IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 1, 2019, accepted September 23, 2019, date of publication September 26, 2019,
date of current version October 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943922

An Approximate Bufferless Network-on-Chip

LING WANG[1,2], (Student Member, IEEE), XIAOHANG WANG[2], (Member, IEEE), AND YADONG WANG[1]

[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[2]School of Software Engineering, South China University of Technology, Guangzhou 510006, China

Corresponding author: Yadong Wang (ydwang@hit.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFC1202302 and Grant 2017YFSF090117, in part by the National Natural Science Foundation of China under Grant 61822108, Grant 61571152, and Grant 61971200, in part by the Natural Science Foundation of Guangdong Province under Grant 2018A030313166, in part by the Research Grant of Guangdong Province under Grant 2017A050501003, in part by the Pearl River S&T Nova Program of Guangzhou under Grant 201806010038, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019MS087.

ABSTRACT Bufferless network-on-chip (NoC) designs have drawn research attention in massively parallel multicore systems via their significant benefits in power and area savings. However, it shows poor throughput and low bandwidth in current bufferless designs due to complex bufferless routing and arbitration. Especially in the NACK-based bufferless network, the network performance will be affected significantly as the network conflicts increased caused by packet retransmissions. To this end, we proposed a novel approximate bufferless network, ABNoC. ABNoC lessens network conflicts and packet retransmissions via an approximate allocation mechanism (AAM) and a packet approximation method. We show that, compared with SCARAB, ABNoC improved the bandwidth up to 1.92 times and achieved 1.2 times faster in runtime. Besides, ABNoC accomplished 83.6% retransmission reduction and 46.7% latency reduction, while maintaining low application error.

INDEX TERMS Network-on-chip, bufferless NoC, approximate allocation, packet approximation.

I. INTRODUCTION

Over the last few decades, chip multiprocessors (CMPs) have replaced uniprocessors and become mainstream for building high-performance computers, due to the limitation of the power wall and the birth of advanced integration technology [1]. The network-on-chip (NoC) serving as an effective interconnection fabric connects these many on-chip components. It provides better scalability and higher bandwidth compared to traditional interconnections such as the bus and crossbar [2]–[4]. However, NoCs consume a significant amount of power in CMPs, that is, 40 percent of the tile power consumption in the 16-tile MIT RAW chip [5], 28 percent in the 80-tile Intel TeraFLOPS chip [6] and 19 percent in the 36-tile SCORPIO chip [7]. Buffers consume a large portion of network power and area [6], [8]. This motivates the bufferless design in low-power NoC architecture.

In the past, several bufferless NoCs have been proposed and achieve great power and area savings at a cost of lower throughput compared to conventional buffered networks [8]–[11]. For example, the CHIPPER [9], a misrouting-based bufferless NoC, reduces the network energy by 55% and achieves 36% area savings by eliminating buffers, but it increases the average runtime of benchmarks by 13.6%. We compared the average latency of bufferless and buffered NoCs as shown in Figure 1. The SCARAB [10] and CHIPPER [9] are two state-of-the-art bufferless designs. The former is an optimized NACK-based network that sends NACK messages to trigger retransmissions of dropped packets in the event of a collision, and the latter enables NACK-less operations by misrouting the conflicting flits. Compared with the buffered NoC, the SCARAB network shows a lower bandwidth, and the CHIPPER NoC runs with a higher average latency and this becomes worse as the injection rate increases. Therefore, bufferless NoCs are only suitable for low injection rates. However, in the big data era, it is essential to design high-performance bufferless NoC that achieves low latency and high bandwidth while maintaining the energy and area benefit of bufferless design.

Many applications, such as machine learning, searching, scientific computing and multimedia applications, have shown inherent fault tolerance. They allow inexactness in outputs and can employ selective approximation to achieve

The associate editor coordinating the review of this manuscript and approving it for publication was Vivek Kumar Sehgal.

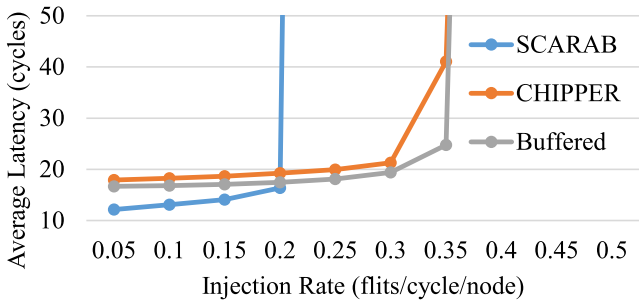


FIGURE 1. Average latency for single-flit uniform random traffic on bufferless and buffered NoCs.

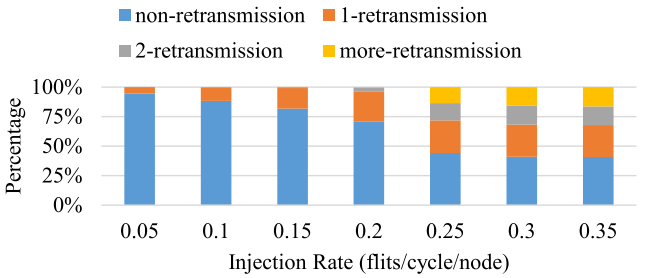


FIGURE 2. The percentage of packets with different retransmissions under uniform random workloads.

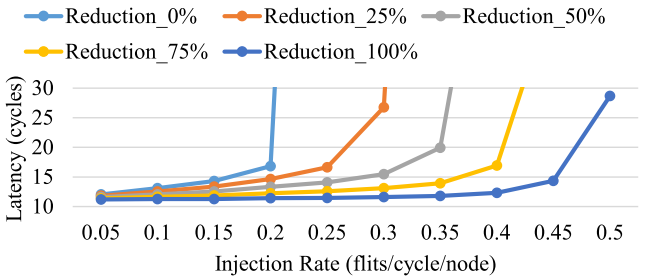


FIGURE 3. Average latency of bufferless networks with different retransmission reduction ratios under single-flit random workloads.

better performance [12] These safe-to-approximate applications motivate the approximate hardware designs which trade off accuracy for performance improvement and energy saving.

Recently, some new approximate NoC techniques focusing on providing disproportionate gains in efficiency have been presented for the scientific community. APPROX-NoC [13] exploits lossy data compression in NoC. It facilitates approximate matching of data patterns to improve the compression ratio. Our previous work ABDTR [14] also shows an approximation-based dynamic traffic regulation mechanism in NoC. The ABDTR can regulate the injection rate of each router by selectively dropping some approximable data based on the network congestion information. These approximate NoCs reduce the volume of data injected and achieve good latency reduction in buffered NoCs. But to date, no literature presents approximation in bufferless NoC, relaxing communication accuracy to increase throughput of bufferless transmission.

Instead of in-router buffering, NACK-based bufferless NoCs discard contending packets in the event of a collision and then retransmit them from the sources. We evaluate the the ratio of retransmitted packets under uniform random workloads. All the injections are single-flit packets. Figure 2 shows that the percentage of retransmitted packets increases as the injection rate increases. When the injection rate is greater than 0.2, more than 50% packets are retransmitted, and about half of them are served more than once. Thus, we intend to reduce packet retransmissions for throughput improvement (part of retransmitted packets are dropped instead of being reinjected). Figure 3 shows the average latency in bufferless network with different retransmission reduction ratios. Obviously, reducing retransmissions results in a great latency reduction and a bandwidth improvement. And the greater the retransmission reduction, the greater the return is. Therefore, the main goal of this paper is to reduce network retransmissions with low quality loss and achieve a performance improvement in bufferless NoC.

In this work, we design the ABNoC, an approximate bufferless NoC. It relaxes the transmission accuracy to reduce packet retransmissions and increase network throughput. The ABNoC is a lossy NACK-based bufferless design. It can discard conflicting approximable flits without retransmission and recover them after packet transmission, thereby reducing packet retransmissions and improving bufferless network performance.

We make the following contributions in this paper:

(1) A novel approximate bufferless NoC to materialize a low latency and high bandwidth bufferless transmission is presented.

(2) We design an approximate allocation mechanism and a packet approximation method to reduce network conflicts and packet retransmissions and recover the approximable missing flits.

(3) Experiments show that ABNoC achieves an average 83.6% retransmission reduction, an average 46.7% latency reduction and a up to 1.92× bandwidth improvement compared to SCARAB.

The rest of the paper is organized as follows. In Section 2, we motivate our work by presenting the big impact of packet retransmission on bufferless NoC performance and the error tolerance of wide range applications. Section 3 gives an overview of the approximation framework. Section 4 explains the microarchitectural implementation and functional principles of ABNoC. Section 5 shows the packet approximation method. In Section 6, we present our experimental setup. Section 7 details the evaluations. Section 8 presents the related work. Finally, we conclude our work in Section 9.

II. BACKGROUND AND MOTIVATION

The SCARAB [10] is an outstanding NACK-based bufferless NoC that implements a single-cycle latency pipeline and provides high-speed in-router transmission. Besides, it equips a physically separate circuit-switched NACK network to eliminate the negative impact that NACKs will further increase

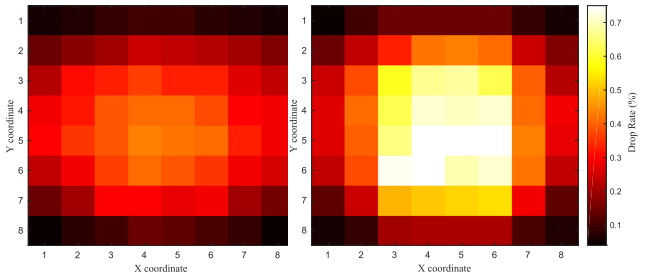


FIGURE 4. Network conflict distribution at a random injection rate of 0.2 flits per cycle per node.

the total network congestion. The SCARAB also introduced an opportunistic buffering technology. It can retransmit the dropping packets from an intermediate node, not the sources, by utilizing the processor-side buffers (miss status handling registers) to opportunistically store some in-flight packets. Thereby, SCARAB shows a lower energy consumption and a lower latency of retransmissions compared with previous NACK-based bufferless NoCs. In this paper, we use the SCARAB as a baseline. All the bufferless technologies are preserved except for opportunistic buffering. Since the opportunistic buffering could likewise be applied to the other bufferless designs to reduce retransmission latency. Furthermore, opportunistic buffering is not a completely bufferless design but needs to take up processor-side buffers. ABNoC is also an NACK-based bufferless NoC, but it introduces approximate designs, relaxing transmission accuracy for retransmission reduction and improving performance. The main motivation of the ABNoC design is that (1) retransmission seriously exacerbates network conflicts in NACK-based bufferless NoC, and (2) lossy communication is acceptable in applications that exhibit some level of error tolerance.

A. RETRANSMISSION SERIOUSLY EXACERBATES NETWORK CONFLICTS

In NACK-based bufferless NoCs, all the conflicting packets are dropped and retransmitted. To analyze the impact of packet retransmission on network performance, we design a BASELINE NoC without retransmissions to compared with the SCARAB. The BASELINE is almost the same as the SCARAB, except it cannot generate NACK messages for retransmission. Thus, in a conflict the BASELINE drops all the conflicting packets but does not try to retransmit them again. It is a lossy bufferless network. Both the SCARAB and the BASELINE are 8×8 2D mesh network, and all injected packets are single-flit and on a random traffic pattern.

Figure 4 shows the conflict distribution in BASELINE and SCARAB at an injection rate of 0.2 flit per cycle per router. Both SCARAB and BASELINE drop at least one packet in the event of a conflict. Therefore, the conflict distribution is described based on the drop rate of each router, which is calculated as in Equation (1),

$$r_i = \frac{d_i}{t + rt}, \tag{1}$$

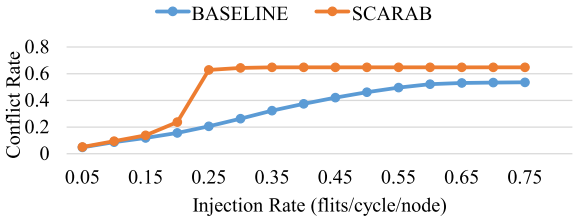


FIGURE 5. Conflict rate of bufferless NoC under single-flit random workloads.

where r_i is the drop rate of router i, d_i is the number of dropped packets in router i, t is the number of total injected packets, and rt is the number of total retransmissions. The drop rate is the ratio of the number of dropped packets to the number of total transmitted packets. The heat map shows the drop rate of each router, whereas the brighter the color, the larger the value is, and vice versa. It is clear that the drop rate in the SCARAB is much larger than that in BASELINE, especially for the nodes in the middle of network. For further comparison, we evaluate the conflict rate in SCARAB and BASELINE at different injection rates, as shown in Figure 5. The conflict rate is the sum of the drop rates of routers. It is calculated as in Equation (2),

$$c = \sum_{i=0}^{n} r_i, \tag{2}$$

where c is the conflict rate, and n is the number of routers in NoC. Figure 5 shows that the conflict rates of SCARAB are always greater than that of BASELINE. As the injection rate increases, the conflict rate of SCARAB grows faster than that of BASELINE. Furthermore, Figure 1 and Figure 5 show that both the average latency and the conflict rate increase sharply when the injection rate exceeds 0.2 flit/cycle/node. In NACK-based bufferless NoCs, network conflict results in packet retransmission, and retransmission exacerbates network conflict and packet latency, hereby the network performance degrades greatly and the throughput quickly saturates at a high injection rate. In this paper, we design an approximate allocation mechanism (AAM) that aims to mitigate network conflicts and reduce retransmissions.

B. LOSSY COMMUNICATION IS ACCEPTABLE

Many applications in domains such as image/video processing and machine learning allow inaccurate outputs. They do not require exact data transmission for accurate computations. Previous works have proposed load value approximation for CPU [15] and GPU [16], which shows great potential in relaxing the accuracy of computing data. Moreover, some annotation frameworks that label sections of the safe-to-approximate data are also proposed [15], [17], [18]. These applications and designs accentuate acceptable inaccuracy of communication and computation. Many approaches, such as APPROX-NoC [13], ABDTR [14] and DAPPER [19], have explored lossy communication in NoCs and achieve a good latency reduction and power saving. These approximation

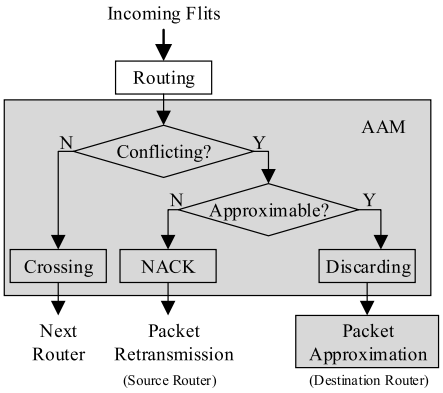


FIGURE 6. ABNoC operation flowchart.

designs show that applications have a sufficient amount of value similarities, and data traversing the network have sufficient capacity to be approximated. Approximation, relaxing transmission accuracy, is a promising technology to achieve high performance and low power consumption in NoC connecting a large number of on-chip components. We propose a novel approximate bufferless architecture. It includes an in-network flit discarding scheme and a packet approximation method to improve bufferless network performance while maintaining low quality loss.

III. APPROXIMATION FRAMEWORK

A. APPROXIMATION DESIGN

In this paper, flits are delivered independently in ABNoC. In each router, any incoming flit performs routing and port allocation. Figure 6 shows the principle of ABNoC. In case of a routing conflict, only one flit can be allocated successfully and cross the router while all the conflicting flits being dropped. The AAM triggers a NACK message for packet retransmission when a non-approximable flit is dropped. For the approximable ones, they will be discarded directly and be recovered at the destination node through packet approximation method after the packet transmission. This design will reduces the retransmission of packets with approximable flits. In ABNoC, not all non-approximable flit drops will trigger NACKs immediately in the network conflicts. Some NACKs are triggers in the destination routers when non-approximate flits in a packet are found missing. The approximate transmission will be expanded in the next subsection.

Another important design is the packet approximation method that works for recovering the approximable missing flits. The packet approximation in ABNoC includes two parts: approximate encoding before packet injection and approximate decoding after packet transmission. We encodes all the the approximable flits of a multi-flit packet into a non-approximable flit and transmit it with the multi-flit packet. The non-approximable flits can be preserved in ABNoC. Thus, the missing approximable flits can be decoded from the encoded flit. This approximate method only works for multi-flit packet transmission. All the single-flit packets

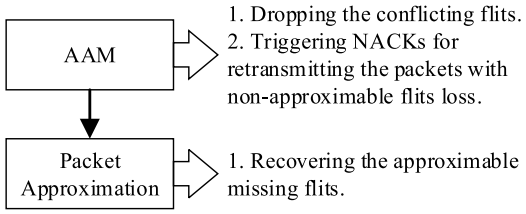


FIGURE 7. Approximation framework overview.

in our design are non-approximable and can achieve lossless transmissions in ABNoC.

Figure 7 shows the overview of the approximate design. In each router the AAM drops the conflicting flits and triggers NACKs for retransmitting the packets with non-approximable flits loss. Thereby, the conflicting approximable flits are discarded without retransmission. And ABNoC results in the packet retransmission reduction and network conflicts mitigation. After a packet transmission, the missing approximable flits in the packet will be recovered through the packet approximation method. The ABNoC provides an in-network traffic reduction can accurately capture the network congestions. In ABNoC, only the conflicting approximable flits are discarded and approximated, while others still have lossless delivery. Therefore, ABNoC could improve bufferless network performance while maintaining low application error.

B. APPROXIMATE NACK-BASED BUFFERLESS TRANSMISSION

In ABNoC, the flit transmissions and ACK/NACK feedbacks are separated. Each packet is transmitted with an exclusive NACK channel for ACK/NACK message transfer. For single-flit packet, the single flit configures the NACK channel while it is transmitted over a router. For multi-flit packet, only the head flit is transmitted with a configured NACK channel, while others enable NACK-free transmission and travel independently in ABNoC. There are no buffers at the switch ports in ABNoC, so flits cannot be stopped. In every cycle, flits that arrive at the input ports contend for the output ports. When two or more flits contend for the same output port, only one can be transferred through the output channel, and others are dropped. If the single-flit packet or the head flit of a multi-flit packet fails in any allocation or there are no remaining NACK channels, it will be dropped. Then, an NACK message travels along the preconfigured NACK channel back to the source to trigger the whole packet retransmission. Other flits are free to transmit and perform independent routing and port allocation in each router. If they are dropped in a conflict, the NACK message will not be generated immediately. After a multi-flit packet finishes its transmission,[1] it will be checked whether

[1]When the head flit and the last flit of a multi-flit packet reach the destination or the limited waiting time of the head flit at destination node is over, the packet transmission is considered complete.

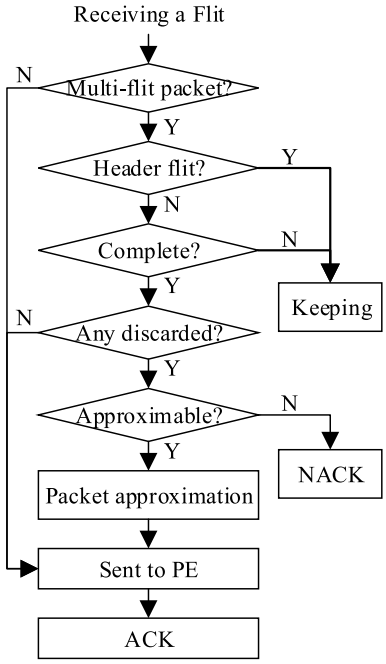


FIGURE 8. Destination node operation flowchart.

all the non-approximable flits have been received.[2] If there is any non-approximable flit being lost, the NACK channel corresponding to this packet will signal an NACK message to its source node for packet retransmission, and the received flits of the packet will be discarded. For other missing flits, they will be recovered through the packet approximation method, while an ACK signal will be triggered for releasing the exclusive NACK channel.

Figure 8 shows the overall flow in the destination node. When a flit successfully reaches its destination, if it is a single-flit packet, it can be sent directly to the processing element (PE) and then an ACK message will be signaled back. Or if it is the head flit of a multi-flit packet, it will be kept in the destination node, waiting for the remaining flits of the packet. All the received flits of a multi-flit packet can be kept in destination node until the packet transmission is over. After a multi-flit packet transfer is completed, the packet will be checked whether there is any flit being discarded. The loss of non-approximable flits will trigger NACK messages and the approximable missing ones can be approximated. If no non-approximable flits are lost, an ACK message will be fed back along the exclusive NACK path.

IV. ARCHITECTURE DESIGN FOR ABNOC

A. MULTIPLANE ARCHITECTURE

The ABNoC router separates packet transmission and ACK/NACK rollback by a multiplane architecture to avoid traffic collisions caused by ACK/NACK messages. The ABNoC provides a two-plane architecture: one for data transfer named data network and the other for ACK/NACK rollback named NACK network. Each packet transmitting in the data network obtains an exclusive channel of the NACK network to the ACK/NACK message rollback. In the NACK-network, the exclusive channel is allocated based on the allocation results of the data network. Thus, the NACK network acts as a preconfigured circuit-switched network. Note that each packet is delivered with only one exclusive channel of the NACK network. For a multi-flit packet, only the head flit shares allocation results with paired routers in the NACK network and is transmitted with an exclusive NACK channel. Moreover, the head flit or a single-flit packet can be successfully allocated in each router only if the productive output is obtained and a free NACK channel exists along its output path.

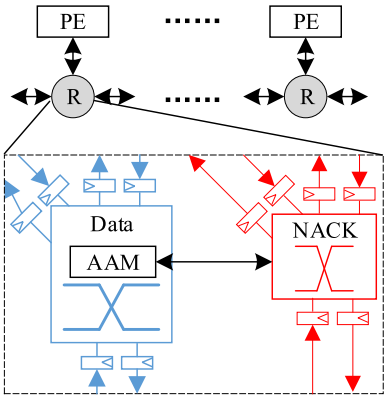


FIGURE 9. ABNoC architectural overview.

Figure 9 shows the high-level architectural depiction of the ABNoC router. The ABNoC connects the processing elements (PEs). In ABNoC, each router consists of two interrelated but physically separate components: the data router and NACK router. The data router is a fully functional bufferless router, while the NACK router does not need routing computation, and it is configured with the allocation results from the data router. Each data router also contains an AAM that can discard the conflicting approximable flits without NACK messages.

B. BUFFERLESS DATA NETWORK

1) BUFFERLESS ROUTER

Figure 10 illustrates the design of the bufferless router in the data network. Each router consists of five multiplexers: four cardinal neighbor connections and one local connection. There are no buffers needed for packet transmission in a router. Each cardinal neighbor connection is only equipped with a latch for storing a flit that may come in from the output port of neighbor direction at any cycle. And all arriving flits contend for output ports and are forced to be routed out in the next cycle, except the flits in injection port. Injection can only occur when the required output slot is free. Therefore,

[2]The approximable flits' quantity of a multi-flit packet is stored in its head flit, and the approximable status is also stored in each flit. Thus, if the number of received non-approximable flits does not equal the total flits number minus the approximable flits' quantity, there is at least one non-approximable flit being dropped.

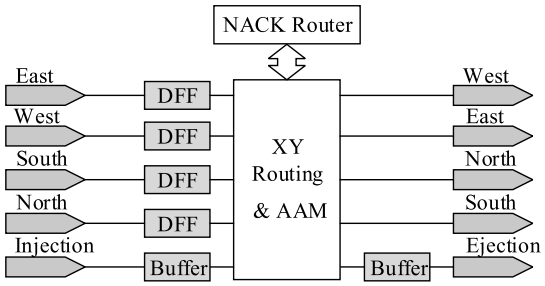


FIGURE 10. Bufferless data router architecture.

some injection buffers are needed to store the flits waiting to be injected. Some ejection buffers are also necessary for receiving the arrived packets. But the injection and ejection buffers can also be replaced with the miss status handling registers (MSHRs) as in previous bufferless designs [9], [10].

The bufferless data router supports XY routing and AAM. The XY routing and AAM are combined together, and can quickly determine whether the incoming flits are being served or discarded without retransmission. Due to the lossy transmission, the data router simplifies the complex conflict prevention design in previous bufferless NoCs. It is with a lightweight router architecture which consumes very little area and power and reduces the pipeline stages for single-cycle delivery in a router. In Figure 10, the assignments of input ports relative to output ports is 'twisted'. The reason is that straight-through router traversals ($N \Leftrightarrow S$ and $E \Leftrightarrow W$) are more common than turns.

In the data router, each flit in a packet starts with some header bits for independent routing and port arbitration. Figure 11 shows the flit packetization in an 8×8 mesh network. The header bits are encoded as follows, a 5-bit priority, 1-bit single/header indicator, 6-bit destination address, 6-bit source address and 3-bit flit ID. The 128-bit data bits are used for storing some int or float values. In ABNoC, the XY routing and AAM is a priority-based design. The non-approximable flits will have a higher priority than approximable ones. And the priority of non-approximable flits is incremented with retransmissions. The first 4 bits of the 5-bit priority are used to record the number of retransmissions of non-approximable flit. The last 1 bit indicates the approximable info of flit (1 for non-approximable, 0 for approximable). The priority of an approximable flit is always 0. The priority and destination info is used for the XY routing and port allocation. The 1-bit single/header indicator determines whether a flits needs to configure an NACK channel during the transmission. Only the single-flit packet and the head flit of multi-flit packet is transmitted with a NACK channel and shares the allocation results with a paired NACK router. The 3-bit flit ID has two functions. It indicates the place of a flit in an 8-flit data packet. And in the encoded head flit, it stores the number of approximable flits in a 8-flit data packet. Based on the source, ID and flit approximable info, at the destination node we can

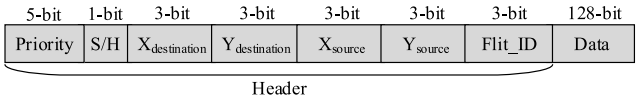


FIGURE 11. The flit packetization.

determine whether the transmission of a multi-flit packet is complete and whether there is a non-approximable flit loss.

2) ROUTING

The bufferless data network employs XY routing. The productive output port of each flit is based on its destination address, which is stored in its header bits as X and Y values. Routing computation in each router determines the correct output port by comparing the destination address with the local address. XY routing provides each packet with a unique transmission path, so the flits in a multi-flit packet are routed independently but with the same output. Therefore, the order of flits in a multi-flit packet will not be changed with the bufferless transmission.

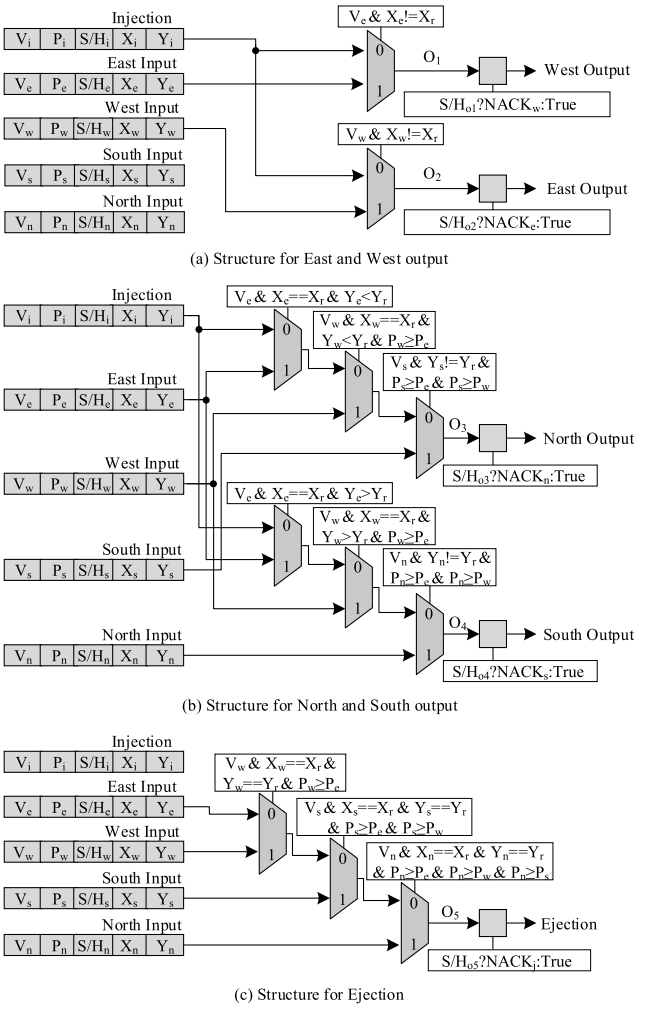
3) AAM

The AAM has three functions: (1) allocating the output port to flits based on the routing result, (2) dropping contending flits with low priority, and (3) triggering an NACK message when a single-flit packet or a head flit is dropped. With XY routing, we fix the arbitration scheme for each output port. All the input ports are hardcoded with a default priority: north > south > west > east > injection, which means if flits with the same priority from two different input ports, one always take precedence based on this default priority. This design allows all the incoming flits to quickly be either directed to their preferred output ports or dropped in a conflict. The fixed arbitration follows a rule that a flit going straight has a higher priority than a flit that is turning. The injection port always has the lowest priority because the flits can be buffered first in the injection port. If a flit in the injection port does not succeed in arbitration for its output port, we can try again in the next cycle until it is injected into the network for transmission.

4) IMPLEMENTATION

The required signals for routing and allocation are obtained from the header bits of the incoming flits at each input port. In Figure 12, the $V_{direction}$, $P_{direction}$, $S/H_{direction}$, $X_{direction}$, $Y_{direction}$ correspond to the valid bit, priority, single or head flit, destination X and Y values respectively. Once a flit arrives at a input port, the valid bit will be activated, which indicates a valid flit is waiting at the input port. X_r and Y_r are the local address of the working router. $NACK_{direction}$ represents whether there is at least one NACK wire not being assigned at the output port direction in the NACK network.

Figure 12(a) shows the east and west output routing and allocation. Based on the XY routing, the east and west direction output only needs to consider the latches of their



(a) Structure for East and West output

(b) Structure for North and South output

(c) Structure for Ejection

FIGURE 12. Routing and AAM.

opposing input ports and the injection port, regardless of the flit priority. Any time that a flit arrives at either the east or west input port with a nonzero X value, it will be guaranteed to be forwarded straight since it has the highest priority in our fixed arbitration scheme. Routing and arbitration for the north and south output ports are more complicated since they need to consider flits turning.

Figure 12(b) shows that the arbitration follows the default hardcoded priority. The multiplexers always check the opposing input port first and then determine if the flit in west input port is turning, followed by the east input port, and finally the injection port, if all the input flits have the same priority. Otherwise, the flit with higher priority will be preferred. Note that a flit traverses at most three 2-to-1 multiplexers from its input port to its output port, which keeps the critical path delay low.

The logic circuit in the ejection port, shown in Figure 12(c), is similar to that of the north and south output ports. Four channels are selected by three multiplexers, and the incoming flits are ranked based on their flit priorities and the default hardcoded rule. The source address of a packet can never be the same as its destination address, thus eliminating the need to connect the ejection port to the injection port. To determine if a packet is destined for the ejection port, both the X and Y values need to be zero.

In addition, each output port is also equipped with a selector. If the output flit of the outermost multiplexer is a head flit or a single-flit packet, the selector will work based on the $NACK_{direction}$ signal. Otherwise, the flit will go directly through the selector. The selector is used to ensure that each head flit or single-flit packet traversing the router configures a 1-bit NACK path. If the $NACK_{direction}$ signal is false, the head flit or single-flit packet will be dropped and an NACK signal will travel along the corresponding NACK path to trigger retransmission. Moreover, the design performs routing and AAM together to allow flits to traverse each router in a single cycle.

5) DEADLOCK AND LIVELOCK

The XY routing and AAM greatly simplify route computation and port arbitration. Our design can quickly direct flits from the input port to their productive output ports and determines which flits to be dropped with or without the NACK signal in the event of a conflict. All the conflicting flits are dropped instead of being blocked in ABNoC. The bufferless data network is deadlock-free in nature due to the XY routing and AAM design.

To prevent the livelock problem caused by saturating priority, we provide 4-bit priority increment, which supports up to 15 retransmissions before saturation. This can satisfy most packet retransmission requests. Even for the worst case in which a packet is retransmitted up to 15 times, we will stop injecting other packets with 15 retransmissions until this packet finishes transmission. This design will ensure that there is at most one 15-priority packet that is transmitted in the network and avoids livelock.

There is another situation that may lead to a livelock. Flits perform independently in each router, so a multi-flit packet injection may be interrupted by transmissions of other flits due to the lowest priority of the injection port. The interruption will result in a long injection interval between two flits of the packet. In addition, for the multi-flit packet, the injection period from the head flit to the last flit is the same as its ejection period due to no network congestion. After the ejection space is filled, if the multi-flit packet has not completed the transmission, a livelock will be generated. To avoid the livelock caused by the limited ejection space being filled, the packet injection period needs to be limited. In our design, each router allocates a counter of fixed E cycles in the injection port. When the head flit of a multi-flit packet is injected, the counter starts to decrease with the number of cycles until it reaches zero or the last flit of the packet is injected. When the count is over, the remaining flits of the multi-flit packet will be dropped. Then, the packet will be stored into MSHR and wait for retransmission. Each router injects one flit per cycle. Thus, the $E + 1$ must be larger than the size of the data packet, where the 1 is for the head flit.

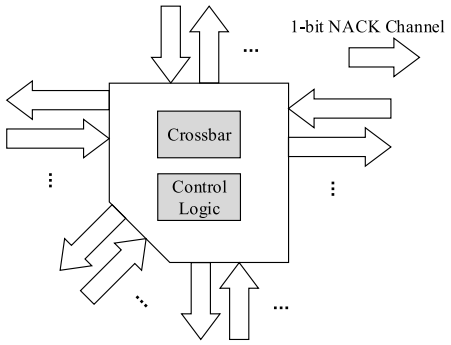


FIGURE 13. Overview of the multichannel NACK router.

The waiting time of head flit in destination node will not exceed E cycles, since it is the same as the packet injection period.

C. NACK NETWORK

1) MULTICHANNEL CIRCUIT-SWITCHED NETWORK

The NACK network in the ABNoC operates as a small 5×5 mux-based switch, and each port is equipped with multiple 1-bit channels, which is similar to the design in SCARAB [10]. Figure 13 illustrates the overview of an NACK router. There are no buffers and routing and arbitrator microarchitecture. The NACK router works with only some logic to control the crossbar for establishing or releasing a 1-bit pathway to transfer the acknowledgment (ACK) and negative acknowledgment (NACK) messages. The routing and allocation information comes from its paired data router. When all channels of a port are assigned, an $NACK_{direction}$ signal will be activated and fed back to the data router. Compared with the traditional NACK-based bufferless NoC that needs to send a full packet to request retransmission, the 1-bit channel NACK network results in a very energy efficient means of signaling feedback.

Moreover, messages in the NACK network take two cycles per hop. Any message sending out on an even clock cycle can only be transferred by the next router in a later even cycle. This enables us to use time division multiplexing (TDM) technology to halve the number of inter-router NACK channels and sustain network bandwidth. This design reduces the number of NACK channels for area and power saving in ABNoC. The TDM allows each inter-router NACK circuit channel to transfer two messages that are differentiated by even and odd cycles. The circuit path setup and message transmission are separate in the NACK network. Thus, any packet transmitted in the data network needs to be NACKed or ACKed after an even period, which drives the maximum injection period (E) to be an even value.

2) PATH ESTABLISHMENT AND RELEASE

The NACK network works as a general circuit-switch network, where a dedicated circuit channel must be set up before data transmission. Every packet traverses a router with configuring a 1-bit channel of the NACK network. This NACK channel is connected along the packet's entire path, such that the ACK or NACK message is transmitted along the preconfigured circuit path. In the NACK network, each feedback message transfers each preconfigured hop with 2 cycles, which creates a deterministic delay of message transmission of $2 \times N$ cycles, where N is the number of hops of the preconfigured circuit path.

Unlike the SCARAB design that allows NACK channels to be reallocated after the implicitly ACKed transmission delay, the ABNoC releases a circuit path with the NACK and ACK message. When a router in the NACK network receives a feedback message, the router releases the corresponding configured connection with the previous node. This means that each established path is only used once for transferring the NACK or ACK message before being released. Moreover, the NACK network in ABNoC is more energy efficient in comparison to that in the SCARAB due to not needing counters to record the implicit ACKed epochs.

In the ABNoC design, a feedback message can be triggered in three situations: (1) An NACK signal is triggered when the head flit or single-flit packet is dropped during its transmission; (2) signaling NACK at the destination node since a non-approximable flit is not received after the packet transmission; and (3) an ACK signal travels from destination node to the source when the packet is delivered successfully, which means all the non-approximable flits are received at the destination node. As the transmission delay in both the NACK network and data network are proportional to the delivery hops ($2 \times$ hops), the time of the transmission delay plus the NACK/ACK message feedback time will be 4 times the hops. At the destination node, the ejection period is the same as injection period which is less than E cycles. The maximum waiting time of head flit is E cycles. Thus, the window of time that a packet could be NACKed or ACKed after initial transmission is deterministically bounded, as shown in Equation (3),

$$L = 4 \times (|X_s - X_d| + |Y_s - Y_d| + 1) + E \tag{3}$$

where L is the maximum delay, E is the maximum injection period, (X_s, Y_s) and (X_d, Y_d) are the source address and the destination address respectively.

V. PACKET APPROXIMATION

Previous approximation designs in NoC focus on the reduction of data injection [13], [14], [19]. They discard some approximable data before being injected into the network. However, the injected flits in these designs still need to face network conflicts and wait, which results in long latency of contending flits. These end-to-end traffic reduction schemes cannot response to network conflicts on the spot. Therefore, we propose an in-network traffic reduction scheme which can accurately capture the network workload. In ABNoC, contending flits are discarded instead of waiting, and the approximable ones can be safely approximated after packet transmission. Packet approximation is a stage for recovering the lost approximable flits. In this paper, we propose a new

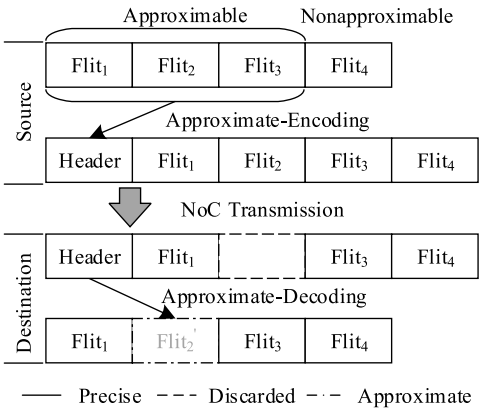


FIGURE 14. Approximate encoding and decoding of a 4-flit packet.

value approximation method for approximating the missing flits with low quality degradation.

In packet approximation, each multi-flit packet starts with a non-approximable head flit. All the approximable flits are encoded into the head flit. ABNoC ensures that the non-approximable flit will not be lost in packet transmission. So that, the discarded approximable flits can be decoded from the head flit at the destination node.

Figure 14 shows a packet approximation example with a 4-flit packet transfer. We assume that the first three flits are approximable and the last one is non-approximable. Note that the number and the place of approximable flits have no effect on our design. All the approximable flits can be encoded into a head flit that is signed non-approximable. The head flit and the existing 4 flits comprise a new 5-flit packet and are injected into the ABNoC. During the packet transmission, any approximable flit could be discarded in a network conflict. In the example, we assume the approximable $flit_2$ is discarded. After the packet reaches the destination router, the $flit_2$ is recovered as the $flit_2'$ by decoding the head flit.

In this work, we focus on 32-bit integer and floating-point value approximation. The 128-bit data of a flit consists of 4 integer or floating-point values. Each flit recovery is to approximate the four values. Note that a flit is annotated as approximable only when the flit stores the words of a same type and they are all approximable. The approximate encoding and decoding will be elaborated in the following subsection.

A. APPROXIMATE ENCODING

The approximate encoder works by using the 128-bit data space of head flit to encode the words in all approximate flits. The number of approximable flits in a packet is 0 to 8. If the number is 0, we will set the last flit as approximable. This will not cause a loss of quality to the packet, since the head flit can duplicate the words in the approximable flit in a one-approximable-flit packet Thus, there is at least one approximable flit in any multi-flit packet. In approximate encoding, the 128-bit data space of head flit is divided according to the number of approximable flits. Then each divided part is used to encode a 4-value approximable flit. The number of approximable flits (1-8) can be stored in the 3-bit ID of head flit.

Algorithm 1 Encoding

Input: $n \leftarrow$ the number of approximable flits
$flit_0, ..., flit_{n-1} \leftarrow$ the data bits in approximable flits
Output: $H \leftarrow$ the data bits in head flit

```
if n == 1 then
    // the approximable data is
    // duplicated into the head
    H = flit_0;
end
if n == 2 then
    // the data space is divided into 2 equal parts
    H_i ← H/2, i ∈ {0, 1}
    // the half stores the encoded flit_0
    H_0 ← approximate(flit_0);
    // the other half stores the encoded flit_1
    H_1 ← approximate(flit_1);
end
if n == 3, 4 then
    // the data space is divided into 4 equal parts
    H_i ← H/4, i ∈ {0, 1, 2, 3}
    for j =0; j<n; j++ do
        // each flit is encoded into an equal part
        H_j ← approximate(flit_j)
    end
end
if n == 5, 6, 7, 8 then
    // the data space is divided into 8 equal parts
    H_i ← H/8, i ∈ {0, 1, ..., 7}
    for j =0; j<n; j++ do
        // each flit is encoded into an equal part
        H_j ← approximate(flit_j)
    end
end
return H
```

Algorithm 1 shows the encoding process. In approximate encoding, the 128-bit data space of head flit can be divided into 1, 2, 4 or 8 equal parts, shown as the Figure 15. Specifically, 1 is for one approximable flit, 2 for two, while 4 is for three and four approximable flits, and 8 is for five, six, seven or eight. Then all the approximate flits are encoded into the divided parts in order.

Encoding a flit is to approximately store the four 32-bit integers or floating-point values of the flit. First, we encode each 32-bit value as a 16-bit data. Figure 16 shows the integer and floating-point value encoding. The most significant bit of the 16-bit data is used to indicate the value type, 0 for integer and 1 for floating-point. For the integer value, we shift data to the right by n digits for converting an integer value to a maximum 10-bit value (A-value). The shifted number n is

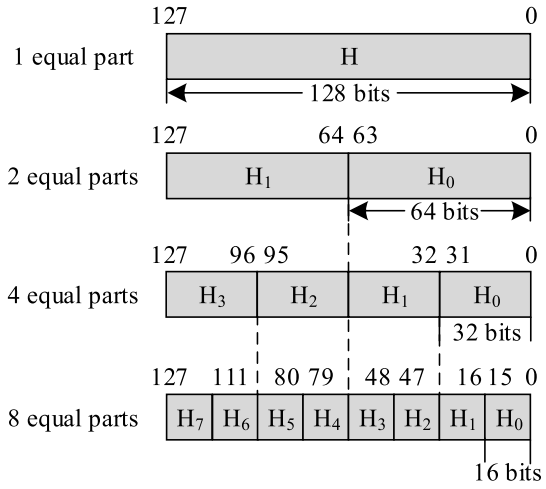


FIGURE 15. 128-bit data space division in head flit.

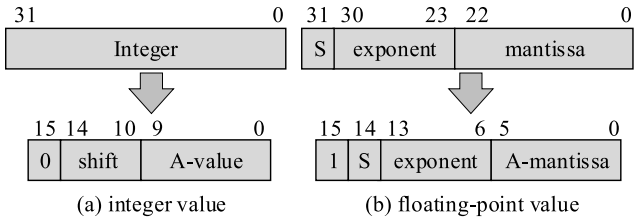


FIGURE 16. Integer and floating-point value approximate encoding.

stored in the 16-bit data with the 10-bit value. For example, an integer value +445566789 will be shifted to the right by 20 digits and stored as +0x1A8 (A-value). The coding result is 0x51A8. The floating-point value is represented as Equation (4),

$$float = (-1)^S \times (1 + .mantissa) \times 2^{exponent} \quad (4)$$

where S is the sign bit. The mantissa and exponent are stored in a 23-bit and an 8-bit spaces separately, as shown in Figure 16(b) We propose to approximate only the mantissa field of the floating-point value. We encode the 23-bit mantissa in a 6-bit A-mantissa by removing the 17-bit LSB and the the sign bit and exponent bits are reserved.

Second, the encoded values are stored into a divided part (H_i). Different divisions result in different sizes to store these values. When the divided part is full, the rest encoded values that can not be stored will be discarded. For example, if there are three approximable flits, each flit will be encoded into a 32-bit space. Only the first 2 values of the approximable flit can be encoded and stored in a divided space. The remaining 2 values of the flit will be discarded.

B. APPROXIMATE DECODING

Decoding is to recover the missing flits. When a multi-flit packet finishes transmission, the decoder will first determine the divisions of the data bits in the head flit by the number of approximable flits. Then, each missing flit will be determined which division it encodes. Finally, we can recover the data in the missing flit.

Algorithm 2 Decoding

Input: $n \leftarrow$ the number of approximable flits
$H \leftarrow$ the data bits in head flit
Output: $flit_0, ..., flit_m \leftarrow$ the data bits in the missing flits

```
// determine the division in head flit
if n == 1 then
    return flit_0 = H
end
if n == 2 then
    H_i ← H/2, i ∈ {0, 1}
end
if n == 3, 4 then
    H_i ← H/4, i ∈ {0, 1, 2, 3}
end
if n == 5, 6, 7, 8 then
    H_i ← H/8, i ∈ {0, 1, ..., 7}
end
//the flit_i is the jth approximable flit
for i=0; i<m; i++ do
    //4-value recovery per flit
    while 4 times do
        if H_j is not empty then
            // recovering a value from 16 bits
            // in jth part of head flit
            32-bit value ← recover(0xFFFF & H_j)
            // storing the value into flit_i
            flit_i ← flit_i << 32 + 32-bit value
            // remove the decoded bits
            H_j ← H_j >> 16
        end
        else
            // repeat the last value to
            // recover the rest values in flit_i
            flit_i ← flit_i << 32 + 32-bit value (the
            last)
        end
    end
end
return flit_0, ..., flit_m
```

Algorithm 2 shows the decoding process. If there is only one approximable flit, the values stored in the head flit will be applied directly to recovering the approximable flit. For more than one approximable flit, the data space in the head flit is divided into 2 or 4 or 8 parts based on the approximable quantity. Each part can recover an approximable flit. Different divisions result in different sizes being used for recovering a flit. For example, 2 divisions will result in each part have 64 bits to recover a flit, while for 8-division scenarios, there are only 16 bits in each division. Each 16 bits of a division can recover a 32-bit data in the flit. The 4 data of a flit will require

TABLE 1. NoC configurations.

	SCARAB	ABNoC
Channel Width	128 bits	149 bits
Routing Algorithm	Adaptive	XY
Data Packet	8 flits	9 flits
Control Packet	1 flits	
Pipeline Stages	2	
Network Size	8×8	

64 bits to recover. If there are not enough bits to recover the flit. The remaining data in the flit can be restored to the last decoded data. The decoding that recovers a 32-bit data from 16 bits in a division is the opposite process of encoding. It pads the missing bits with zeros.

VI. METHODOLOGY

We evaluate the effectiveness of the ABNoC in comparison with the outstanding NACK-based bufferless network (SCARAB) [10] and a compression-based approximate design (APPROX-NoC) [13]. The network configuration parameters are listed in Table 1.

1) NoC CONFIGURATIONS

We compare ABNoC against a non-approximate bufferless NoC and an approximate bufferless NoC. The configurations are listed below:

- **SACRAB**: In this configuration, the NoC is the non-approximate bufferless SCARAB. The workload is uncompressed and achieves losslessly delivery. All the data packets are filled with 8 flits. As detailed in [10], each flit needs to perform two stages (switch traversal and link delay) in each SCARAB router. For purposes of fair comparison, the SCARAB network referenced in this paper is designed with priorities but without the opportunistic buffering, as the opportunistic buffering needs router-side buffers and could likewise be applied to the other bufferless design for reducing retransmission latency.
- **SACRAB_APPROX**: There is no approximate design in bufferless NoC so far. We apply the approximation framework APPROX-NoC [13] to the SACRAB network. In this configuration, the network is the non-approximate SCARAB, but the injection packets are compressed and damaged. Following the APPROX-NoC design, all the data packets are compressed with the frequent-pattern approximate compression technique which is called FP-VAXX. Its error threshold is set as 10%. The compression latency is three cycles and the decompression latency is two cycles, which is the same as the default configuration in APPROX-NoC.
- **ABNoC**: This is our proposed approximate bufferless NoC. All the data packets are equipped with an extra encoded head flit, but the data are uncompressed. The flits in ABNoC contain more 21-bit header bits for routing and port allocation and routers have the same channel width as the flit size.

2) SYNTHETIC TRAFFIC

We evaluate the average latency and average number of retransmissions in NoCs under several synthetic traffic patterns. We use a cycle-accurate, in-house NoC simulator. All injected packets are multi-flit packets, and we randomly select 50% of the packets to be approximable. A varying percentage of approximable flits are also studied to show their impact on the performance of ABNoC. As stated in [13], the APPROX-NoC enhances the compression ratio by up to 41% and increases the non-approximate compression ratio by 30%. Therefore, we reduce the approximable packet by 3 flits and the non-approximable packet by 2 flits. The approximable data packet is compressed into a 5-flit packet, while the non-approximable ones become 6-flit packets.

We also evaluate the arrival rate in ABNoC to show the percentage of received flits. Its value is the ratio of the number of received flits (not including decoding flits) to the number of input flits. The arrival rate determines the output quality. A high arrival rate means that only a small amount of flits will be approximated. Bandwidth is another important parameter for the NoCs. It is the injection rate where the network latency becomes excessive. A large bandwidth means low packet latency at high injection rates. In this paper, we evaluate the bandwidth as the maximum injection rate where network latency is less than 100 cycles.

3) FULL-SYSTEM SIMULATION

To evaluate the impact of our ABNoC mechanism on the overall application output error, we utilize the Pin tool, a dynamic binary instrumentation framework [20]. We hand-annotate the benchmarks mentioned below, in a similar fashion to the load value approximation [15], to identify the data regions that can be approximated. The ABNoC only focuses on integer and floating-point value approximation. An important consideration while hand-annotating approximable data regions is the data type of the variables being determined to be approximable. We annotate the flits as approximable only if their words are approximable and have the same type (integer or floating-point). In ABNoC, there is at least one approximable flit in each data packet due to the encoding and decoding algorithm. If all the flits of a data packet are non-approximable, the last flit will be annotated as approximable. For a full-system evaluation, we use a modified event-driven many-core simulator [21]. Table 2 lists the full-system simulation parameters.

4) BENCHMARKS

The benchmarks used for performance evaluation are selected from the PARSEC [22]. We configure each application with 64 threads and run the benchmarks for 100 million instructions with small input size. All the simulation benchmarks and their quality metrics are listed in Table 3.

TABLE 2. Full–system simulation system configurations.

Number of cores	64 (x86 ISA)
L1 D cache (private)	128 KB, 64-B lines, 4 cycles
L1 I cache (private)	128 KB, 64-B lines, 4 cycles
L2 cache (shared)	512 KB slice/node, 64-B lines, 8 cycles
Caching protocol	MESI
DRAM controllers	4 (on the node 0, 16, 32, 48)
DRAM size	2 GB, 45 cycles

TABLE 3. Benchmarks.

Name	Quality Metric
blackscholes	Percentage of results with error greater than 1%
bodytrack	Average error of the output vectors
canneal	Relative error of the final routing
fluidanimate	Percentage of particles with location error greater than 0.2%
raytrace	Average error of the output vector
streamcluster	Percentage of points that are not accurately clustered
swaptions	Average error of the output prices
vips	Average relative error of final output RGB image's pixel values
x264	The peak signal-to-noise ratio

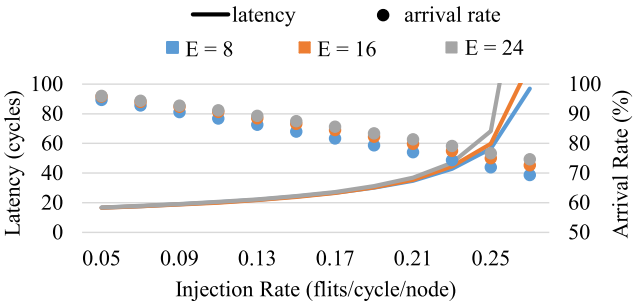


FIGURE 17. Maximum count of injection counter sensitivity analysis.

VII. EXPERIMENTAL RESULTS

A. DESIGN PARAMETERS EXPLORATION

1) INJECTION PERIOD

In ABNoC, packet injection period needs to be limited in E cycles. Figure 17 shows the average packet latency and the arrival rate in ABNoC by varying the limited injection period (E). All the packets are injected randomly. At low injection rates, the average packet latencies with different E values have hardly changed. When the injection rate exceeds 0.2, with E decreases, the average latency and the arrival rate weakly decreases. The reason is that the reduction of E results in more flits to be discarded before injection, thereby the waiting time before injection will be reduced. At a low injection rate, there is little waiting time before injection and the reduction of E reduces the arrival rate and achieves little latency benefit. In ABNoC, considering the arrival rate and the latency benefit, the default value of E is set as 16.

2) NACK CHANNELS

Generally, multiple NACK channels are to meet the high injection rate requirements. We evaluate an 8×8 network with 4, 8, 12, and 16 logical NACK channels. Figure 18 shows the percentage of failed transmissions due to the lack of the NACK channel. The workload is random single-flit traffic, so that each flit applies a dedicated NACK channel. As the injection rate increases, the NACK channel competition becomes increasingly more intense. At a low injection rate, a small number of NACK channels can satisfy NACK channel contention, whereas it results in a large amount of packet drops due to NACK channel contention at a high injection rate. In our tests, when the NACK network is equipped with 16 logical channels in each port, there is little packet drop due to lack of NACK channel. Therefore, we choose the 16 logical NACK channels in ABNoC. In physical design, there are 8 NACK channels per port due to the TDM design.

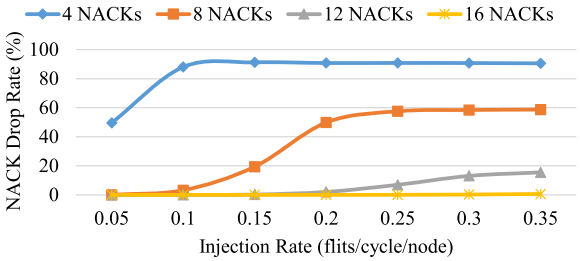


FIGURE 18. Percentage of flits dropped due to lack of the NACK channel under a random single-flit traffic workload.

B. PERFORMANCE ANALYSIS

In this section, we present the NoC level performance evaluation of the ABNoC using simulated workloads with different patterns.

1) AVERAGE LATENCY AND ARRIVAL RATE

We evaluate the average latency of ABNoC on a random and tornado traffic pattern. Figure 19(a) and Figure 19(b) show the results comparing with the SCARAB and SCARAB_APPROX. ABNoC shows a great decrease in average latency compared to SCARAB and SCARAB_APPROX. ABNoC achieves the lowest average packet latency and the highest bandwidth. The average packet latency of SCARAB_APPROX is higher than that of SCARAB when the injection rate is very low. It is due to the SCARAB_APPROX needs more time for compression and decompression packets. At a high injection rate, SCARAB_APPROX reduces the queuing time at source node and slightly reduces network conflicts due to fewer injections. The ABNoC is dedicated to reducing network conflicts by allowing the approximable collision flits to be discarded without retransmission. Therefore, as the injection rate increases, the network conflict intensifies, ABNoC achieves great benefits in latency reduction. Under random workloads, ABNoC achieves an approximate 1.92× and 1.47× bandwidth increase compared to SCARAB and SCARAB_APPROX. For the tornado traffic pattern, ABNoC increases the bandwidth by 1.73× with respect to SCARAB and 1.27× compared to SCARAB_APPROX.

Figure 19(a) and Figure 19(b) also show the arrival rate in the ABNoC under random and tornado traffic patterns. Across the simulation workloads, the arrival rate is always

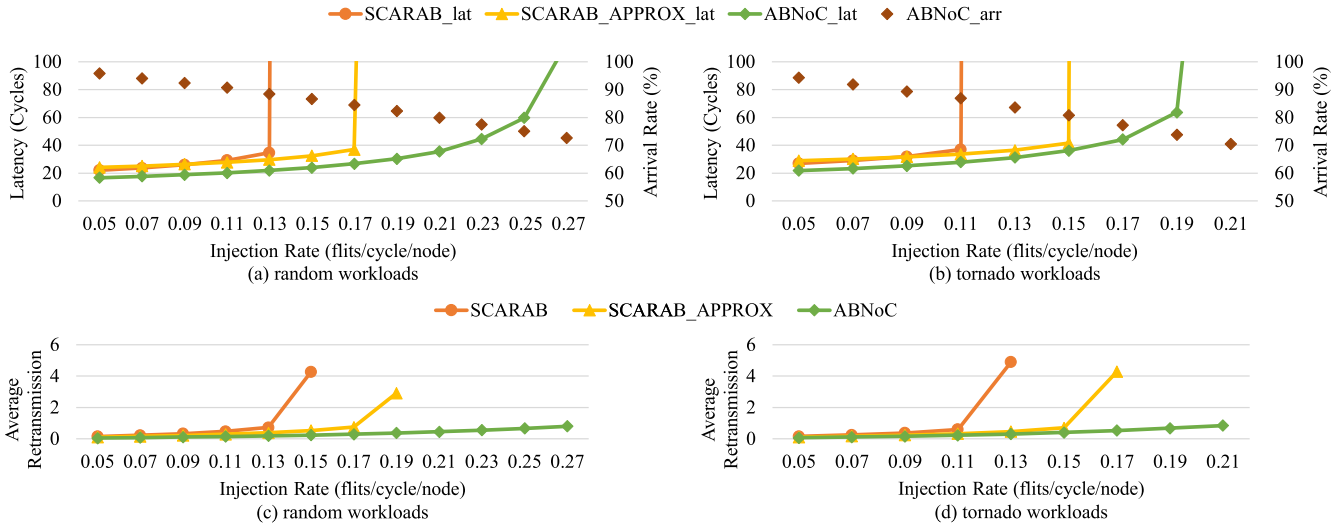


FIGURE 19. Throughput and retransmission analysis under random and tornado traffic patterns.

higher than 70% even when the network saturates. This means that most of the flits are delivered without loss, even for the approximable flits.

Packet Retransmission is an important parameter affecting network performance in NACK-based bufferless NoCs. The retransmitted packets could exacerbate network conflicts. Figure 19(c) and Figure 19(d) show the results of the average number of retransmissions. It shows that the ABNoC can greatly reduce packet retransmissions. The average number of retransmissions in ABNoC is very small even when network throughput has been saturated. ABNoC can greatly reduce network conflicts, thereby most of the packets travel through ABNoC are without retransmission. In addition, due to reducing network injections, the average number of retransmissions is also reduced by the SCARAB_APPROX. But the reductions are much smaller than that in ABNoC. The reason is that the end-to-end traffic reduction cannot immediately react to network congestions, while the in-router AAM of ABNoC can directly discard the conflicting flits.

2) SENSITIVITY

We show the sensitivity of the percentage of approximable flits. Figure 20 shows the average packet latency and the arrival rate of the ABNoC as the percentage of approximable flits is varied by 25%, 50% (default) and 75%. The arrival rates are plotted until network saturation. Obviously, the throughput increases and the arrival rate decreases as the percentage of approximable packets increases. The reason is that a high approximable ratio increases the chances of flit discarding without retransmission. Furthermore, for the arrival rates of the 50% and 75% approximable ratios, there is little difference between them. It is due to that at the 50% approximable ratio, network conflicts have been greatly reduced in ABNoC. The arrival rate will be mainly determined by the injection rate and the workload pattern.

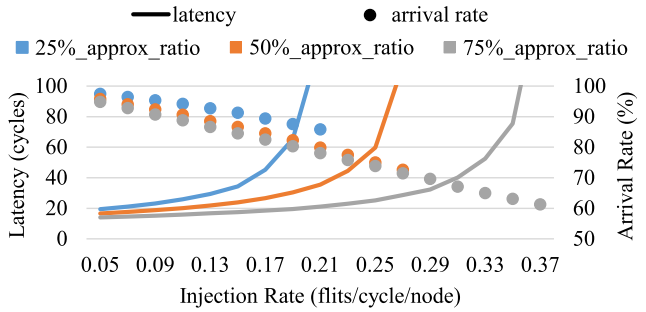


FIGURE 20. Approximable flits ratio sensitivity analysis.

At a higher approximable ratio, the arrival rate in ABNoC changes very little.

C. FULL SYSTEM SIMULATION

In this section, we evaluate the performance of ABNoC in a full-system simulation. We present the average packet latency, average number of retransmissions, system runtime and application error on a range of PARSEC benchmarks.

1) OVERALL SYSTEM PERFORMANCE

We analyze the impact of ABNoC on the average network latency and overall system speedup. Figure 21(a) shows the normalized average latency for benchmarks compared against the SCARAB and the SCARAB_APPROX. The average latency is normalized to SCARAB. Across the benchmarks, ABNoC reduces the latency by average 34.6% with respect to SCARAB_APPROX and by average 46.7% to SCARAB. This is mainly because the approximate transmission of ABNoC brings more reduction in the number of packet retransmissions, leading to latency benefits. Figure 21(b) shows the average number of retransmissions of benchmarks. Obviously, ABNoC greatly reduces packet retransmissions for all the benchmarks. On average, ABNoC reduces packet retransmissions by 83.6% with

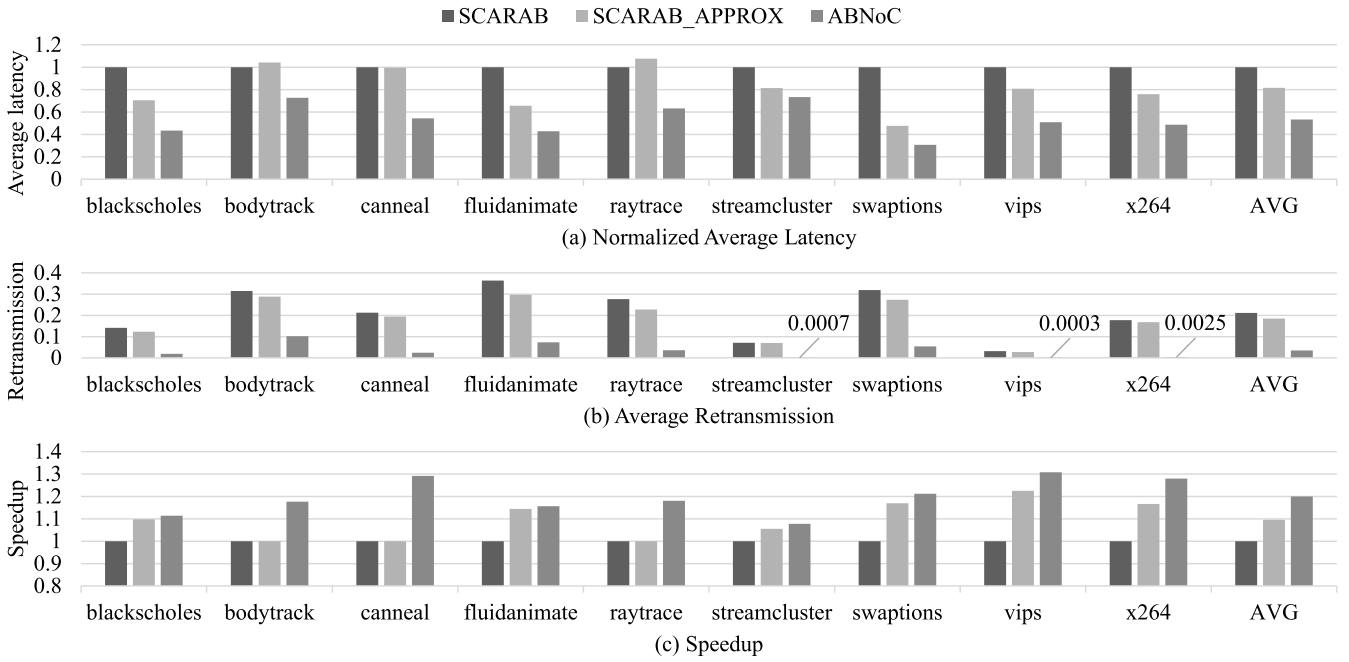


FIGURE 21. Network latency, retransmission, and system performance for full-system analysis.

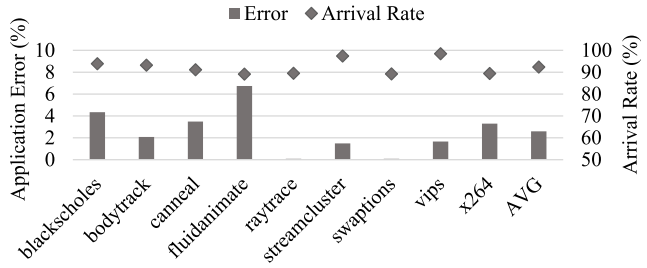


FIGURE 22. Benchmark result accuracy and arrival rate in ABNoC.

respect to SCARAB and 81.3% to SCARAB_APPROX. The SCARAB_APPROX also contributes a minor reduction in the average retransmission. Thus, the SCARAB_APPROX also achieves an average 18.5% reduction in latency compared to SCARAB. However, for *bodytrack* and *raytrace*, SCARAB_APPROX slightly increases the latency, and for *canneal*, it only achieves moderate improvement. The reason is that the latency benefit of traffic reduction is offset by the compression/decompression overheads in SCARAB_APPROX.

Figure 21(c) shows how ABNoC affects overall system runtime. All the results are normalized to SCARAB. The evaluation shows that ABNoC and SCARAB_APPROX achieve an average 1.20× and 1.09× speedup compared to SCARAB, respectively. And for all the benchmarks ABNoC achieves a higher speedup than SCARAB_APPROX. This is due to the higher reduction of ABNoC in packet latency.

2) ARRIVAL RATE AND ERROR

The arrival rates and the result errors in ABNoC is shown in Figure 22. The average arrival rate is over 92%. This means that ABNoC delivers most of the flits without loss. Besides, we analyze the relative error of the encoding algorithm in ABNoC. For integer value and floating-point value, their relative error can be calculated by the Equation (5) and Equation (6).

$$error_{int} = \begin{cases} \dfrac{\sum_{k=0}^{n} X_k 2^k}{\sum_{k=0}^{n+7} X_k 2^k + 2^{n+8}}, & int < -512, int \geqslant 512 \\ 0, & -512 \leqslant int < 512 \end{cases} \tag{5}$$

where the $error_{int}$ is the relative error of a integer value; the X_k is the kth bit value of the integer value and $X_k \in \{0, 1\}$; n is the shifted bits ($1 \leq n \leq 22$).

$$error_{float} = \frac{\sum_{k=7}^{23} X_k 2^{-k}}{1 + \sum_{k=1}^{23} X_k 2^{-k}} \tag{6}$$

where the $error_{float}$ is the relative error of a floating-point value; the X_k is the kth bit value of the mantissa and $X_k \in \{0, 1\}$. Therefore, according to the summation of geometric progression,[3] the maximum error of a integer value must be less than 0.39% and the maximum relative error of a floating-point value encoding is less than 1.56%.[4] These relative errors are very low. In addition, The data discarded in encoding are adjacent to the encoded data in the cache. They could be very similar. Thus, they could be recovered by the encoded data with low error. Besides, most of the flits does not need to be approximated due to the high arrival rate

[3] $\sum_{k=1}^{n} q^{k-1} = (1 - q^n)/(1 - q)$, where n is the number of terms, and q is the common ratio in the sequence.

[4] $error_{int} < 2^{-8}$, $error_{float} < 2^{-6}$

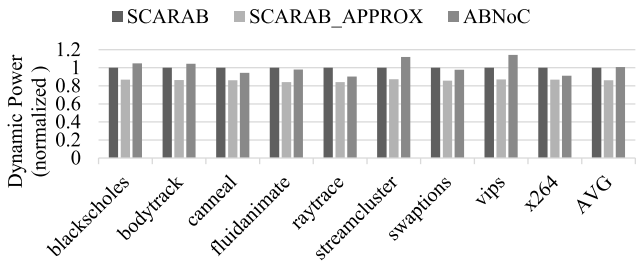


FIGURE 23. Normalized dynamic power consumption analysis.

in ABNoC. Evaluations shows that all applications have an error of less than 7%.

D. POWER CONSUMPTION AND AREA OVERHEAD

In this section, we evaluate the effect of ABNoC on the network power consumption and area overhead. To show the variation in dynamic power consumption among SCARAB, SCARAB_APPROX and ABNoC, we depict the dynamic power of different benchmarks in Figure 23 using the Orion power simulator [23]. All the results are normalized to SCARAB. In ABNoC, some benchmarks (*canneal, fluidanimate, raytrace, swaptions, x264*) consume less dynamic power than that in SCARAB. This is primarily attributed to the reduction in packet retransmissions. However, the others increase their dynamic power consumptions. The main reason is that flits in ABNoC contain more 21-bit bits than that in SCARAB and need more energy to pass a router. When the increase in power consumption is larger than the power savings of retransmission reduction, the dynamic power will increase. For an average comparison, ABNoC has a similar dynamic power consumption to SCARAB. SCARAB_APPROX achieves dynamic power reduction for all the benchmarks compared to SCARAB, It is due to the reduction in the number of injected flits.

To ensure accurate hardware modeling, the ABNoC router is implemented in RTL based on the open source RTL router design [24]. We use the Synopsys design compiler with TSMC 65nm technology to evaluate the power and area. The results of SCARAB router power and area are cited from [10]. For purposes of fair comparisons with SCARAB, the injection and ejection buffers are also designed to use MSHRs in the ABNoC router, which is the same as the SCARAB design. We also compare with a regular buffered router that equips with 5 ports (128-bit channel width), 4 virtual channels (4-flit each) per port and a 3-stage pipeline based on the XY routing. All the routers work at a 1.9GHz frequency (same as the SCARAB). And we assume a constant uniform load on all input ports (same as the SCARAB). The results of router power and area are listed in Table 4. Compared with SCARAB, the ABNoC router has slightly more consumption in terms of power and area. It incurs 5.0% more power consumption and 14.1% more area overhead. It is due to the more bits of channel width in ABNoC and the additional packet approximation design. However, these increments are much smaller relative to the area and power overhead in the buffered design. ABNoC reduces 76.2% power consumption and 88.9% area overhead compared to the buffered router. ABNoC still maintains the area and power benefit of bufferless design.

TABLE 4. Router power and area comparison.

	Power	Area
SCARAB	$46.2mW$	$34.1\text{K}\mu m^2$
ABNoC	$48.5mW$	$38.9\text{K}\mu m^2$
Buffered	$194.1mW$	$307.3\text{K}\mu m^2$

VIII. RELATED WORK

A. APPROXIMATE NoC DESIGNS

Recent studies have been conducted regarding approximate computing in NoC architecture design for applications that allow inaccurate outputs [13], [14], [19], [25]–[28]. These articles explore the performance improvement or energy efficiency of approximate computing for reducing communication bottlenecks by two techniques: communication reduction and dynamic power management. The APPROX-NoC [13], ABDTR [14], DAPPER [19] and DEC-NoC [25] belong to the former. APPROX-NoC reduces injected flits by approximate compression, which improves the compression ratio based on a value approximation matching technique. ABDTR proposes approximate communication to address congestion in NoCs, which selectively discards approximable data that may affect network congestions before injecting. DAPPER coalesces approximable data waiting for transmission to reduce the number of injected packets in the NoC of the GPGPU architecture. DEC-NoC relaxes transmission accuracy in order to reduce retransmissions due to communication errors. These approximation designs focus primarily on the data being transmitted and do not change packet transmissions in the NoC. Ahmed *et al.* [26] and Ascia *et al.* [27], [28] use dynamic power management to transfer approximable data at lower power in order to trade the result quality for energy savings. In AxNoC [26], a dual-voltage look-ahead scheme is proposed for the power management in approximate NoC design, which isolates a critical path for providing headers and critical flits with perfect transmission under high voltage and the remaining flits with bit flips by decreasing the supply voltage. In [27], Ascia *et al.* selectively reduce the voltage swing of a link based on the forgiving nature of the approximable communication flow for a trade-off between quality degradation and energy saving. In [28], Ascia *et al.* introduce a dynamic power management into approximate wireless NoC design, in which both wireless and wired communications are controlled based on their expected reliability levels. Approximable communications will be applied with low power and transferred with low expected reliability. These approximate NoCs achieve a high energy benefit with dynamic power management, but exert little effort in network contention. Channel contending during transmission greatly reduces network performance, especially in bufferless NoCs. We propose the approximation mechanism in bufferless NoC.

It reduces channel competitions by discarding the approximable flits in transmission conflicts and approximating the dropped flits at the destination node and hence improves performance of a bufferless network. Betzel *et al.* [29] and Reza and Ampadu [30] summarize the approximate techniques in NoCs and present a bright future of approximate communication in energy-efficient and high-performance NoC design.

B. BUFFERLESS NoC DESIGNS

have received significant research attention [8]–[11], [31]–[35]. BLESS [11] and CHIPPER [9] are designed based on deflection routing. Kim *et al.* [32] and Daya *et al.* [34] improve deflection routing of bufferless networks with clumsy flow control and virtual express paths, respectively. The BPS [31] is a hybrid-solution bufferless NoC that combines both deflection routing and an NACK-based mechanism, which rolls back packets through loopback channels for retransmission. The SCARAB [10] is an optimized NACK-based design that transfers packets with a single-cycle latency pipeline and issues NACK messages through a physically separate network upon contentions. Zhao *et al.* [33] propose a heterogeneous NoC architecture using both buffered routers and bufferless routers for a low-cost design. In the Chameleon [35] and Runahead [8] network, which are multiple-NoCs, a bufferless NoC working as a subnetwork must be paired with a buffered NoC for an energy-efficient design. Our ABNoC operates by first utilizing an approximation engine in an NACK-based bufferless NoC to reduce retransmissions, and it achieves throughput and bandwidth improvement with a low amount of quality degradation.

IX. CONCLUSION

In this paper, we propose the ABNoC, a hardware approximation design relaxing transmission accuracy in exchange for high-performance bufferless NoC. Packet retransmission shows severe impact on performance of the NACK-based bufferless NoC. ABNoC provides an in-network traffic reduction scheme to reduce packet retransmissions and network conflicts. It can discard the conflicting approximable flits during the deliveries based on the AAM design and recovers them after packet transmission via the packet approximate method. In addition, with a separate preconfigured circuit-switched NACK network, ABNoC can send an NACK message to trigger a retransmission of a packet with non-approximable flits loss. Evaluations on synthetic traffic show that ABNoC increases the bandwidth up to 1.92× compared to SCARAB. And compared to the compression-based approximation framework (SCARAB_APPROX), ABNoC also achieves up to 1.47× bandwidth improvement. For a range of PARSEC benchmarks, ABNoC reduces the average packet latency by 28.2% with respect to SCARAB_APPROX and 46.7% compared to SCARAB. Besides, it achieves an average 1.20× speedup compared to SCARAB, while maintaining less than 7% application error. As a bufferless design, ABNoC also achieves 76.2% and 88.9% savings in power and area respectively compared to the buffered design. ABNoC reduces packet latency and increases bandwidth of bufferless network, while maintaining low application error and small area and power overhead. These results confirm that ABNoC is a promising deign to improve bufferless NoC performance in applications that exhibit some level of error tolerance.

REFERENCES

[1] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. 44th Annu. Design Autom. Conf.*, 2007, pp. 746–749.

[2] S. Bell *et al.*, "TILE64-processor: A 64-core SoC with mesh interconnect," in *Proc. IEEE Int. Symp. Solid-State Circuits Conf.*, Feb. 2008, pp. 88–598.

[3] A. Agarwal, C. Iskander, and R. Shankar, "Survey of network on chip (NoC) architectures & contributions," *J. Eng., Comput. Archit.*, vol. 3, no. 1, pp. 21–27, 2009.

[4] S. Borkar, "Future of interconnect fabric: A contrarian view," in *Proc. 12th ACM/IEEE Int. Workshop Syst. Level Interconnect Predict.*, Jun. 2010, pp. 1–2.

[5] M. B. Taylor, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "Evaluation of the raw microprocessor: An exposed-wire-delay architecture for ILP and streams," *ACM SIGARCH Comput. Archit. News*, vol. 32, no. 2, p. 2, 2004.

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

[7] B. K. Daya, C.-H. O. Chen, S. Subramanian, W.-C. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L.-S. Peh, "SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit. (ISCA)*, Jun. 2014, pp. 25–36.

[8] Z. Li, J. S. Miguel, and N. E. Jerger, "The runahead network-on-chip," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2016, pp. 333–344.

[9] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2011, pp. 144–155.

[10] M. Hayenga, N. E. Jerger, and M. Lipasti, "SCARAB: A single cycle adaptive routing and bufferless network," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2009, pp. 244–254.

[11] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2009, pp. 196–207.

[12] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62-1–62-33, Mar. 2016.

[13] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim, "APPROX-NoC: A data approximation framework for network-on-chip architectures," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 666–677.

[14] L. Wang, X. Wang, and Y. Wang, "ABDTR: Approximation-based dynamic traffic regulation for networks-on-chip systems," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 153–160.

[15] J. S. Miguel, M. Badr, and N. E. Jerger, "Load value approximation," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2014, pp. 127–139.

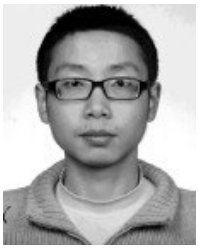
[16] A. Yazdanbakhsh, G. Pekhimenko, B. Thwaites, H. Esmaeilzadeh, O. Mutlu, and T. C. Mowry, "RFVP: Rollback-free value prediction with safe-to-approximate loads," *ACM Trans. Archit. Code Optim.*, vol. 12, no. 4, 2016, Art. no. 62.

[17] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "EnerJ: Approximate data types for safe and general low-power computation," *ACM SIGPLAN Notices*, vol. 46, no. 6, pp. 164–174, Jun. 2011.

[18] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *Proc. 17th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, New York, NY, USA, 2012, pp. 301–312. [Online]. Available: http://doi.acm.org/10.1145/2150976.2151008

[19] V. Y. Raparti and S. Pasricha, "DAPPER: Data aware approximate NoC for GPGPU architectures," in *Proc. 25th IEEE/ACM Int. Symp. Netw.-on-Chip (NOCS)*, Piscataway, NJ, USA: IEEE Press, 2018, Art. no. 7. [Online]. Available: http://dl.acm.org/citation.cfm?id=3306619.3306626

[20] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proc. ACM SIGPLAN Conf. Program. Lang. Design Implement. (PLDI)*, New York, NY, USA, 2005, pp. 190–200. [Online]. Available: http://doi.acm.org/10.1145/1065010.1065034
[21] X. Wang, M. Yang, Y. Jiang, P. Liu, M. Daneshtalab, M. Palesi, and T. Mak, "On self-tuning networks-on-chip for dynamic network-flow dominance adaptation," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2s, 2014, Art. no. 73.
[22] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, New York, NY, USA, 2008, pp. 72–81. [Online]. Available: http://doi.acm.org/10.1145/1454115.1454128
[23] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proc. 35th Annu. ACM/IEEE Int. Symp. Microarchitecture (MICRO)*, Los Alamitos, CA, USA: IEEE Computer Society Press, Nov. 2002, pp. 294–305. [Online]. Available: http://dl.acm.org/citation.cfm?id=774861.774893
[24] D. U. Becker, "Efficient microarchitecture for network-on-chip routers," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Palo Alto, CA, USA, 2012.
[25] Y. Chen, M. F. Reza, and A. Louri, "DEC-NoC: An approximate framework based on dynamic error control with applications to energy-efficient NoCs," in *Proc. IEEE 36th Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 480–487.
[26] A. B. Ahmed, D. Fujiki, H. Matsutani, M. Koibuchi, and H. Amano, "AxNoC: Low-power approximate network-on-chips using critical-path isolation," in *Proc. 25th IEEE/ACM Int. Symp. Networks-on-Chip (NOCS)*, Piscataway, NJ, USA: IEEE Press, Oct. 2018, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=3306619.3306625
[27] G. Ascia, V. Catania, S. Monteleone, M. Palesi, D. Patti, and J. Jose, "Improving energy consumption of NoC based architectures through approximate communication," in *Proc. 7th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2018, pp. 1–4.
[28] G. Ascia, V. Catania, S. Monteleone, M. Palesi, D. Patti, and J. Jose, "Approximate wireless networks-on-chip," in *Proc. Conf. Design Circuits Integr. Syst. (DCIS)*, Nov. 2018, pp. 1–6.
[29] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, and U. Karpuzcu, "Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems," *ACM Comput. Surv.*, vol. 51, no. 1, Apr. 2018, Art. no. 1. [Online]. Available: http://doi.acm.org/10.1145/3145812
[30] M. F. Reza and P. Ampadu, "Approximate communication strategies for energy-efficient and high performance NoC: Opportunities and challenges," in *Proc. Great Lakes Symp. VLSI (GLSVLSI)*, New York, NY, USA, 2019, pp. 399–404. [Online]. Available: http://doi.acm.org/10.1145/3299874.3319455
[31] C. Gómez, M. E. Gómez, P. López, and J. Duato, "Reducing packet dropping in a bufferless NoC," in *Proc. Eur. Conf. Parallel Process.* Berlin, Germany: Springer, 2008, pp. 899–909.
[32] H. Kim, Y. Kim, and J. Kim, "Clumsy flow control for high-throughput bufferless on-chip networks," *IEEE Comput. Archit. Lett.*, vol. 12, no. 2, pp. 47–50, Jul. 2013.
[33] H. Zhao, M. Kandemir, W. Ding, and M. J. Irwin, "Exploring heterogeneous NoC design space," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, Piscataway, NJ, USA: IEEE Press, Nov. 2011, pp. 787–793. [Online]. Available: http://dl.acm.org/citation.cfm?id=2132325.2132496
[34] B. K. Daya, L. Peh, and A. P. Chandrakasan, "Towards high-performance bufferless NoCs with SCEPTER," *IEEE Comput. Archit. Lett.*, vol. 15, no. 1, pp. 62–65, Jan./Jun. 2016.
[35] J. Wu, D. Dong, X. Liao, and L. Wang, "Chameleon: Adaptive energy-efficient heterogeneous network-on-chip," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, Oct. 2015, pp. 419–422.



LING WANG received the B.S. degree in monitoring and control technology from the Harbin University of Science and Technology, China, in 2010, and the M.S. degree in biomedical engineering from the Harbin Institute of Technology, China, in 2012, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests include high-performance many-core architecture and bioinformatics applications.

XIAOHANG WANG received the B.Eng. and Ph.D. degrees in communication and electronic engineering from Zhejiang University, in 2006 and 2011, respectively. He is currently an Associate Professor with the South China University of Technology. His research interests include many-core architecture, power efficient architectures, optimal control, and NoC-based systems.

YADONG WANG is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology. He is also the Dean of the School of Computer Science and Technology, the Director of the Center for Biomedical Information Technology and Software Systems of Heilongjiang Province, and the Director of the Bioinformatics and Computational Biology Key Lab of Heilongjiang Province. His research is focused on high-performance computing, bioinformatics, machine learning, and knowledge engineering. He is a member of the Chinese Computer Association.