Flexible and Efficient QoS Provisioning in AXI4-Based Network-on-Chip Architecture

Boqian Wang and Zhonghai Lu, *Senior Member, IEEE*

***Abstract*—We propose a network-on-chip (NoC)-based whole system design, whose communication architecture is compatible with the advanced microcontroller bus architecture advanced extensible interface 4 (AXI4) protocol and supports high-performance multiple quality-of-service (QoS) schemes. In our system, the network interface (NI) between the NoC and the master/slave node is proposed to make the NoC architecture independent from the AXI4 protocol via message format conversion between the AXI4 signal format and the packet format, offering high flexibility to the NoC design. Besides, a QoS inheritance mechanism is applied in the slave-side NI to support QoS during packets' round-trip transfer in the NoC. The NoC system contains time-division multiplexing (TDM) and virtual channel (VC) subnetworks to apply multiple QoS schemes to AXI4 signals with different QoS tags, and the NI is responsible for signals distribution between two subnetworks. Besides, a traffic converter is proposed in each NI to balance the traffic between the two subnetworks when necessary. The experimental results show that our proposed architecture ensures a high-throughput and low-latency NoC system. By applying the traffic converter, the packet latency can be improved.**

***Index Terms*—Advanced extensible interface 4 (AXI4), computer architecture, network-on-chip (NoC), quality of service (QoS), system-on-chip (SoC).**

I. Introduction

ADVANCED extensible interface 4 (AXI4) is the fourth generation of the advanced microcontroller bus architecture (AMBA) interface specification from ARM, which supports high-performance, high-frequency system designs [1]. It uses five individual channels for communication between master and slave interfaces and includes specific signals for quality-of-service (QoS) schemes. It was initially a bus-oriented protocol. But with the development of semiconductor technology, the number of cores increases, requiring a more efficient interconnect architecture for high-throughput communication. In this context, network on chip (NoC) provides a potential alternative, which can support multiple QoS and enhance the communication among multiple cores [2]–[4].

Manuscript received September 2, 2020; revised January 2, 2021 and March 29, 2021; accepted May 30, 2021. Date of publication June 22, 2021; date of current version April 21, 2022. This article was recommended by Associate Editor L. Benini. *(Corresponding author: Zhonghai Lu.)*

Boqian Wang was with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 164 40, Kista, Sweden. He is now with the Beijing Institute of Biotechnology, Beijing 100 071, China.

Zhonghai Lu is with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 164 40, Kista, Sweden (e-mail: zhonghai@kth.se).

Digital Object Identifier 10.1109/TCAD.2021.3091410

However, supporting the AXI4 protocol and its required multiple QoS schemes in the NoC-based architecture is still a challenge. There are some prior system architectures supporting AXI4, such as [5]–[8]. But none of them show the details in the network interface (NI) between the NoC and the master/slave node, which should have included the description of the NI architecture and the message format conversion between the AXI4 signal format in the master/slave node and the packet format in the NoC. Besides, their NoC architectures do not consider the support of QoS scheme, which could be required by the AXI4 signal. In terms of QoS provisioning, related works, such as [9]–[12], only support one or two QoS schemes, which are not enough for the multiple requirements of different processors or devices. For example, CPU is latency critical, GPU requires guaranteed bandwidth, and the SATA device only needs best-effort transfer.

In this article, we propose an AXI-based NoC architecture to provide three different QoS schemes. In summary, our work has the following contributions.

1) We design a message format conversion process between the AXI4 signal format and the packet format in the NI, making the NoC design independent from the AXI4 protocol.
2) We define three different QoS services and design a NoC-based communication architecture with two subnetworks to efficiently support them. The subnetwork switch and QoS inheritance mechanism are applied for the implementation.
3) We propose a traffic converter unit in each NI to smartly distribute packets from one subnetwork with heavy traffic congestion to the other subnetwork with less traffic congestion, improving the NoC performance.
4) We propose three different flow control mechanisms in the VC subnetwork and we compare their performance results in packet latency in the experimental part.
5) We build up a cycle-accurate simulator to simulate the behavior of our proposed system. We show the performance evaluation results in terms of system throughput and packets' transfer latency and queuing delay.
6) In the simulator, we propose a two-level Markov modulated process (MMP)-based traffic generator for better simulation of realistic process/thread in a node. The traffic generation results are reported in our experiments.

II. Related Work

A. AXI4-Based Communication Support

In order to support AXI4 in communication architectures, in [5], Kwon *et al.* utilized an in-network reorder buffer to satisfy the in-order packet delivery requirement and move the reorder buffer resource and related functions from NI to network routers to increase the utilization of reorder buffer resource. Yang *et al.* [6] proposed an AXI-compliant on-chip NI architecture to offer the transaction reordering processing. Ebrahimi *et al.* [13] proposed a hybrid NI architecture with reorder buffer sharing in both the memory side and the processor side for NoCs. Neishaburi and Zilic [14] proposed a debug-aware NI, which is compatible with the AXI standard. There are other works focusing on the NI design between the interconnect and AXI-based master cores to support protocol conversion. Ramirez *et al.* [7], Tidala [15], and Nguyen *et al.* [16] took the NoC interconnect into consideration to design an AXI-based system in FPGA. In these related works, they show the design results of the NoC-AXI interface and the performance of the whole system. Radulescu *et al.* [8] proposed an on-chip NI offering backward compatibility with existing bus protocols, such as AXI, etc. Liao *et al.* [17] implemented the AXI protocol on the 3-D system on chips (SoCs) to allow high-performance communication and scalability of design.

In this aspect, most works focus on satisfying the transactions' ordering requirements of AXI4, but they did not consider the signal format conversion between the AXI4 protocol and the NoC protocol. This should be the most important part to adapt the AXI4 protocol in a NoC-based communication architecture. Besides, they did not mention the support of different QoS schemes in NoC, which could be required in many application scenarios, for example, the baseband processing in the 5G station [18], [19].

B. QoS Provisioning in NoCs

From the perspective of QoS in communication architectures, most research works focus on the support of only one or two QoS schemes in the interconnect architecture design. Sharifi *et al.* [10] offered a higher priority to packets that will access idle memory banks and at the same time, give higher priority to response packets with high latency. Chen *et al.* [9] focused on the DRAM access packets. It models the round-trip latency prediction, based on which different priorities are offered to packets in the DRAM access process. By the priority scheme, service could be offered to the latency-critical packets, which will influence the system performance on a large scale. Liu *et al.* [11] proposed a highway-based time-division multiplexing (TDM) NoC. It builds a highway with special buffer queues to ideally enhance the throughput and reduce data transfer delay. Goossens *et al.* [12], Goossens and Hansson [20] added buffering function into the TDM network. By combining guaranteed and best-effort services together, the network can support two QoS schemes at the same time. These works discuss QoS support in a more general scenario. However, in this article, we design QoS schemes in a more dedicated scenario, where the AXI4 protocol, including its ordering rules, various message channels, etc., is supported.

C. Flow Control in NoCs

The flow control in NoCs determines the resource allocation principle, which influences the network performance. Such flow control is usually executed in the router. Besides, the admission control is a kind of end-point flow control, which is executed in the NI before packets are injected into the NoC.

From the perspective of the router-based flow control, Ma *et al.* [21] proposed a flow control design based on fully adaptive routing algorithms. A nonempty VC can be reallocated if the VC has enough free buffers for an entire packet in the whole packet forwarding mechanism. Since the buffers in NoCs are more precious than in off-chip networks due to the tight area and power budgets, this mechanism ensures that a reserved but nonempty VC can be reallocated to another packet. Pérez *et al.* [22] also proposed a mechanism that allows to bypass flits even if the buffers to bypass are not empty. It can be applied to wormhole and virtual cut through to reduce packet latency and power consumption. Evain and Diguet [23] proposed a prereservation mechanism. It is realized by path coding in the NI and applied to mutual exclusive communication processes.

From the perspective of admission control, in [24], Lu *et al.* proposed (σ, ρ)-based flow regulation as a design instrument for SoC architects to control QoS and achieve cost-effective communication, which may exert significant positive impact on communication performance and buffer requirements. It also shows that the regulation spectrum can be exploited to decrease latency and backlog bounds. Jafari *et al.* [25] proposed a flow control regulation, which is performed per flow for its peak rate and burstiness. In this article, the regulation problem is formulated with buffer optimization as objectives and it comes to the conclusion that optimal flow regulation can be a highly valuable instrument for buffer optimization in NoC designs. Lu and Yao [26] used a fuzzy-algorithm-based control method to regulate the injection rate in each NI. By applying dynamic fuzzy control, the system injection rate can be regulated to a smoother level, which makes more efficient use of the system interconnect, achieving significant reduction in packet delay and improvement in system throughput. Wang *et al.* [27] proposed an artificial neural network (ANN)-based admission control for the on-chip network. It utilizes centralized ANN admission controller, which can improve system performance by predicting the most appropriate injection rate of each node via the network performance information.

D. TDM Routing Algorithm

For the TDM-based NoC, the routing algorithm is a very important consideration during the network design. Lu and Jantsch [28] utilized a two-step approach for routing in the TDM-based virtual circuit network: 1) path selection and 2) slot allocation. During the path selection step, they use a backtracking algorithm to explore the path diversity, and during the slot allocation step, path-overlapped VCs must be configured such that no conflict occurs and their bandwidth requirements are satisfied. Patil *et al.* [29] utilized a multiiteration routing algorithm to combine path and time slot selection in a single pass. Instead of the dimension-ordered routing, their approach takes any available shortest path from the source to the destination and uses multiple rip-up and reroute passes to eliminate the effects of network ordering. Kapre *et al.* [30] used two different routing algorithms: 1) greedy and 2) Pathfinder [31]. The greedy router simply allocates paths for messages on the shortest available path in space and earliest possible slot in time, avoiding congested resources, and the Pathfinder router allows routes to be allocated on congested resources and attempts to negotiate congestion through rerouting. Shpiner *et al.* [32] introduced a latency-based scheduling algorithm, which runs offline to build up the routing path. To minimize conflicts, the algorithm first schedules the packets with the longest latencies. Stefan *et al.* [33] used a distributed routing mechanism. Each router contains a slot table to store the TDM schedule and incoming packets are "blindly" routed based on this schedule. Winter and Fettweis [34] presented and evaluated different realizations of a central hardware unit, which allocates at runtime guaranteed service VCs providing QoS in packet-switched NoCs, which turns out to be very suitable for the runtime task scheduling programming model. Liu *et al.* [35] proposed a probe-based distributed solution for dynamic path searching. It can allocate paths and time slots at runtime and at the same time, be fast with predictable and bounded setup latency by a parallel probing setup method.

E. Router Microarchitecture Design

The NoC microarchitecture plays a central role in the performance of an on-chip network, especially for a buffered router, which usually focuses on the pipeline design or the buffer organization design in a router to reduce the packet transfer latency or throughput. Ramanujam *et al.* [36] proposed a new router design that aims to emulate an output-buffer router (OBR) practically, based on a distributed shared-buffer (DSB) router architecture. This technique has been successfully used in high-performance Internet packet routers [37], [38]. A DSB router uses two crossbar stages with buffering sandwiched in between, which are referred to as middle memories. The incoming packets will be assigned to one of the middle memory buffers with two constraints: 1) packets that are arriving at the same time must be assigned to different middle memories and 2) an incoming packet cannot be assigned to a middle memory that already holds a packet with the same departure time, which is tagged into each packet when they first arrive at the router. Lu *et al.* [39] proposed a low-latency wormhole router for packet-switched NoC designs. It achieves a low packet propagation latency of only two cycles per hop, including both router pipeline delay and link traversal delay, which is a significant enhancement over existing FPGA designs. Xin and Choy [40] used dynamic look-ahead bypass to reduce packet latency in NoC. In the proposed router, special look-ahead controlling pipeline is applied to speed up allocation computation so that the input buffers' bypassing rate increases. The look-ahead pipeline and bypasses can not only reduce network latency but also save the energy due to writing and reading buffers. Besides, Wang *et al.* [41] proposed a high-performance and low-cost router design based on a generic two-stage router. To realize high reliability, five fault-tolerant strategies are employed in the reliable router. Li *et al.* [42] found a way for both low latency and power efficiency by relaxing the constraint of lossless communication. In this context, they propose the Runahead NoC, a lightweight network that provides single-cycle hops.

The flow control, TDM router, and router microarchitecture design are three aspects we have considered during our NoC design. In this article, they are only related to the NoC-part design, and based on this, we elaborate the discussion of the NoC design in Section IV-D, where different QoS schemes are also provided.

F. Industrial Designs for AXI4-Based QoS

From the perspective of industrial designs, in [43], Xilinx provides QoS schemes in the AXI-based Zynq-7000 AP SoC devices. The basic QoS mechanism is a per-transaction priority based on the AXI QoS signals, which can be used by configuring the priorities. The advanced QoS mechanism can additionally provide capability to control peak rate, burstiness, and so forth. In [44], Synopsys improves QoS for AMBA-interconnect-based DesignWare IP via the combination of the QoS regulator and the QoS arbiter. The QoS regulator is applied to limit the traffic if the incoming traffic is more than the desired rate and the QoS arbiter can be configured to ensure that high-priority requests are serviced first to meet the latency requirements. It can finally provide QoS schemes to bandwidth-sensitive master (GPU), latency-sensitive master (CPU), and best-effort master (USB/SATA). Qsys interconnect is a high-bandwidth structure proposed by Intel (Altera) for connecting components. It allows to connect IP cores with various interfaces such as AMBA AXI series [45]. It utilizes different QoS components, such as the end-to-end flow controller, arbiter, bandwidth regulator, etc., to realize QoS schemes. In the 5G area, Kazaz *et al.* [18] proposed a 5G wireless infrastructure architecture where the NoC is applied to supported the communication among IP cores that follow AXI4 standards. Liß *et al.* [19] introduced and discussed the architecture of a novel networking device that provides low-latency switching and routing, where the AXI4 IP cores are integrated. It can be applied to 5G access networks, industrial networks, etc.

However, these solutions separately focus on a specific architecture in an industrial area, not providing detailed information of system architectures, the discussion and comparison among different optional solutions.

G. Compared With Our Previous Work

This article is an extension of our previous 6-page conference publication [46]. In the conference publication, we focus on the basic system design, which includes a novel NI, the definition of different QoS services, and the system

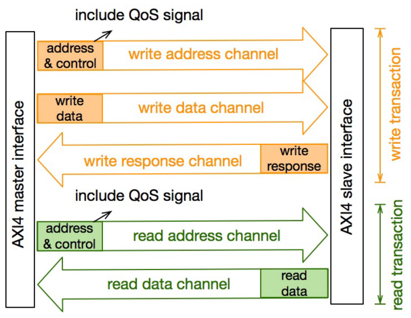

Fig. 1. Channel architecture of reads and writes.

simulator as listed in the contribution points 1), 3), and 5) in Section I. Compared with the conference publication, this article additionally includes a novel traffic conversion mechanism proposed for traffic balance, a detailed explanation for the establishment of VC (multiple flow control schemes) and TDM (static routing algorithm) subnetworks, and traffic generator controlled by a two-level MMP, which are listed in contribution points 2), 4), and 6) in Section I. Besides, a more comprehensive discussion of related work and many new experimental results related to the new mechanisms are involved. Based on the basic system design shown in the conference publication, the new contributions in this article can furthermore improve the NoC performance in terms of packet latency and throughput under an improved synthetic traffic generator, which behaves more closely to real-world traffic generation scenarios from multithread applications.

III. Background and Motivation

A. AXI4 Channels

AXI4 is a burst-based protocol, which supports high-performance, high-frequency system design. As shown in Fig. 1, the read transaction defines two independent transaction channels, which are the *read address channel* and the *read data channel*. For the write transaction, it defines three independent transaction channels, which are separately the *write address channel*, *write data channel*, and *write response channel*. All AXI4 signals will be transferred through these five channels. For signals in the read or write address channel, they include a 4-bit QoS tag to identify the corresponding transaction service they require to utilize.

B. QoS Support in AXI4-Compatible NoC

From the perspective of message format, the AXI4 signals format is quite different from the packet format in the NoC system. For example, in the virtual-channel (VC)-based wormhole network, messages will be transferred as VC packets, which will be further divided into flits, and in the TDM-based virtual circuit network, messages will be transferred as the TDM packet, which will be further divided into frames. The mismatch between AXI4 signals and NoC packets requires a message format conversion process in the NI between the AXI4-based master/slave node and the packet-based VC/TDM NoC system. When designing the NoC architecture, we should consider a reasonable method to convert the AXI4 message into packet format and allocate the NoC resources accordingly.

Besides, from the perspective of QoS, AXI4 signals may have multiple QoS requirements, such as the guaranteed bandwidth requirement, priority requirement, and best-effort requirement. However, the existing NoC architectures can only support up to two QoS schemes. So a new NoC architecture is needed to support multiple QoS requirements as well as high-performance computing.

C. AXI4 Ordering Requirements and QoS

The AXI4 protocol restricts messages' delivery sequence to master and slave nodes. Due to the possible out-of-order delivery in NoC-based communication, master-side and slave-side AXI4 ordering units are required. For the master-side and slave-side AXI4 ordering units, we have already discussed them in [47], detailing the problems of previous solutions and architectures of our proposed ordering units. On the one hand, compared with the previous solutions, our proposed mechanism can avoid the deadlock, and at the same time, provide higher performance. On the other hand, the ordering units in NIs are responsible for maintaining the ordering restriction. Thus, the designs of the router, the switch to subnetworks, and the traffic converter do not need to consider the ordering issue.

As we do not need to consider AXI4 ordering requirements anymore when designing the whole system that supports QoS schemes, we will mainly focus on the message format conversion, the QoS services, and the traffic converter design in the NI in Section IV.

IV. System Design

A. QoS Definition and Overall Architecture

We define three different QoS schemes as follows.

1) *Latency Critical Service (LCS):* It is a fast-forwarding service for burst but nonstreaming message transmission. It provides low-latency delivery service but does not guarantee delivery bandwidth for, e.g., CPU-like masters.
2) *Guaranteed Rate Service (GRS):* It is a streaming service that provides guaranteed bandwidth for large-volume flows, which request bandwidth but can tolerate latency (e.g., GPU-like masters).
3) *Unspecified Rate Service (URS):* It is a best-effort service for the delivery of certain fairness (fair treatment) to traffic. It delivers messages as soon as possible and as much as possible based on the current available resources but provides neither low-latency service nor bandwidth guarantee (e.g., SATA and USB interfaces).

Since the three QoS schemes listed above can satisfy most devices' requirements in real application scenarios, we focus on the implementation design of LCS, GRS, and URS. For the master/slave that requires multiple QoS services, their injected messages will be handled separately in the NoC with LCS, GRS, or URS scheme.

The overall system architecture is shown in Fig. 2, which can support three different QoS schemes in the AXI4-based

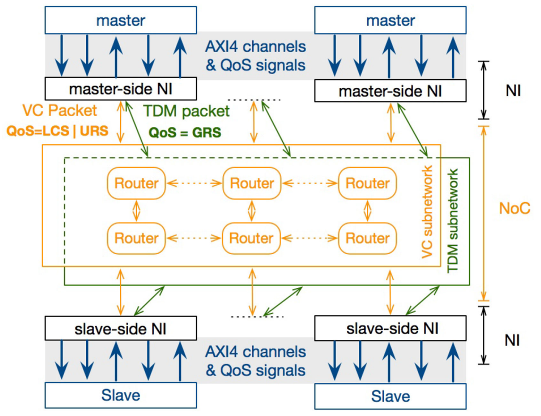

Fig. 2. Layout of the whole system architecture.

communication architecture. The whole system can be divided into three parts: 1) master/slave nodes; 2) NIs; and 3) the NoC.

The master/slave node utilizes the AXI4 protocol to communicate with the NI. For the NI, which will be detailed in Section IV-B, it realizes the protocol conversion between the AXI4 signal in the node and the packet in the master/slave NoC. The specific target format of packet in the NoC depends on the QoS identifier in the AXI4 signal. The AXI4 signal with QoS identifier LCS or URS will be packed as VC packet, transferring in the VC subnetwork. The AXI4 signals with QoS identifier GRS will be packed as TDM packet, transferring in the TDM subnetwork. This scheme makes NoC design independent from the AXI4 protocol, which, regardless of the five-channel AXI4 architecture, offers possibility of high-performance NoC design. Besides, since the AXI4 response signal does not include the QoS identifier, to support QoS for packets' round-trip transfer, we apply a QoS inheritance mechanism in the slave-side NI. In this mechanism, the response packet inherits the QoS identifier from its corresponding request packet.

For the NoC, we utilize two subnetworks, VC subnetwork and TDM subnetwork, to support three QoS schemes as listed above. The design of the NoC system is a co-consideration of hardware utilization efficiency and realistic QoS communication requirements. The detailed description and discussion will be given in Section IV-D.

B. Supporting Message Format Conversion in the NI

There are two options for message format conversion. The first option is an intuitive way, where we can convert the AXI4 signal in each AXI4 channel into a packet format and utilize five individual channels to transfer them in the NoC. In this solution, the TDM and VC subnetworks both have five independent channels separately connected to the five AXI4 channels. This scheme makes the NoC architecture adapt to the AXI4 protocol, but has less flexibility for the NoC design. Besides, the individual usage of the channel resources without share permission will decrease the resource utilization, leading to inferior NoC performance.

In another option, we can make the AXI4 protocol adapt to the traditional NoC architecture. The primary idea of this

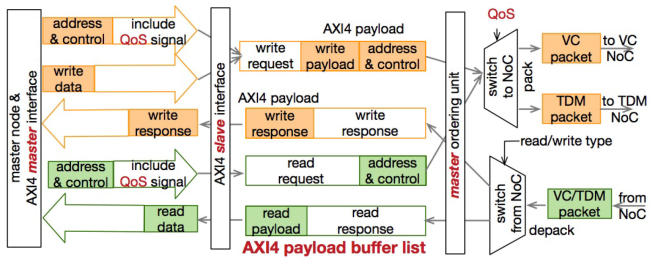

Fig. 3. Layout of the master-side NI architecture.

option is to make the NoC independent from the AXI4 protocol, so that we can design the NoC architecture individually without the restrictions from the AXI4 protocol. In this solution, the AXI4 signals will be converted into read request and response packets, and write request and response packets, transferred by the shared NoC resources. This option offers us the possibility of establishing a high-performance NoC architecture. Besides, it can also make all kinds of communication architectures, such as bus, NoC, etc., compatible with the AXI4 protocol. Thus, the second solution is favorable to build up a high-performance NoC system.

Based on the second option, our proposed NI has three functionalities. First, it receives signals from AXI4 channels and NoC subnetworks, which will be then dispatched to the corresponding AXI4 channels according to the AXI4 signal type (read or write) or to the corresponding NoC subnetwork according to the QoS identifier (LCS, URS, or GRS). Second, it is responsible for message format conversion between the AXI4 signal and NoC packet. Finally, the slave-side NI applies QoS *inheritance* mechanism to the response packet so that the QoS scheme can be applied to the packet's round-trip transfer in the NoC system.

The detailed architecture of the master-side NI is shown in Fig. 3. AXI4 signals in five AXI4 channels are forwarded to/from four AXI4 payload buffers through the AXI4 slave interface in the master-side NI. The address and control signal from the write address channel and write data signal(s) from the write data channel will be integrated together as a complete AXI4 payload stored in the write request buffer. The AXI4 signal in the write response or read address channel will be recognized as a complete AXI4 payload, which is stored in the write response or read request buffer, respectively. The read payload in the read response buffer will be divided into one or several read data signals according to the data length, and then transferred via the read data channel. The *switch to NoC* unit determines the output subnetwork of the read/write request payload according to the QoS identifier. Afterward, the corresponding routing information will be added to the AXI4 payload to support its transfer in the NoC. The *switch from NoC* unit receives packets from the two subnetworks. After removing packet routing information, it forwards the AXI4 payload to the corresponding read or write response buffer according to the read or write type.

The routing information for the packet is packed to the AXI4 payload during message format conversion process, as shown in Fig. 4. In the GRS traffic, the routing information of

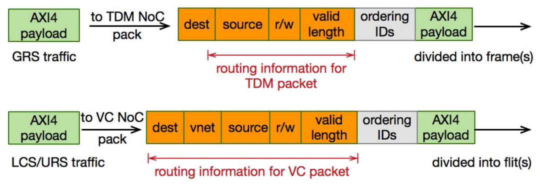

Fig. 4. Message format conversion process.

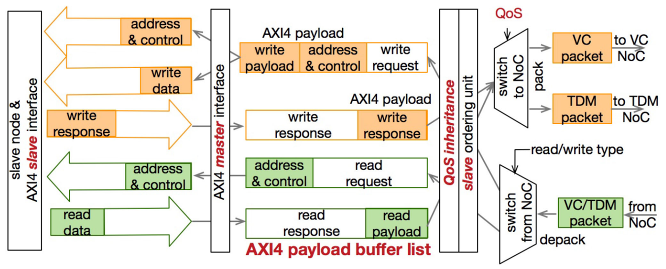

Fig. 5. Layout of the slave-side NI architecture.

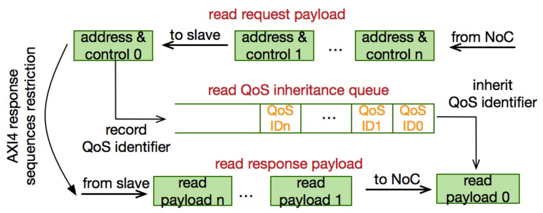

Fig. 6. QoS inheritance mechanism in the slave-side NI.

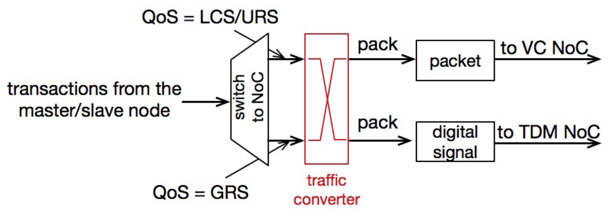

Fig. 7. Location of the traffic conversion unit.

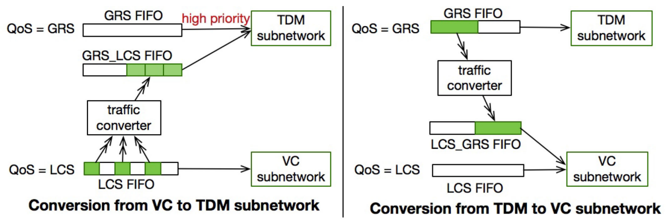

Fig. 8. Traffic conversion unit.

the TDM packet includes fields of destination (dest), source, read/write (r/w) fields, and the valid length of the TDM packet. The destination field is utilized to record the routing target node in the TDM subnetwork and the source field is utilized to record the source node where the packet comes from. The read/write field is utilized by the *switch from* NoC unit in the NI to distribute the AXI4 payload to the corresponding AXI4 payload buffer list. As for the valid length field, it records the length of packet in frames, as a TDM packet is divided into one or multiple frames to transmit. In the LCS/URS traffic, in addition to the four fields given above, the routing information of VC packet also includes the virtual network (vnet) fields. The virtual network field indicates the virtual network the VC packet will get through in the VC subnetwork, which is determined by the packet type (request or response). Besides, the ordering identifiers (IDs) in both the TDM packet and the VC packet include the relevant IDs utilized for the AXI4 transactions' ordering target, which are determined by the ordering unit in the NI and from the AXI4 signal.

As shown in Fig. 5, the architecture of the slave-side NI is almost the same as the master-side NI architecture except that the slave-side NI utilizes the AXI4 master interface to communicate with the slave node. Therefore, the write request buffer and the read request buffer are connected to the *switch from* NoC unit, and the write response buffer and the read response buffer are connected to the *switch to* NoC unit.

Since the AXI4 payload of write response or read response does not include the QoS identifier, to support QoS in the round-trip communication, we propose a QoS *inheritance* mechanism in the slave-side NI. By this mechanism, the response packet inherits the QoS identifier in its corresponding AXI4 request payload. The *QoS inheritance* unit is located between the *AXI4 payload buffer list* and the *slave ordering unit*. As shown in Fig. 6, taking the read process as an example, the read request payloads from the NoC are sent to the slave in sequence. Their QoS identifiers (QoS IDs) will be put into the *read QoS inheritance queue* unit, which is a FIFO structure. According to the AXI4 protocol restrictions, the read responses will be sent out to the AXI4 slave interface in the same sequence as the arrivals of their corresponding read requests in the AXI4 slave interface. So a FIFO queue can assure the correctness of the QoS identifier *inheritance* mechanism. When the read response payload goes to the *switch to* NoC unit, it will inherit the QoS identifier from the *read QoS inheritance queue* unit.

C. Traffic Converter

As shown in Fig. 7, the traffic converter locates after the *switch to* NoC unit and it will convert traffic between the VC and TDM subnetworks once their loads are imbalanced and necessary conditions are met before the AXI4 signals are packed into packets (VC or TDM packets).

In detail, if the average latency of LCS packets in the VC subnetwork is high and at the same time, the utilization of TDM subnetwork is low, some of the LCS packets can be switched to and transferred through the TDM subnetwork instead of the VC subnetwork, as shown in the left part of Fig. 8. The switched packet will be held in the *GRS_LCS FIFO* and the *GRS FIFO* has a higher priority to ensure the bandwidth requirement of the original GRS packets. Since the utilization of the TDM subnetwork is low, the switched packet can have a lower average latency in the TDM subnetwork compared with the latency in the VC subnetwork. In this way, the average latency and throughput of all the LCS packets can be improved without having negative influence on the GRS packets.

As shown in the right part of Fig. 8, if the average latency of LCS packets in the VC subnetwork is low and at the same time, the utilization of the TDM subnetwork is high, some of the GRS packets can be switched to and transferred through the VC subnetwork instead of the TDM subnetwork. The switched packets are usually picked up from the back of the *GRS FIFO* and must satisfy the restriction that their estimated queuing delay in the TDM NI plus the transfer latency in the TDM subnetwork must be larger than the maximal latency of the current LCS packets. A controller is applied in the converter for estimation and calculation, which will only choose the packets that satisfy the restriction from the *GRS FIFO*. Under the restriction, the converted packet will arrive at the slave through the VC subnetwork earlier than utilizing the original TDM subnetwork and thus, the guaranteed throughput of the TDM packet can satisfy the requirement. The converted packet will be held in the *LCS_GRS FIFO*, and the arbitration between the *LCS_GRS FIFO* and the *LCS FIFO* can be determined by their arrival times. In this way, the average latency and throughput of GRS packets will be improved without having much influence on the latency of the original LCS and URS packets in the VC subnetwork.

Compared with the NI designed in [46], we have enhanced the NI architecture with the switch units, QoS inheritance mechanism, and traffic converter.

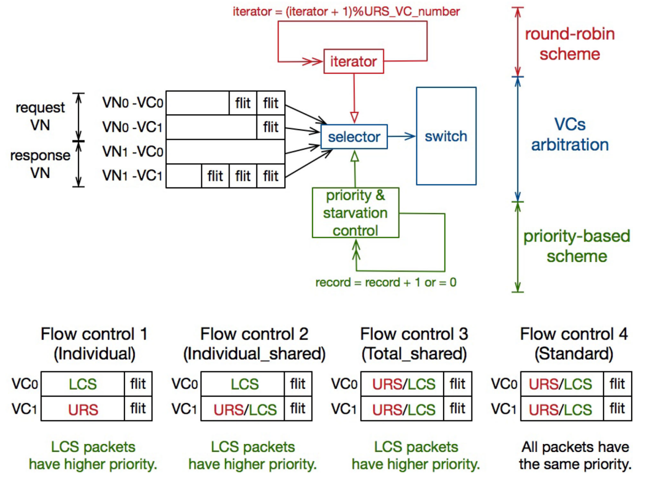
iterator = (iterator + 1)%URS_VC_number
iterator
round-robin scheme
request VN
response VN
VN0 -VC0
VN0 -VC1
VN1 -VC0
VN1 -VC1
flit
selector
switch
VCs arbitration
priority & starvation control
priority-based scheme
record = record + 1 or = 0
Flow control 1 (Individual)
Flow control 2 (Individual_shared)
Flow control 3 (Total_shared)
Flow control 4 (Standard)
VC0
VC1
LCS
URS
URS/LCS
LCS packets have higher priority.
All packets have the same priority.

Fig. 9. Architecture of VC router with four different flow control schemes.

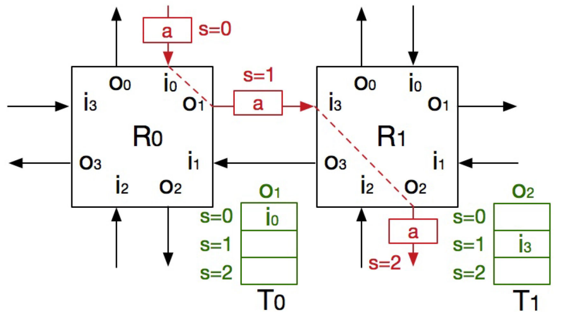

Fig. 10. Architecture of TDM router.

D. Supporting QoS in the NoC-Based Architecture

According to the various communication requirements of different processors, such as CPU and GPU, and I/O devices, such as SATA and USB, we define three QoS schemes and offer the support in the NoC architecture. For the related works. such as [11] and [12], they maximally support two QoS schemes. While, in our system design, orienting multiple QoS requirements, we propose a NoC architecture to support three different QoS schemes at the same time. The design of NoC architecture should also consider both performance and hardware utilization.

Goossens and Hansson [20] discussed the scenario where the architectures for two QoS schemes are integrated into one network. It will indeed increase the link utilization. While, the buffer for the best effort service will influence the frequency and hardware area consumption of TDM NoC and the virtual circuit switch in the TDM NoC will in turn block packets in the buffer, which results in the throughput decrease of both guaranteed and best-effort services due to frequency decrease and the latency increase of best-effort service due to the traffic interference. In this context, the integration of all three different QoS schemes into one network architecture is less likely a reasonable solution.

Therefore, we adopt an eclectic architecture of two subnetworks, a VC-based wormhole subnetwork and a TDM-based virtual-circuit subnetwork, to separately support the LCS, URS in the VC subnetwork, and GRS in the TDM subnetwork.

In Fig. 9, as an example, we show a VC subnetwork that utilizes two virtual networks (VN) to separately support request and response packets to avoid protocol-level deadlock. Each VN consists of two VCs. In our design, we propose three different flow control mechanisms to better serve the LCS and URS packets compared with the traditional flow control mechanism, which are described below. The test results will be shown in Section V.

1) *Individual:* In each VN, the LCS and URS packets can only utilize their individual VCs and the LCS VC has higher priority than the URS VC during VC allocation and switch arbitration, which means that the LCS packets have higher priority over the URS packets.
2) *Individual_Shared:* In each VN, all VCs are shared (can be utilized) by the LCS packets, while the URS packets can only utilize one fixed VC. The LCS packets have higher priority over the URS packets.
3) *Total_Shared:* In each VN, all VCs are shared by the LCS and URS packets, but the LCS packets still have higher priority over the URS packets.
4) *Standard:* In each VN, all VCs are shared by the LCS and URS packets and the LCS packets have the same priority as the URS packets during VC allocation and switch arbitration. This is the best-effort service.

In the TDM subnetwork, we utilize the virtual-circuit TDM router for message transfer. As shown in Fig. 10, each router has it own input port (i_n), output port (o_n), and a global sense of synchronized time slot (s). The routing table in each router is based on the time slot, showing the match relationship between corresponding inport and outport under certain slot s.

Algorithm 1 Pseudocode for TDM Routing Table Establishment

```
1:  Function **routingAlgorithm**:
2:   for each node
3:       for each destination
4:           do Function **findPath**;
5:
6:  Function **findPath**:
7:   for each available slot {
8:       if **findAPath** from source to destination{
9:           for each available slot {
10:              if **findAPath** from destination to source
11:                  record the round-trip path and return;
12:          }
13:      }
14:  }
15:  do not record and return;
16:
17: Function **findAPath**:
18:  if slot to be checked is available {
19:      if current node equals to the destination of this path
20:          return true;
21:      else{
22:          if next hop in the x-axis exists
23:              if **findAPath** from next hop in the x-axis to the
    destination of this path
24:                  return true;
25:          if next hop in the y-axis exists
26:              if **findAPath** from next hop in the y-axis to the
    destination of this path
27:                  return true;
28:          return false;
29:      }
30:  }
31:  return false;
```

Taking the frame a as an example, in time slot 0 ($s = 0$), a is in the input port 0 (i_0) of R_0 and according to its routing table (T_0), it will be transferred to o_1. If the link transfer latency is 1 slot, then at the next time slot 1, according to the routing table T_1 in R_1, frame a will be transferred to o_2. The routing table in each TDM router for round-trip communication is preset statically by a X-first and depth-first search algorithm, which is based on the shortest path, offering guaranteed bandwidth for large-volume flows. The routing algorithm for round-trip communication between the source and destination pair is shown in Algorithm 1.

The router in the TDM NoC only has one buffer per port to store the incoming message, which will be transferred to the downstream router in the next time slot. Adding additional buffers will impact the frequency and area of TDM NoC. From the perspective of area, the buffers will consume more than 90% area in VC router. The result will be much worse in the TDM router for its simpler control logic. From the perspective of area, additional buffers require more complex selector and buffer management unit. It will either influence the critical path, decreasing the frequency, or prolong the pipeline stage, increasing the transfer latency in the TDM router [11].

V. Experiments

A. Methodology

There are various simulators for NoCs such as BookSim2 [48] and NoC-based many-core systems such as

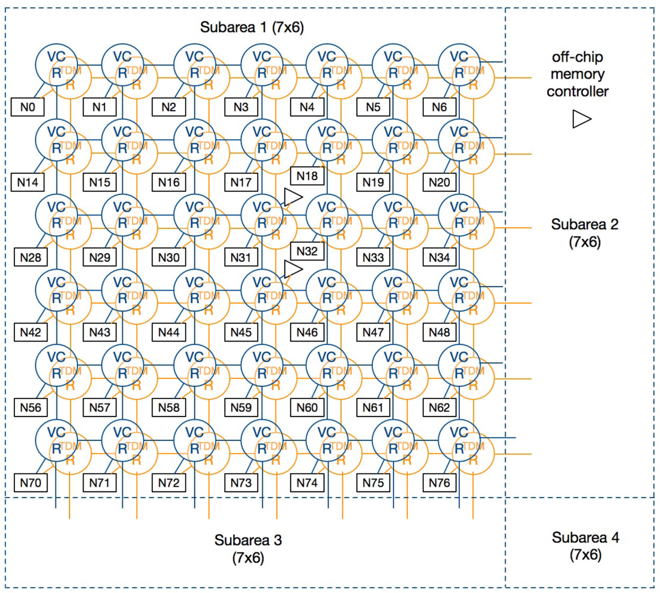

Fig. 11. Architecture of the simulated system.

Gem5 [49]. But none of them supports AXI4, two-subnetwork NoC architecture, and three QoS schemes at the same time. So the best choice was to build up a simulator based on the principles of these known simulators and add additional functions.

Thus, we build up a cycle-accurate simulator in C++, which consists of 168 nodes, 2 subnetworks, and 8 off-chip memory controllers. The architecture of the simulated system is shown in Fig. 11. The system has four identical subareas, each of which is in a 7×6 mesh structure, with a node connected to a router. Each node (N) is composed of a processor (C), a private L1 cache, a shared L2 cache, and both the master-side NI and the slave-side NI so that each node can act as the master node and the slave node at the same time. Besides, each subarea has two off-chip memory controllers, which are connected to the central routers in the subarea.

The VC subnetwork in our simulator is based on the NoC architecture in Garnet of Gem5 [50]. For ease of implementation, we apply a cycle-triggered model, such as Booksim2, instead of the event-triggered method in Gem5. The TDM subnetwork (router) has a simple architecture, which makes it easy for comparison. Then, a customized NI is designed to be compatible with the AXI4 protocol. Therefore, the results produced by our proposed simulator can be directly compared with the results generated by the Gem5 and Booksim2 simulators as long as their NoCs are instantiated with the same configuration.

In realistic scenarios, the process/thread will be occasionally suspended due to the limited hardware resources in the computer system. Besides, the event of the request message injection to the NoC system of each processor is distributed during the process running phase, which will happen once data miss occurs in the L1 cache. To simulate the realistic message injection of each node in a better way, in our simulator, we apply a two-level MMP model in each traffic generator to control its overall injection rate, including LCR, unspecific rate service (URS), and GRS requests. As shown in Fig. 12, the

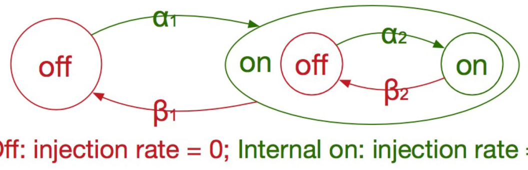

Fig. 12. Two-level MMP model in each traffic generator.

TABLE I
NoC SYSTEM CONFIGURATION IN THE SIMULATOR

Topology	Network size	link	TDM routing / VC routing
mesh	14×12	128 bits	shortest path / XY
Vnet	LCS/URS VC	VC depth	TDM period
2	1/1	4 flits	64 slots (64 cycles)

external level MMP (α_1 and β_1) simulates the state of the process/thread, whose execution interval varies from nanosecond (ns) to millisecond (ms). The internal level MMP (α_2 and β_2) simulates the injection state of request message during the process/thread execution period. The injection rates in the external "off" state, the internal "off" state, and the internal "on" state are separately 0, 0, and r_1. The overall injection rate r of a processor can be calculated by (1). In the simulator, a centralized traffic controller is applied to limit the average injection rate of the whole system by controlling (α and β) values

$$r = \frac{\alpha_1 \times \alpha_2 \times r_1}{(\alpha_1 + \beta_1) \times (\alpha_2 + \beta_2)}. \tag{1}$$

The parameters of the MMP model will be set according to real-world thread behavior, which reflects the processing interval and message injection rate during thread processing. Compared with a real-world system, the difference is that the MMP is a behavior-level function, which is less complex for ease of implementation and results collection. The QoS tag is determined randomly according to predefined rates. So a processor will restrict to one or two types of QoS schemes, which is similar to the processor in a real-world system. The address of each request is also generated randomly in each processor among the processor's communication pairs, which records the possible communication between a master and a slave. Since we did not discuss the influence by the AXI4 ordering requirement, the ID of each request is also randomly decided, and thus, ordering requirements could exist among some requests. But this will not influence the performance in the NoC interconnect, which excludes the ordering units.

From the perspective of the NoC system, its related parameters are shown in Table I. In our simulator, we apply a 14×12 mesh NoC topology. The preset average hop in the NoC of all requests is 4. The TDM period consists of 64 slots (64 simulation cycles, one cycle per slot). The VC subnetwork has two virtual networks (vnet). By default, we utilize the *Individual* flow control mechanism and each of which consists of one virtual channel (VC) for the LCS packets and one VC for the URS packets. The buffer depth of each VC is four flits. The VC router is a two-stage pipeline and each stage as well as the link transfer cost one cycle. As with [51], we assume our simulator runs at 2 GHz.

Currently, the NI implementation can only run up to 600 MHz based on the synthesis result for 40-nm technology [47]. For ease of behavior simulation, in our simulator, we assume that it runs at 500 MHz, and we adapt the 500-MHz NI to a 2-GHz global clock by proportionally prolonging the packet's one-stage pipeline transfer latency in the NI from one NoC cycle to four NoC cycles. Since this adaptation is only for the transfer latency in NI, it will not influence the results of packet transfer latency in each subnetwork. Besides, we also need to double the channel width of the 500 MHz NI from 128 to 256 bits in the simulator as well as would be done in a real-world implementation. Only in this way can the NI support up to 128 Gb/s (= 500 MHz‘$\times$ 256 bits) throughput to satisfy the maximum overall throughput requirement of the 2-GHz subnetworks, which is 117 Gb/s.

Since our proposed simulator utilizes a synthetic traffic generator to conduct a behavior-level execution, the cache coherence is not considered, and thus, no coherence traffic is generated. The cache is simulated by a FIFO buffer with predefined latency, which is the same as the cache latency in a real-world system. The distributed memories controlled by different memory controllers are also simulated in the same way. The bandwidth of memory is set as one flit per cycle. Since all requests have been reordered in the NI, there is no need for the memory to do the reordering job.

B. Experimental Results

1) Throughput: The upper bound on ideal throughput of the VC subnetwork is determined by the critical link. Under uniform random traffic and *X-Y* routing, the maximal per-node throughput ($\Theta_{\text{max,node}}$) of the mesh network can be calculated by (2), where x is the number of routers on the x axis. This formula is derived according to [52, pp. 51–55]. For a critical link, there are maximally half of the nodes sending messages through it. For these nodes, only half of their messages will utilize the critical link in a synthetic traffic pattern where the destinations are generated randomly. The result of $\Theta_{\text{max,node}}$ is ($128 \times 2 \times 2 \div 14$) 36.57 bits/cycle, which is 73.14 Gigabits per second (Gb/s). Hence, the overall VC subnetwork throughput Θ_{max} is up to 12 288 Gb/s for 168 nodes

$$\Theta_{\text{max,node}} \times \frac{x}{2} \times \frac{1}{2} = \text{link\_bandwidth}. \tag{2}$$

The upper bound on ideal throughput of the TDM subnetwork is determined by the overall available preestablished communication paths. By our proposed shortest path routing search algorithm, we totally establish 1841 virtual circuit paths, and the maximal overall throughput of the TDM subnetwork system can be calculated by (3). *TDM_period* stands for the number of slots in a TDM period, which is a unitless number. The maximal throughput of the TDM subnetwork is 3682 bits/cycle (7364 Gb/s)

$$\Theta_{\text{max}} = \frac{\text{paths\_num} \times \text{link\_bandwidth}}{\text{TDM\_period}}. \tag{3}$$

Thus, the overall system throughput can be up to 19652 (12288 + 7364) Gb/s, which is around 117 Gb/s per node.

TABLE II
PARAMETERS IN TRAFFIC GENERATOR

Read	Write	Read-Write Ratio	α_1	β_1	α_2	β_2
128 bits	128 × 5 bits	1:1	7	18	14	86

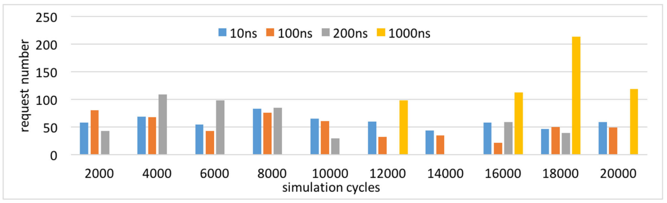

Fig. 13. Packet injection results under test interval of 2000 cycles.

2) Traffic Injection: We show the traffic injection pattern with two-level MMP-based traffic generator. We take four nodes as an example to show different traffic injection scenarios. These four nodes have different process/thread execution interval controlled by the external level MMP of the traffic generator, which are separately 10, 100, 200, and 1000 ns (20, 200, 400, and 2000 simulation cycles). At the beginning of each interval, the process/thread will be decided to execute or hang up during this interval. The execution interval of internal level MMP is 0.5 ns (1 simulation cycle). The packet injection will be decided in each cycle once the process/thread is under "on" state.

The rest of the parameters in the traffic generator is listed in Table II. The read packet is 128 bits (1 flit) and the write packet is 128 × 5 bits (5 flits), which consists of 4-flit data part and 1-flit control part. They will trigger a corresponding response packet, which is five flits for the read response packet and one flit for the write response packet. So a read and write request will both contribute to six-flit throughput. In this case, r_1 in (1) will be six flits/cycle (768 bits/cycle) and each traffic generator (node) will averagely contribute 30 bits/cycle to the NoC throughput, which is 60 Gb/s.

We first show the traffic injection results under a test interval of 2000 cycles in Fig. 13, which means the number of injected requests is collected during each 2000 cycles. In this figure, 2000 on the x axis represents the collection result between cycle 0 and cycle 2000 and 4000 represents the results collected between cycle 2000 and cycle 4000. For the nodes with external interval of 10 and 100 ns, they inject packets in each test interval and the values are relatively uniform compared with the results from the other two nodes. For the node with external interval of 200 ns, the process has been hung up between cycle 10 000 and cycle 14 000 and between cycle 18 000 and cycle 20 000. For the node with external interval of 1000 ns, the distribution of results is much more uneven. It only has packet injection in four test intervals out of all the ten test intervals and the values in these four test intervals are higher than the results from any other nodes.

When increasing the test interval to 10 000 cycles, as shown in Fig. 14, the number of the injected packets by the nodes with external intervals of 10 and 100 ns has a very small fluctuation and the node with external intervals of 200-ns injected packet in each test interval. Under this test interval, the node

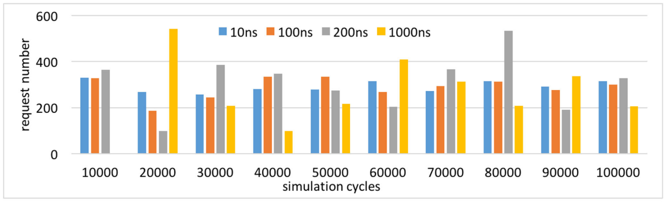

Fig. 14. Packet injection results under test interval of 10 000 cycles.

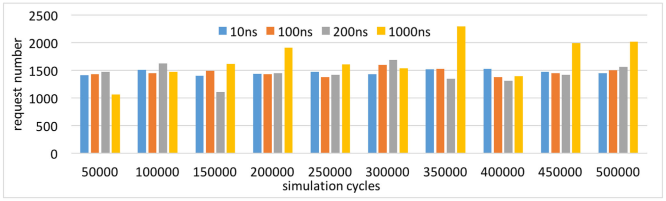

Fig. 15. Packet injection results under test interval of 50 000 cycles.

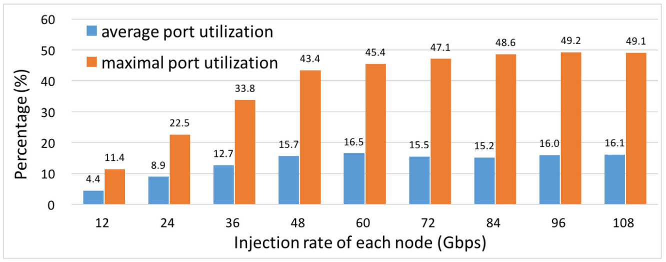

Fig. 16. Average and maximal VC router port utilization.

with external intervals of 1000 ns contributes no packet injection only in the first test interval, although there is still large fluctuation.

We show another results in Fig. 15 with test interval of 50 000 cycles, where the results from each node are almost the same in each test interval. We can predict that for a long-term execution period, each node will have an even packet injection number.

The results in these three figures show that the external level in the two-level MMP traffic generator model can simulate the execution state (executed or suspended) of the process/thread with different execution intervals. We can simulate the characteristic of a process/thread by adjusting the internal level MMP (low injection rate for the computing intensive process/thread and high injection rate for the communication intensive process/thread).

As shown in Fig. 13, our proposed method will have such a characteristic that in some test intervals, the injection rate will be as low as 0, and in some other test intervals, the injection rate can be higher than the average level, especially for a longer internal time, 1000 ns for example. However, by the traditional MMP, there will always be packets injected in each interval. But in reality, some threads will be hung up, resulting in no packet injection, or be executed, resulting in large packet injection. Therefore, compared with the traditional MMP, our proposed two-level MMP can simulate real scenarios in a better and more flexible way.

3) Resource Utilization: We report the average utilization of VC router ports in Fig. 16 with different VC subnetwork traffic injection rate (per node) from 12 Gb/s, which is 0.047 flit/cycle (= 12 Gb/s ÷ 2 GHz ÷ 128 bits/flit), to 108 Gb/s, which is 0.422 flit/cycle (= 108 Gb/s÷2 GHz÷128 bits/flit). The average port utilization of VC routers is calculated by (4), where *utilized_port* stands for the number of utilized ports by network flits, which is obtained from the simulation result and *port_num* stands for the total number of ports/links of all routers in the VC subnetwork, which equals 788 (= $5 \times 12 \times 10 + 4 \times (12 \times 2 + 10 \times 2) + 3 \times 4$) in our architecture. The average port utilization increases from 4.4% to 16.5% and reaches its maximal value at the injection rate of 60 Gb/s, which is 0.235 flit/cycle (= 60 Gb/s÷2 GHz÷128 bits/flit) in each node. After the VC subnetwork is saturated, its utilization fluctuates around 16%. The test results show that the maximal port utilization for a single router can reach up to 49.2% under our experimental setups

$$\text{port\_utilization} = \frac{\text{utilized\_port}}{\text{port\_num} \times \text{simulated\_cycle}}. \quad (4)$$

The ideal average port utilization of VC routers under the saturation throughput and without any traffic contention can be calculated by (5), where $\Theta_{\max}$ stands for the saturated throughput in Gb/s and hop_{ave} stands for the average hop of all packets. The *link_bandwidth* in Gb/s equals to link width multiplied by frequency. We can calculate the result, which equals to 24.36% ($(12288 \cdot 4)/(788 \cdot 2 \cdot 128)$). But in reality, there will always be traffic contention in the NoC under saturated throughput. The utilization results with contention are difficult to be calculated, which also depends on the traffic characteristic. However, we can test it in the full system simulator, such as Gem5, to get the baseline in a real-world scenario. Even for a memory-intensive benchmark, the average utilization among all routers can be very low. In comparison, the average port utilization of 16.5% is an excellent result for the VC subnetwork

$$\text{port\_utilization}_{\text{ideal}} = \frac{\Theta_{\max} \times \text{hop}_{\text{ave}}}{\text{port\_num} \times \text{link\_bandwidth}}. \quad (5)$$

The slot utilization of the TDM NoC can be reflected by that of TDM NIs using (6), which is equal to 17.12% ($1841/(168 \cdot 64)$). Here *NI_num* equals to number of nodes (168), and one TDM path occupies one time slot in NI. The establishment of TDM paths for request–response messages always requires adjacent slots among all routers in the round-trip path, making it difficult to find an available path. The dimension-order routing algorithm can decrease the packet's average hop and transfer latency in NoC. However, compared with adaptive and oblivious routing algorithms, the deterministic routing will make it more difficult to find a path between the source and destination, especially in a large-scale NoC architecture. Therefore, in this case, 17.12% is an acceptable result

$$\text{slot\_utilization} = \frac{\text{paths\_num}}{\text{NI\_num} \times \text{TDM\_period}}. \quad (6)$$

4) Latency: In Fig. 17, we first show the transfer latency of the LCS packets in NoC under different injection rates with four different flow control mechanisms: 1) *Individual*; 2) *Total_shared*; 3) *Individual_shared*; and 4) *Standard*. The

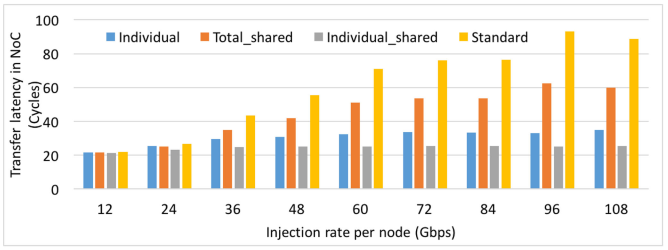

Fig. 17. Transfer latency of the LCS packets in NoC under four different flow control mechanisms.

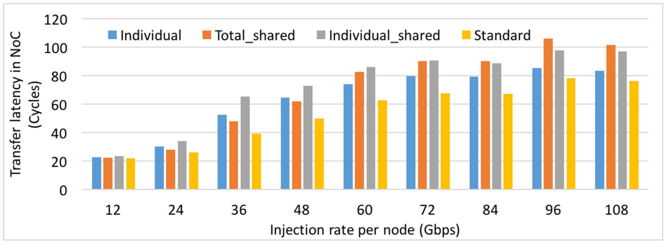

Fig. 18. Transfer latency of the URS packets in NoC under four different flow control mechanisms.

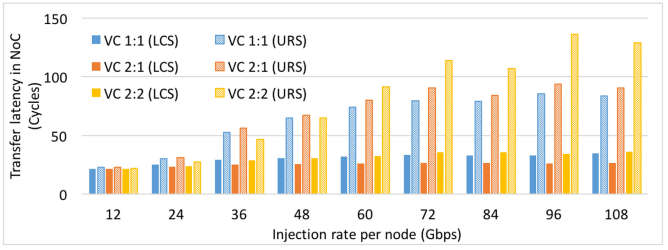

Fig. 19. Transfer latency of the LCS and URS packets in NoC under different VC numbers.

results represent the average transfer latencies of all packets in the NoC. The measurement begins when the packet enters the NoC and ends when the packet leaves the NoC.

It is clearly shown that the *Individual_shared* can always contribute to the best LCS packets performance under different injection rates, especially when the injection rate gets larger. The transfer latency of LCS packets only slightly increases from 21.2 cycles to 25.3 cycles when the injection rate greatly increases from 12 to 108 Gb/s. This is because that the LCS packets will have less contention delay and more resources under this condition. Comparatively, the results by the *Individual* mechanism is a little worse than the results by the *Individual_shared* but is much better than the other two cases. Besides, all the results by the three mechanisms (excluding the *Standard* mechanism) fluctuate a little after saturation (73.14 Gb/s).

As for the results of URS packets, we show them in Fig. 18. Obviously, the optimization mechanisms for the LCS packets will influence the URS packets' performance. Especially in the *Individual_shared* mechanism, the results are the worst before saturation and only better than the *Total_shared* mechanism after saturation. For a system that can tolerate big latency for

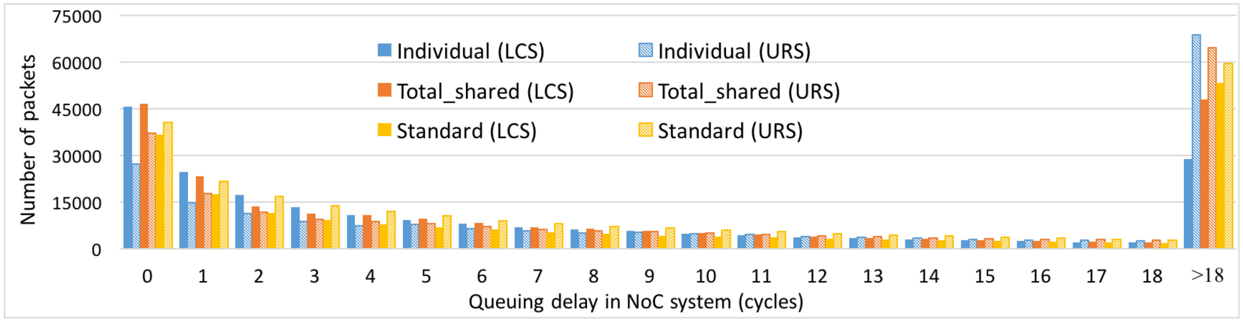

Fig. 20. Distribution of transfer delay for LCS and URS packets.

the URS packets, the *Individual_shared* mechanism is certainly the best choice where the LCS packets can have the lowest transfer latency in NoC.

In Fig. 19, under the *Individual* mechanism, we show the LCS and URS packets' latency results under different VC numbers. For example, *VC 2:2 (LCS)* in the figure means the LCS packets' average latency results with the architecture of two VCs for the LCS packets and two VCs for the URS packets per VN. The results show that the LCS packets' latency can be slightly improved by the 2-VC-for-LCS and 1-VC-for-URS architecture, compared with the 1-VC architecture for both URS and LCS packets. The biggest improvement can reach up to 23.5% when the injection rate is 108 Gb/s. It provides a way to decrease LCS packets' latency at the cost of one additional VC per input port in a router. However, the additional hardware resource cannot always provide performance improvement. For the results by the *VC 2:2* architecture, both the LCS and the URS packets' average latency results are worse than the counterparts in the *VC 2:1* architecture. This is because too many VCs per inport will increase packets queueing delay caused by the switch arbitration.

Fig. 20 shows the transfer delay distribution of LCS and URS packets. The transfer delay represents the time when packets are blocked in the NoC during VC and switch arbitration. The results are under the injection rate of 65 Gb/s (0.254 flit per cycle), where LCS and URS are both 32.5 Gb/s. The LCS packets in the *individual* and *Total_shared* mechanisms show better results, enjoying lower transfer delay. The worst case of *individual* is 546 cycles and of *Total_shared* is 3867 cycles (not shown in this figure), which is much worse than results of the *individual*. So the *individual* mechanism can deliver the best performance.

In Fig. 21, we separately show the average transfer latencies of LCS and URS packets in the NoC and the average queuing delay of the GRS packets in the NI under *individual* mechanism and different injection rates. For example, 10 Gb/s indicates that the injection rate of LCS, URS, and GRS packets is all 10 Gb/s. The average transfer latency of URS packets increases from 22.2 to 91.5 cycles. But comparatively, the average transfer latency of LCS packets only increases slightly from 21.6 to 31.8 cycles. Besides, the queuing delay of the GRS packets in the NI before being divided into frames is also sensitive to the injection rate. Its average value rises from 32.0 cycles to 143.5 cycles with the increment of the GRS packets' injection rate from 10 to 50 Gb/s.

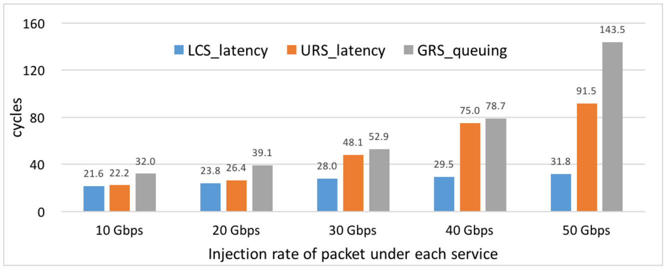

Fig. 21. Transfer latencies of LCS and URS packets and the queuing delay in the NI of the GRS packets.

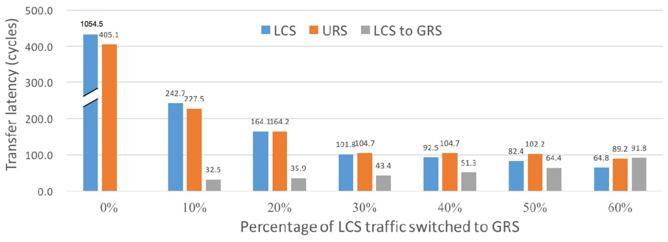

Fig. 22. Scenario 1: Converting LCS packets to GRS packets.

5) Performance of Traffic Conversion: We show the performance improvement by the traffic conversion mechanism in terms of transfer latency, which includes the transfer time of packets in the NI and router. In the first scenario, the injection rate of each node is 70 Gb/s (0.273 flit/frame per cycle) and the URS, LCS, and GRS signals separately take 32.5%, 32.5%, and 35% of traffic. The injection rate of a test node located at (0, 2) in the 14×12 mesh topology is 100 Gb/s (0.391 flit/frame per cycle) and the URS, LCS, and GRS signals separately take 20%, 70%, and 10%. Under this condition, packets to the VC subnetwork suffer heavy contention while packets to the TDM subnetwork do not. We can decrease the transferring latency of LCS signals and increase the utilization of the TDM subnetwork by switching part of LCS traffic in the VC subnetwork to the GRS traffic in the TDM subnetwork. In Fig. 22, we show the packet transfer latency from the test node in the scenario where a portion of LCS packets is switched to the TDM subnetwork. The average transfer latency of LCS packets in the VC subnetwork decreases from 1054.5 to 64.8 cycles when the percentage of switched LCS packets varies from 0% to 60%. The URS packets can also enjoy the benefit, resulting in average latency decreased from

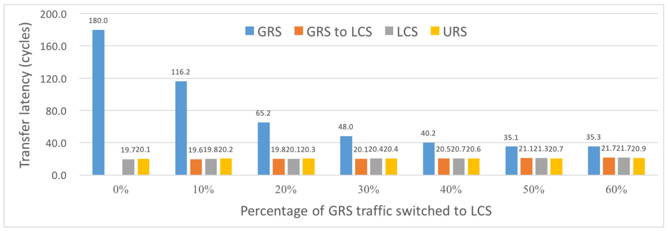

Fig. 23. Scenario 2: Converting GRS packets to LCS packets.

405.1 to 89.2 cycles. For the LCS packets in the TDM subnetwork, the average transfer latency, under each percentage of traffic conversion, is better than the results without traffic conversion mechanism. Since the GRS packets have higher priority than the converted LCS packets in the TDM subnetwork, the throughput of GRS packets in the TDM subnetwork will not be influenced.

As we have explained before, since the GRS packets have higher priority than the converted LCS packets in the TDM subnetwork, the throughput of GRS packets in the TDM subnetwork will not be influenced by the converted LCS packets. So the total throughput of GRS packets will be larger than the original results, which will satisfy the requirements. Therefore, we did not discuss the throughput results here, and instead, the latency for GRS traffic is discussed as a comparison with the transfer latency of LCS and URS packets.

In the second scenario, the injection rate of each node is 20 Gb/s (0.078 flit/frame per cycle) and the URS, LCS, and GRS signals separately take 32.5%, 32.5%, and 35% of traffic. The injection rate of the test node is 120 Gb/s (0.469 flit/frame per cycle) and the URS, LCS, and GRS signals separately take 20%, 10%, and 70% of traffic. Under this condition, packets to the TDM subnetwork suffer heavy contention while packets to the VC subnetwork do not. In Fig. 23, we show the packet transfer latency from the test node in the scenario where a portion of GRS packets is switched to the VC subnetwork using the LCS-oriented VC. The average transfer latency of GRS packets in the TDM subnetwork greatly decreases from 180.0 to 35.3 cycles when the percentage of the switched GRS packets varies from 0% to 60%. For the GRS packets in the VC subnetwork, the average transfer latency is around 20 cycles under each percentage of traffic conversion. The performance of GRS packets increases a lot by the traffic conversion without obvious influence on the average transferring latency of the original LCS and URS packets.

We can compare our proposed conversion method with the TDM highway scheme in [11]. Both methods focus on improving the resource usage. The best improvement result in [11] is 52% for the Ericsson radio system (ERS) benchmark, while our approach can increase 93.85% maximally. However, we note that our tests utilized synthetic traffic while [11] used application-oriented benchmarks. Furthermore, [11] tries to improve the TDM NoC performance only, but our proposed method can improve the performance of both TDM and VC networks at the same time, and improve the load balance.

VI. Conclusion

In this article, we build up a NoC-based communication system to support both the AMBA AXI4 protocol and three different QoS schemes. In the NI design, it supports the message format conversion between the AXI4 signal in the master/slave node and the packet in the NoC. The NI is set in both the slave side and master side to make the NoC design independent from the AXI4 protocol. It also provides an *inheritance* mechanism to support QoS in the round-trip NoC transfer. After defining three different QoS schemes, we design a NoC architecture with two subnetworks to support them efficiently. In the experimental part, we compared the average transfer latencies of LCS and URS packets and the average queuing delay of the GRS packets under different injection rates, and our proposed *Individual_shared* architecture can achieve the best results for the LCS packets compared with the *Individual*, *Total_shared*, and *Standard* flow control mechanisms. Besides, we show traffic generation results by the two-level MMP-based traffic generator. Finally, the decrease of packet latency by the traffic conversion mechanism indicates that our proposed traffic converter can contribute to improved NoC performance.

There remains a valuable problem to be discussed in future research: how will the conversion percentage be determined at runtime to minimize the latency of LCS packets on both VC and TDM subnetworks based on runtime utilization measurements? This problem can be addressed by presenting an initial value and updating it by a learning method to be adaptive to the runtime characteristics of the system.

References

[1] ARM. (2011). *AXI and ACE Protocol Specification*. [Online]. Available: https://https://static.docs.arm.com/ihi0022/g/IHI0022G_amba_axi_protocol_spec.pdf

[2] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th ACM Design Autom. Conf.*, Las Vegas, NV, USA, 2001, pp. 684–689.

[3] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[4] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "SPIN: A scalable, packet switched, on-chip micro-network," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, Munich, Germany, 2003, pp. 70–73.

[5] W.-C. Kwon, S. Yoo, J. Um, and S.-W. Jeong, "In-network reorder buffer to improve overall NoC performance while resolving the in-order requirement problem," in *Proc. Conf. Design Autom. Test Eur.*, Nice, France, 2009, pp. 1058–1063.

[6] X. Yang, Z. Qing-Li, F. Fang-Fa, Y. Ming-Yan, and L. Cheng, "NISAR: An AXI compliant on-chip NI architecture offering transaction reordering processing," in *Proc. 7th Int. Conf. ASIC*, Guilin, China, 2007, pp. 890–893.

[7] M. Ramirez, M. Daneshtalab, J. Plosila, and P. Liljeberg, "NoC-AXI interface for FPGA-based MPSoC platforms," in *Proc. 22nd Int. Conf. Field Program. Logic Appl. (FPL)*, Oslo, Norway, 2012, pp. 479–480.

[8] A. Radulescu *et al.*, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 1, pp. 4–17, Jan. 2005.

[9] X. Chen, Z. Lu, S. Liu, and S. Chen, "Round-trip DRAM access fairness in 3D NoC-based many-core systems," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, p. 162, 2017.

[10] A. Sharifi, E. Kultursay, M. Kandemir, and C. R. Das, "Addressing end-to-end memory access latency in NoC-based multicores," in *Proc. 45th Int. Symp. Microarchit.*, Vancouver, BC, Canada, 2012, pp. 294–304.

[11] S. Liu, Z. Lu, and A. Jantsch, "Highway in TDM NoCs," in *Proc. 9th Int. Symp. Netw. Chip (NOCS)*, 2015, p. 1–8.

[12] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal network on chip: Concepts, architectures, and implementations," *IEEE Design Test Comput.*, vol. 22, no. 5, pp. 414–421, Sep./Oct. 2005.

[13] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "A high-performance network interface architecture for NoCs using reorder buffer sharing," in *Proc. 18th Euromicro Conf. Parallel Distrib. Netw. Based Process.*, Pisa, Italy, 2010, pp. 546–550.

[14] M. H. Neishaburi and Z. Zilic, "Debug aware AXI-based network interface," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, Vancouver, BC, Canada, 2011, pp. 399–407.

[15] N. Tidala, "High performance network on chip using AXI4 protocol interface on an FPGA," in *Proc. 2nd Int. Conf. Electron. Commun. Aerosp. Technol. (ICECA)*, Coimbatore, India, 2018, pp. 1647–1651.

[16] T. Nguyen, D.-H. Bui, H.-P. Phan, T.-T. Dang, and X.-T. Tran, "High-performance adaption of ARM processors into network-on-chip architectures," in *Proc. IEEE Int. SOC Conf.*, Erlangen, Germany, 2013, pp. 222–227.

[17] X. Liao, J. Zhou, and X. Liu, "Exploring AMBA AXI on-chip interconnection for TSV-based 3D SoCs," in *Proc. IEEE Int. 3D Syst. Integr. Conf. (3DIC)*, Osaka, Japan, 2011, pp. 1–4.

[18] T. Kazaz, C. Van Praet, M. Kulin, P. Willemen, and I. Moerman, "Hardware accelerated SDR platform for adaptive air interfaces," 2017. [Online]. Available: arXiv:1705.00115.

[19] C. Liß, M. Ulbricht, U. F. Zia, and H. Müller, "Architecture of a synchronized low-latency network node targeted to research and education," in *Proc. 18th Int. Conf. High Perform. Switch. Routing (HPSR)*, Campinas, Brazil, 2017, pp. 1–7.

[20] K. Goossens and A. Hansson, "The aethereal network on chip after ten years: Goals, evolution, lessons, and future," in *Proc. 47th Design Autom. Conf.*, 2010, pp. 306–311.

[21] S. Ma, Z. Wang, N. E. Jerger, L. Shen, and N. Xiao, "Novel flow control for fully adaptive routing in cache-coherent NoCs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2397–2407, Sep. 2014.

[22] I. Pérez, E. Vallejo, and R. Beivide, "Efficient router bypass via hybrid flow control," in *Proc. 11th Int. Workshop Netw. Chip Archit. (NoCArc)*, Fukuoka, Japan, 2018, pp. 1–6.

[23] S. Evain and J.-P. Diguet, "Efficient space-time NoC path allocation based on mutual exclusion and pre-reservation," in *Proc. 17th Great Lakes Symp. VLSI (GLSVLSI)*, 2007, pp. 457–460.

[24] Z. Lu, D. Brachos, and A. Jantsch, "A flow regulator for on-chip communication," in *Proc. 22nd Int. SOC Conf.*, Belfast, U.K., 2009, pp. 151–154.

[25] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Buffer optimization in network-on-chip through flow regulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1973–1986, Dec. 2010.

[26] Z. Lu and Y. Yao, "Dynamic traffic regulation in NoC-based systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 556–569, Feb. 2017.

[27] B. Wang, Z. Lu, and S. Chen, "ANN based admission control for on-chip networks," in *Proc. 56th Design Autom. Conf. (DAC)*, Las Vegas, NV, USA, 2019, pp. 1–6.

[28] Z. Lu and A. Jantsch, "TDM virtual-circuit configuration for network-on-chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 8, pp. 1021–1034, Aug. 2008.

[29] S. B. Patil, T. Liu, and R. Tessier, "A bandwidth-optimized routing algorithm for hybrid FPGA networks-on-chip," in *Proc. 26th Annu. Int. Symp. Field Program. Custom Comput. Mach. (FCCM)*, Boulder, CO, USA, 2018, pp. 25–28.

[30] N. Kapre *et al.*, "Packet switched vs. time multiplexed FPGA overlay networks," in *Proc. 14th Annu. IEEE Symp. Field Program. Custom Comput. Mach.*, Napa, CA, USA, 2006, pp. 205–216.

[31] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. 3rd IEEE Int. ACM Symp. Field-Program. Gate Arrays*, Napa Valley, CA, USA, 1995, pp. 111–117.

[32] A. Shpiner, E. Kantor, P. Li, I. Cidon, and I. Keslassy, "On the capacity of bufferless networks-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 492–506, Feb. 2015.

[33] R. A. Stefan, A. Molnos, and K. Goossens, "dAElite: A TDM NoC supporting QoS, multicast, and fast connection set-up," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 583–594, Mar. 2014.

[34] M. Winter and G. P. Fettweis, "Guaranteed service virtual channel allocation in NoCs for run-time task scheduling," in *Proc. Design Autom. Test Eur. (DATE)*, Grenoble, France, 2011, pp. 1–6.

[35] S. Liu, A. Jantsch, and Z. Lu, "Parallel probe based dynamic connection setup in TDM NOCs," in *Proc. Design Autom. Test Eur. (DATE)*, Dresden, Germany, 2014, pp. 1–6.

[36] R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Design of a high-throughput distributed shared-buffer NoC router," in *Proc. 4th Int. Symp. Netw. Chip*, Grenoble, France, 2010, pp. 69–78.

[37] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," *Proc. SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 251–264, 2002.

[38] A. Prakash, A. Aziz, and V. Ramachandran, "Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies," in *Proc. INFOCOM*, vol. 3. Hong Kong, 2004, pp. 2026–2037.

[39] Y. Lu, J. McCanny, and S. Sezer, "Generic low-latency NoC router architecture for FPGA computing systems," in *Proc. 21st Int. Conf. Field Program. Logic Appl.*, Chania, Greece, 2011, pp. 82–89.

[40] L. Xin and C.-S. Choy, "A low-latency NoC router with lookahead bypass," in *Proc. Int. Symp. Circuits Syst.*, Paris, France, 2010, pp. 3981–3984.

[41] L. Wang, S. Ma, C. Li, W. Chen, and Z. Wang, "A high performance reliable NoC router," *Integration*, vol. 58, pp. 583–592, Jun. 2017.

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

[43] Xilinx. (2015). *Using Quality of Service (QoS) Capabilities in Zynq-7000 AP SoC Devices*. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1266-zynq-7000-qos.pdf

[44] Synopsys. *Improving Quality of Service with DesignWare IP for AMBA Interconnect*. Accessed: 2019. [Online]. Available: https://www.synopsys.com/designware-ip/technical-bulletin/improving-quality-of-service.html

[45] Altera. (2014). *Qsys Interconnect*. [Online]. Available: https://www.intel.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/hb/qts/qsys_interconnect.pdf

[46] B. Wang and Z. Lu, "Supporting QoS in AXI4 based communication architecture," in *Proc. Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Limassol, Cyprus, 2020, pp. 548–553.

[47] B. Wang and Z. Lu, "Efficient support of AXI4 transaction ordering requirements in many-core architecture," *IEEE Access*, vol. 8, pp. 182663–182678, 2020.

[48] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. J. Dally, *Booksim 2.0 User's Guide*, Standford Univ., Standford, CA, USA, 2010.

[49] N. Binkert *et al.*, "The Gem5 simulator," *ACM Trans. SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[50] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Boston, MA, USA, 2009, pp. 33–42.

[51] Y. Yao and Z. Lu, "iNPG: Accelerating critical section access with in-network packet generation for NoC based many-cores," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Vienna, Austria, 2018, pp. 15–26.

[52] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Amsterdam, The Netherlands: Elsevier, 2004.

Boqian Wang received the B.S. and M.S. degrees in computer science from the National University of Defense Technology, Changsha, China, in 2013 and 2016, respectively, and the Ph.D. degree in electronic engineering and computer science from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2020.

He is currently an Assistant Researcher with the Beijing Institute of Biotechnology, Beijing, China. His research areas include interconnection network and computer architecture.

Zhonghai Lu (Senior Member, IEEE) received the B.S. degree in radio and electronics from the Beijing Normal University, Beijing, China, in 1989, and the M.S. degree in system-on-chip design and the Ph.D. degree in electronic and computer system design from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2002 and 2007, respectively.

He is a Professor with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Kista, Sweden. He has authored about 200 peer-reviewed papers. His research interests include interconnection network, computer architecture, and embedded system.