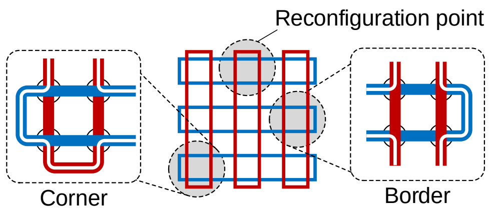


Fig. 8. The pathways within reconfiguration points that are in different locations of a mesh network. If a reconfiguration point is granted, the pathway can be correctly reconfigured by simply exchanging the output ports of the vertical ring and horizontal ring in each ring interface.

SMART [2] is a recent microarchitecture method that can adapt traffic variation by cycle-by-cycle reconfiguration. Once a bypass link is established, the packet can traverse multiple hops within one cycle. In the comparisons, SMART-8_$1D$ is adopted.

Runahead [19] is a multi-NoC architecture that augments a buffered network with a bufferless network. A packet can be sent by both the buffered network and the bufferless network.

S-channel [6] is a hybrid NoC that augments a mesh NoC with reconfigurable spanning channels. Although this is a software-based approach, it is still compared in the experiments because the complexity of the Non-Split (NS) algorithm is even as low as the proposed reconfiguration algorithm in this paper. The reconfigurable architectures in [12]–[14] are not compared because they rely on sophisticated reconfiguration algorithms, the running time of which is much longer than the time interval in this paper.

The basic configurations (without reconfiguration) for the routers in **SMART**, **Runahead** and **S-channel** are also the same as the Baseline router.

7.2 Packet Latency for Synthetic Traffic Patterns

The average packet latency of the network is evaluated using 4 synthetic traffic patterns: uniform, shuffle, transpose and hotspot. In the hotspot traffic pattern, there are 6 randomly inserted hotspots on a uniform traffic pattern. The time interval for reconfiguration is 1000. The results are presented in Figure 9. It shows that the proposed reconfigurable architecture, RR-Net can have obvious advantages in terms

TABLE 2
Configurations of the router-based mesh NoC

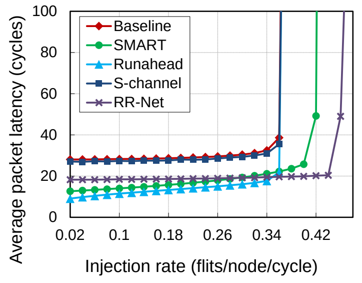
Network topology	8×8 or 16×16 2D mesh
Channel width	64
Router latency	3 cycles
Link latency	1 cycle
Number of VCs	8 (for synthetic traffic) or 2 (for benchmarks)
Input buffer	4-flit depth
Routing	XY routing algorithm
Benchmarks	PARSEC and SPLASH2 from Synful [5]

of saturation throughput for all the 4 synthetic traffic patterns. Especially for transpose and hotspot traffic patterns, the saturation throughput improvements over Baseline are over 85% and 100%, respectively. This result is because the proposed RR-Net adds additional channels for the traffic flows. And each combined ring in RR-Net connects at most 28 nodes in 8×8 network, the traffic flows have more chances to traverse in the low-latency ring interconnects.

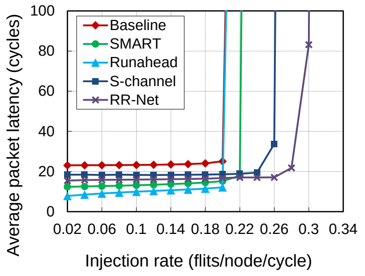
S-channel has a very limited throughput improvement because only a few pairs of nodes are connected by the augmented interconnects. Although the reconfiguration algorithm Non-Split (NS) also has a low complexity, S-channel only provides global reconfigurable interconnects for the nodes in the same row or the same column. It cannot provide global channels for any source-destination pairs on demand. Moreover, the point-to-point channels also make it not applicable to the less predictable traffic. Runahead achieves the best zero-load latency for all traffic patterns but almost no improvement on the saturation throughput. This result is because a packet can be sent by both buffered and bufferless network and the latency is calculated by the bufferless network. SMART achieves both a low zero-load latency and a high saturation throughput because it allows the packets to traverse multiple hops within a cycle. However, compared to RR-Net, SMART are agnostic to the global information of traffic flows, making it less effective in the high load for some traffic patterns.

Similar results can be observed Figure 10, which depicts the latency comparison in 16×16 NoC. The proposed RR-Net also results in around 116% and 100% saturation throughput improvement over Baseline for transpose and bit-reverse traffic patterns in 16×16 NoC. For other traffic patterns, RR-Net also leads to obvious improvement over other reconfigurable architectures in terms of saturation throughput. Figure 10 also shows that RR-Net is still effective for a larger NoC size. Therefore, RR-Net has a high scalability.

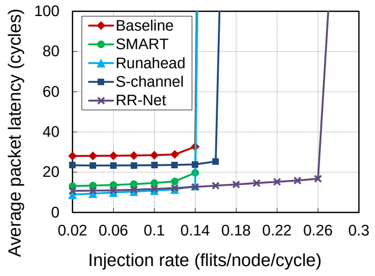
Although RR-Net achieves a high saturation throughput, the zero-load latency is not as low as SMART and Runahead for most of the synthetic traffic patterns. The reason is that the combined rings incur non-minimal paths on the rings for some packets. However, the negative effect of the non-minimal paths is limited because packet latency on the rings is 1 cycle per hop. In most cases the total latency of the non-minimal path is still lower than the latency in the router-based network which is 4 cycles per hop.



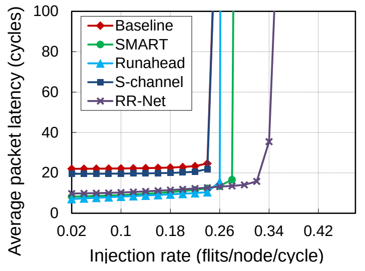
(a) Uniform traffic



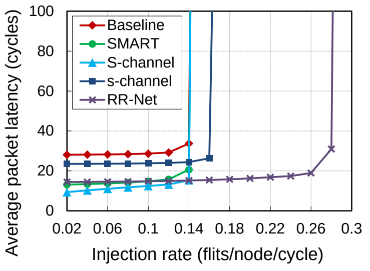
(b) Shuffle traffic



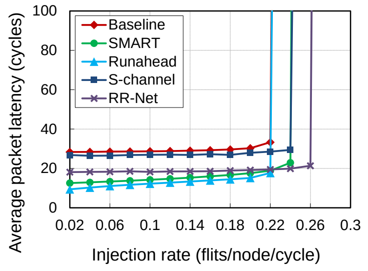
(c) Transpose traffic



(d) Tornado traffic

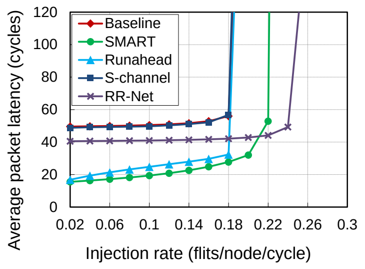


(e) Bit_reverse traffic

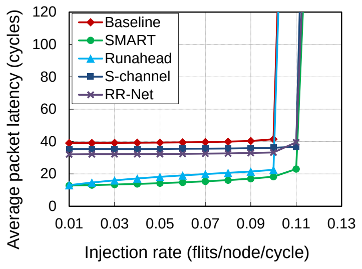


(f) Hotspot traffic. Hotspot nodes: (1, 19, 26, 31, 42, 44), percentage: 5%

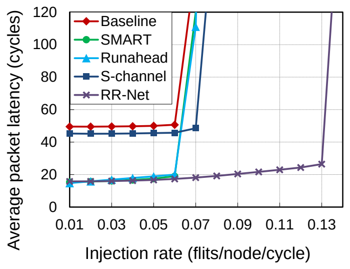
Fig. 9. Average packet latency for synthetic traffic patterns a 8×8 NoC



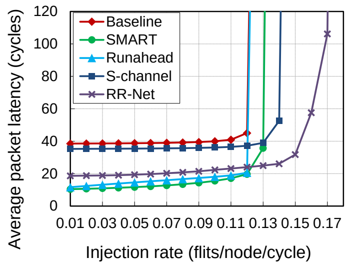
(a) Uniform traffic



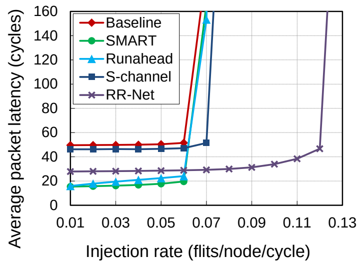
(b) Shuffle traffic



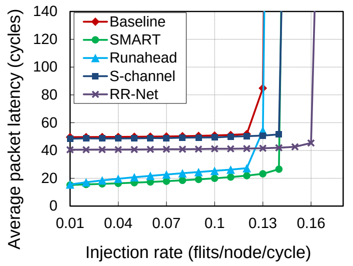
(c) Transpose traffic



(d) Tornado traffic



(e) Bit_reverse traffic



(f) Hotspot traffic. Hotspot nodes: (18, 43, 102, 177, 189, 231), percentage: 2%

Fig. 10. Average packet latency for synthetic traffic patterns in a 16×16 NoC

7.3 Analysis of Zero-Load Packet Latency

To further analyze the zero-load latency, we give a mathematical analysis for the zero-load latency of RR-Net. It is assumed that the communication pattern is uniform random.

The length of a combined ring is $L = 4(N-1)$, which is also equal to the number of nodes in each combined ring. The total distance (minimal distance) between node 0 and all other nodes can be calculated as follows,

$$\frac{L/2 \cdot (L/2+1)}{2} + \frac{L/2 \cdot (L/2-1)}{2} = \frac{L^2}{4} \qquad (2)$$

which is the total distance of $(L-1)$ pairs of nodes. The average distance of all the nodes in the combined ring can be calculated as

$$\frac{L^2}{4(L-1)} \approx \frac{L}{4} = N-1 \ \ (\text{if } L \text{ is large enough}) \qquad (3)$$

Since each hop in the ring takes 1 cycle, the average packet latency in the combined ring is around N cycles (there are N ring interfaces or routers between two nodes with distance $(N-1)$). In contrast, the average distance for a $N \times N$ mesh network is approaching $2N/3$ if N is very

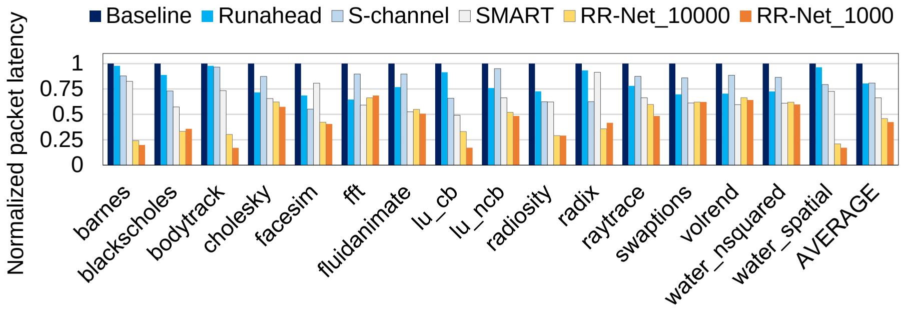


Fig. 11. Normalized packet latency for realistic benchmarks

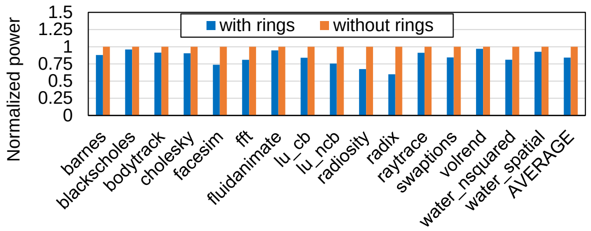


Fig. 12. Power Comparison

large[2]. If per-hop latency in the mesh network is 4 cycles, the average latency is around $8(N+1)/3$ cycles. It can be concluded that the average latency in the rings can also scale linearly with N as a mesh network.

The average zero-load latency of RR-Net can be estimated as follows,

$$\delta N + (1-\delta)\frac{8(N+1)}{3} \tag{4}$$

where δ is the probability that two nodes are connected by a combined ring. δ is also approximately equal to the percentage of nodes that a combined ring can connect, i.e., $\delta \approx 4(N-1)/N^2$. In other words, in a larger topology each combined ring connects a lower percentage of nodes, and more packets are transmitted through the router-based network. As a result, the zero-load latency of RR-Net is closer to that of Baseline in a larger network.

It should be noted that this analysis is more applicable to uniform traffic pattern. The results in Figure 9 and Figure 10 show that for other traffic patterns such as transpose and bit_reverse, the zero-load latency of RR-Net can get close to the lowest zero-load latency even in the 16×16 mesh NoC.

7.4 Packet Latency for Realistic Benchmarks

The proposed RR-Net is also evaluated using realistic benchmark traces, which are obtained from Synful [5]. To support 8×8 network, we adopt a similar approach as [19], which assumes 4 identical instances of 4×4 nodes are arranged in the network and the directories are located at the left and right edge nodes of the 8×8 mesh. Figure 11 shows the comparisons of the packet latency, which is normalized to the Baseline. RR-net_1000 and RR-net_10000 are corresponding to the configuration intervals that are set 1000 and 10000, respectively. The figure shows that compared to Runahead, S-channel and SMART, RR-Net_1000 can achieve over 57.6%, 47.3%, 47.5% and 36.1% latency reduction, respectively. Especially for *barnes*, *bodytrack*, *lu_cb*, *water_spatial*, etc., RR-Net shows greater advantages due to relatively high traffic loads of these benchmarks. Moreover, a finer-grained time interval for RR-net also results in a 7.7% improvement on the packet latency.

The main reason for the low latency of RR-Net is that it has a higher saturation point than others. Different from synthetic traffic patterns that are with stable injection rates, the traffic behaviors of realistic benchmarks show high temporal variations. The average packet latency is related to the latencies at different traffic loads. When the traffic load is high, the networks are more easily saturated for Baseline, Runahead, S-channel and SMART. The saturated networks lead to much higher packet latency than that of RR-Net. In contrast, the traffic loads have fewer chances to saturate the network of RR-Net. Therefore, although RR-Net does not show obvious advantages on the zero-load latency over SMART and S-channel, it can still incur a lower packet latency on average.

7.5 Hardware Overhead

The allocator and the ring interface are implemented with Verilog HDL and synthesized using Synopsys Design Compiler with 45nm TSMC library. The implementations target 8×8 processing nodes. In other words, there are 4 horizontal rings and 4 vertical rings in the ring network. The number of interfaces is 64. For comparison, we synthesize a baseline 5-port router [26] with 8 virtual channels per port. The power results of the ring interface and the allocator are obtained from PrimeTime given 0.1 flits/cycle/node packet injection rate. The power of the baseline router is evaluated using DSENT [29]. The results are presented in Table 3. From the table, it can be observed that both a ring interface and an allocator have much lower area and power overhead than a baseline router. Compared to a baseline router, the proposed architecture incurs additional 6.5% area and 9.8% power consumption. In contrast, SMART, Runahead and S-channel incur approximately 1%, 3% and 20% additional area overhead respectively as reported in the literatures.

7.6 Power Consumption for Realistic Benchmarks

Since the ring interface has much lower power consumption than a router, the packet transmission through the rings can

2. This can be derived statistically or mathematically.

TABLE 3
Comparison of area and power

Type	Baseline Router	Allocator	Ring interface
Area (per node)	166204.4 μm^2	204.7 μm^2	10812.1 μm^2
Power (per node)	21.5mW	0.151mW	2.12 mW

lead to a lower power consumption for the network. Figure 12 presents the comparison of the power consumption between RR-Net and a baseline NoC using realistic benchmark traces. It can be seen that RR-Net results in a 15.7% improvement on average in terms of power consumption. The reason is that a ring interface has much lower power consumption than a router. With the ring interconnect, the router-based interconnect has fewer packets to transmit, leading to power saving of NoC. As a conclusion, the proposed RR-Net achieves a significant improvement on the power efficiency.

8 Conclusions

In this paper, we present a hybrid NoC that augments a router-based mesh network with a set of reconfigurable rings. The rings can be reconfigured to accommodate to traffic variations to achieve high communication performance. Due to simplicity of the reconfiguration point, we propose a reconfiguration algorithm with a linear-time complexity and implement it using a distributed hardware-based allocator. The experimental results show that the proposed NoC architecture can greatly improve both the saturation throughput and the zero-load latency.

Future work on reconfigurable NoCs can target how to design extremely low-latency reconfigurable NoCs for future large many-core systems, how to better adapt to the realistic traffic behaviors, etc.

(1) To design extremely low-latency reconfigurable NoCs, one possible future work is to combine the proposed reconfigurable architecture with the single-cycle multi-hop traversal [2], which can greatly reduce the zero-load latency. However, the complexity of the reconfiguration algorithm would be much higher because of more decision variables. Another possible future work is to adopt other interconnects (e.g. more complex ring shapes, wireless fabrics, etc.) for the design of reconfigurable NoCs. One of the main difficulties is to consider both the performance of the reconfigurable architecture and the complexity of the reconfiguration algorithm.

(2) The future work on reconfigurable NoCs should better adapt to realistic traffic variations for higher performance or lower power consumption. As we analyzed in this paper, the realistic traffic variations can be possibly predicted using machine learning algorithms on a long-term scale but exhibit less predictable pattern on a short-term scale. Reconfigurable interconnects targeting different scales can be designed separately to better exploit the characteristics of the traffic behaviors. New machine learning algorithms can also be developed to predict realistic traffic behaviors in a finer time granularity.

Acknowledgments

This work is supported in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grant No. 2018ZX01028201, in part by the National Natural Foundation of China under Grant No. 61672317 and No. 61834002, in part by Natural Science Foundation of Guangdong Province No. 2018A030313166, in part by Research Grant of Guangdong Province No. 2017A050501003, and in part by Pearl River S&T Nova Program of Guangzhou No. 201806010038.

References

[1] N. E. Jerger, T. Krishna, and L.-S. Peh, "On-chip networks, second edition," *Synthesis Lectures on Computer Architecture*, vol. 12, no. 3, pp. 1–210, 2017.

[2] T. Krishna, C. O. Chen, W. C. Kwon, and L. Peh, "Breaking the on-chip latency barrier using smart," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2013, pp. 378–389.

[3] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[4] T. Krishna, C. O. Chen, S. Park, W. Kwon, S. Subramanian, A. P. Chandrakasan, and L. Peh, "Single-cycle multihop asynchronous repeated traversal: A smart future for reconfigurable on-chip networks," *Computer*, vol. 46, no. 10, pp. 48–55, October 2013.

[5] M. Badr and N. E. Jerger, "Synfull: Synthetic traffic models capturing cache coherent behaviour," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, June 2014, pp. 109–120.

[6] Y. Jin, E. J. Kim, and T. M. Pinkston, "Communication-aware globally-coordinated on-chip networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 2, pp. 242–254, Feb 2012.

[7] N. Barrow-Williams, C. Fensch, and S. Moore, "A communication characterisation of splash-2 and parsec," in *IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 86–97.

[8] T. Sherwood, E. Perelman, G. Hamerly, S. Sair, and B. Calder, "Discovering and exploiting program phases," *IEEE Micro*, vol. 23, no. 6, pp. 84–93, Nov 2003.

[9] S. Abadal, A. Mestres, J. Torrellas, E. Alarcon, and A. Cabellos-Aparicio, "Medium access control in wireless network-on-chip: A context analysis," *IEEE Communications Magazine*, vol. 56, no. 6, pp. 172–178, June 2018.

[10] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," in *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA)*, 2007, pp. 150–161.

[11] N. E. Jerger, M. Lipasti, and L. Peh, "Circuit-switched coherence," *IEEE Computer Architecture Letters*, vol. 6, no. 1, pp. 5–8, Jan 2007.

[12] M. Modarressi, A. Tavakkol, and H. Sarbazi-Azad, "Application-aware topology reconfiguration for on-chip networks," *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, vol. 19, no. 11, pp. 2010–2022, Nov 2011.

[13] Y. Xue and P. Bogdan, "Improving noc performance under spatio-temporal variability by runtime reconfiguration: a general mathematical framework," in *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, Aug 2016, pp. 1–8.

[14] X. Wang, T. Mak, M. Yang, Y. Jiang, M. Daneshtalab, and M. Palesi, "On self-tuning networks-on-chip for dynamic network-flow dominance adaptation," in *2013 Seventh IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*, April 2013, pp. 1–8.

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

[16] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit, "An energy-efficient reconfigurable circuit-switched network-on-chip," in *19th IEEE International Parallel and Distributed Processing Symposium*, April 2005, pp. 4–8.

[17] J. Cong, M. Gill, Y. Hao, G. Reinman, and B. Yuan, "On-chip interconnection network for accelerator-rich architectures," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 8:1–8:6.

[18] R. Hesse, J. Nicholls, and N. E. Jerger, "Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels," in *Proceedings of IEEE/ACM Sixth International Symposium on Networks-on-Chip*, May 2012, pp. 132–141.

[19] Z. Li, J. S. Miguel, and N. E. Jerger, "The runahead network-on-chip," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, March 2016, pp. 333–344.

[20] U. Y. Ogras and R. Marculescu, ""it's a small world after all": Noc performance optimization via long-range link insertion," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 693–706, July 2006.

[21] S. Das, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "Design-space exploration and optimization of an energy-efficient and reliable 3-d small-world network-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 5, pp. 719–732, May 2017.

[22] R. Ausavarungnirun, C. Fallin, X. Yu, K. K. Chang, G. Nazario, R. Das, G. H. Loh, and O. Mutlu, "Design and evaluation of hierarchical rings with deflection routing," in *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*, Oct 2014, pp. 230–237.

[23] S. Liu, T. Chen, L. Li, X. Feng, Z. Xu, H. Chen, F. Chong, and Y. Chen, "Imr: High-performance low-cost multi-ring nocs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1700–1712, June 2016.

[24] F. Alazemi, A. AziziMazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *Proceedings of IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2018, pp. 492–503.

[25] H. Kim, G. Kim, S. Maeng, H. Yeo, and J. Kim, "Transportation-network-inspired network-on-chip," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014, pp. 332–343.

[26] D. U. Becker, "Efficient microarchitecture for network-on-chip routers," in *PhD thesis, Stanford University*, 2012.

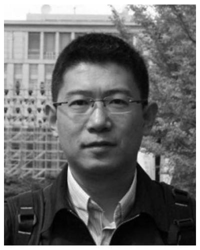
[27] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, New York, NY, USA, 2009, pp. 196–207.

[28] N. Agarwal, T. Krishna, L. shiuan Peh, and N. K. Jha, "Garnet: a detailed onchip network model inside a full-system simulator," in *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009, pp. 33–42.

[29] C. Sun, C. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. Peh, and V. Stojanovic, "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Proceedings of IEEE/ACM Sixth International Symposium on Networks-on-Chip*, May 2012, pp. 201–210.

Liang Wang received the BEng and MSc degree in electronics engineering from Harbin Institute of Technology, China, in 2011 and 2013 respectively, and the Ph.D degree in Computer Science and Engineering from The Chinese University of Hong Kong, Hong Kong, in 2017. He is currently a postdoctoral research fellow at Institute of Microelectronics, Tsinghua University, China. His research interests include power-efficient and reliability-aware design for network-on-chip and many-core system.

Leibo Liu received the BS degree in electronic engineering from the Tsinghua University, Beijing, China, in 1999 and the PhD degree in the Institute of Microelectronics, Tsinghua University, in 2004. He now serves as a Tenured Professor in the Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing and VLSI DSP.



Jie Han received the BSc degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the PhD degree from Delft University of Technology, Delft, The Netherlands, in 2004. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, the University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computation, reliability and fault tolerance, nanoelectronic circuits and systems, novel computational models for nanoscale and biological applications.

Xiaohang Wang received the BEng and PhD degrees in communication and electronic engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. He is currently an Associate Professor with the South China University of Technology, Guangzhou, China. His current research interests include manycore architecture, power efficient architectures, optimal control, and network-on-chip-based systems. Dr. Wang was a recipient of the PDP 2015 and VLSI-SoC 2014 Best Paper Awards.

Shouyi Yin received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tsinghua University, China, in 2000, 2002, and 2005, respectively. He was with the Imperial College London as a Research Associate. He is currently with the Institute of Microelectronics, Tsinghua University, as an Associate Professor. He has authored or co-authored over 40 refereed papers, and served as TPC member or reviewers for the international key conferences and leading journals. His research interests include SoC design, reconfigurable computing, and mobile computing.

Shaojun Wei was born in Beijing, China, in 1958. He received the Ph.D. degree from the Fault Polytechnique de Mons, Belgium, in 1991. He is currently a Professor with the Institute of Microelectronics, Tsinghua University. He is also a Senior Member of the Chinese Institute of Electronics. His main research interests include VLSI SoC design, EDA methodology, and ASIC design.