IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 14, 2018, accepted July 22, 2018, date of publication August 2, 2018, date of current version August 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2862633

Cache Access Fairness in 3D Mesh-Based NUCA

ZICONG WANG[1], XIAOWEN CHEN[1,2], ZHONGHAI LU[2], (Senior Member, IEEE), AND YANG GUO[1]

[1]College of Computer, National University of Defense Technology, Hunan 410073, China
[2]School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 16440 Stockholm, Sweden

Corresponding author: Xiaowen Chen (xwchen@nudt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61502508 and Grant 61572025.

ABSTRACT Given the increase in cache capacity over the past few decades, cache access efficiency has come to play a critical role in determining system performance. To ensure efficient utilization of the cache resources, *non-uniform cache architecture* (NUCA) has been proposed to allow for a large capacity and a short access latency. With the support of networks-on-chip (NoC), NUCA is often employed to organize the last level cache. However, this method also hurts cache access fairness, which denotes the degree of non-uniformity for cache access latencies. This drop in fairness can result in an increased number of cache accesses with overhigh latency, which leads to a bottleneck in system performance. This paper investigates the cache access fairness in the context of NoC-based 3-D chip architecture, and provides new insights into 3-D architecture design. We propose *fair-NUCA* (F-NUCA), a co-design scheme intended to optimize cache access fairness. In F-NUCA, we strive to improve fairness by equalizing cache access latencies. To achieve this goal, the memory mapping and the channel width are both redistributed non-uniformly, thereby equalizing the non-contention and contention latencies, respectively. The experimental results reveal that F-NUCA can effectively improve cache access fairness. When F-NUCA is compared with the traditional static NUCA in a simulation with PARSEC benchmarks, the average reductions in average latency and latency standard deviation are 4.64%/9.38% for a $4 \times 4 \times 2$ mesh network, as well as 6.31%/13.51% for a $4 \times 4 \times 4$ mesh network. In addition, a 4.0%/6.4% improvement in system throughput can be achieved for the two scales of mesh networks, respectively.

INDEX TERMS 3D chip architecture, cache memory, memory architecture, memory mapping, multiprocessor interconnection networks, networks-on-chip, non-uniform cache architecture.

I. INTRODUCTION

As the number of cores integrated on chips continues to increase, Networks-on-Chip (NoC) is becoming the fundamental infrastructure for use in chip multi-processors (CMPs) due to its good scalability and flexibility in interconnection [1]. However, given the concurrent increases in network scale, the conventional two-dimensional (2D) chip architecture has begun to suffer from performance degradation. This degradation occurs due to the increase in network diameter, which in turn has a significant impact on deciding system performance [2]. As a result, three-dimensional (3D) chip architecture is envisioned to be a viable solution for the issues arising from further increases in transistor density and system performance [3], [4]. Through the use of 3D die-stacking technology, it is feasible to partition a single large die into several smaller segments and then stack these in a 3D fashion [5]. In 3D die-stacking, multiple layers are stacked together using vertical links such as Through Silicon Vias (TSVs), which are currently the most popular and practical way to implement vertical interconnection [6], [7]. In short, the advent of 3D die-stacking technology has provided a new horizon for NoC design, and the traditional 2D NoC fabric can be extended to the third dimension by means of 3D integration.

Cache capacity has also continued to increase over the past few decades. To efficiently utilize the capacity of the shared *last level cache* (LLC) and allow for short access latency, *non-uniform cache architecture* (NUCA) has been proposed in order to allow the cache access time to vary depending on the distance traversed along the chip [8]. As NUCA can provide non-uniform cache access latencies, it can thus distribute the large-capacity shared LLC across different cores to facilitate scalability and efficiency. However, two distinct trends begin to emerge as network size is scaled up. Firstly, network latency gradually comes to dominate the cache access latency; secondly, the communication distance and latency gaps between different nodes continue to grow, and larger

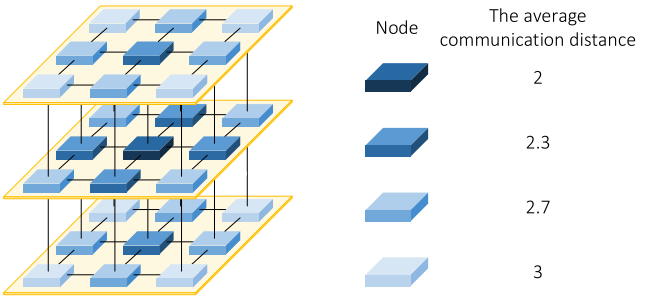

FIGURE 1. The average communication distance of each node for a 3 × 3 × 3 mesh network. The closer a node is to the central region of the network, the smaller its average communication distance.

gaps of this kind can exacerbate the degree of non-uniformity for network latencies.

Consider a $3 \times 3 \times 3$ mesh network as an example. Figure 1 depicts the average communication distance of each node. We can observe that the closer each node is to the center of the network, the smaller its average communication distance. As can be seen in Figure 1, the average communication distance for the central node is two hops, but the eight nodes in the corner have an average communication distance of three hops. Hence, the central nodes have an advantage over the peripheral nodes in terms of communication, and this advantage grows more significant with increasing network size. The consequences of increasingly unbalanced network latencies include larger latency gaps between packets, and there will be more high-latency packets overall.

The issue of unbalanced network latencies can be a direct cause of high variances across cache access latencies, which in turn leads to more cache accesses suffering from overhigh latencies. A cache access experiencing overhigh latency can block the core's running progress, become a system bottleneck, and degrade overall system performance. From the perspective of improving system performance, it is crucial to provide more uniform cache access latencies, notwithstanding that NUCA was initially designed to support cache access latencies that are non-uniform. In this paper, we use *fairness* to denote the degree of latency equalization for cache accesses.

To increase cache access fairness, we must endeavor to equalize cache access latencies. It is first necessary to analyze the composition of cache access latencies. In general, cache access latency has three components: non-contention latency, contention latency, and cache bank access latency.

$$T_{access} = H \cdot \tau_{1hop} + T_c + T_{bank} \tag{1}$$

In (1), H denotes the hop count of a packet traveling from the source node to the destination node. τ_{1hop} indicates the delay through one hop without contention on the network. H and τ_{1hop} commonly determine the non-contention latency. T_c denotes the contention latency, which reflects the time spent waiting for resources. T_{bank} refers to the latency of accessing the cache bank. It should be noted that τ_{1hop} and T_{bank} are constant under fixed network parameters and router architecture. Therefore, our approach strives to narrow the gap of each bank's access latencies by regulating $H \cdot \tau_{1hop}$ and T_c, which affect the non-contention and the contention latency respectively.

In this paper, we point out that the cache access fairness problem can result in more high-latency cache accesses, leading to a system bottleneck that can seriously degrade system performance. Consequently, we propose *fair-NUCA* (F-NUCA), a co-design scheme for optimizing cache access fairness in the context of 3D mesh-based NUCA. The contributions of this paper can be outlined as follows:

- We propose a location-aware memory-to-LLC mapping scheme for 3D mesh-based NUCA, which can better adapt to 3D NUCA architecture in terms of cache access fairness.
- Based on the proposed mapping scheme, we then propose a non-uniform link distribution design to further improve cache access fairness by affecting contention latencies.

II. BACKGROUND AND RELATED WORK

3D chip architecture is considered to be a promising option for traditional chip architecture. Owing to the numerous benefits resulting from the higher density of 3D-stacked integration, this architecture has attracted significant attention in the course of many prior research projects. For example, Xie *et al.* [9] explore the design space of 3D architecture. Black *et al.* [10] discuss the architecture advantages and challenges of 3D die-stacking. Loh *et al.* [11] present different approaches to the implementation of 3D architecture, and discuss the impact of 3D integration technology on processor design. The emergence of 3D chip architecture has also brought about opportunities and challenges for the development of NoC, and many researchers have explored the design space of 3D NoC by means of architectural and algorithmic techniques [5], [12]–[18]. Feero and Pande [12] evaluate the performance of multiple 3D NoC architectures and demonstrate their superior functionality and overhead relative to traditional 2D implementations. Pavlidis and Friedman [13] present various possible topologies for 3D NoC, describing analytic models for zero-load latency and power consumption. Yin *et al.* [14] propose a honeycomb topology as an alternative to 3D NoC design, while Kim *et al.* [5] present a detailed exploration of inter-strata communication architectures in 3D NoC. Moreover, new routing algorithms have also been investigated in the context of 3D NoC architecture [15]–[18].

The fundamental concept behind of NUCA technology is that a large uniform monolithic cache can be split into multiple small cache banks that are associated with each core, thus facilitating faster access to cache data that resides closer to the processor. Kim *et al.* [8] initially introduced the concept of NUCA in high-performance cache designs, proposing two NUCA schemes that differ based on whether or not a given block in memory is mapped to a unique cache bank.

The static NUCA (S-NUCA) maps a block to a unique bank based on the block's index in a physical address; therefore, every cache bank corresponds to a predetermined section in physical address space. In dynamic NUCA (D-NUCA), a cache block is associated with a set of cache banks (bank-set), and the frequently accessed blocks can migrate dynamically towards the requesting cores, which works to reduce cache access latencies [19]. Due to the flexibility attributed to block migration, D-NUCA requires a sophisticated search mechanism to locate cache blocks, meaning that the hardware costs are considerably increased compared with S-NUCA; in addition, D-NUCA does not always perform better than S-NUCA [8], [20]. As a result, S-NUCA is more widely used compared with D-NUCA in modern CMPs architecture, and our design is based on and aims to optimize S-NUCA.

Since the introduction of NUCA, more and more modern processors have gradually adopted NUCA technology when designing the cache system. It is foreseeable that most future architectures with large caches will be organized according to NUCA principles [8], [20]. It is equally predictable that a similar development trend will be observed in 3D architecture. Li *et al.* [21] study the challenges for LLC design and management in 3D NUCA-based CMPs. Madan *et al.* [22] postulate 3D NUCA-based CMPs, which are employed using OS-based page coloring to reduce horizontal communication cost, and design a heterogeneous reconfigurable cache hierarchy. Jung *et al.* [23] explore a design for 3D NUCA with NoC interconnection and investigate the problem of partitioning shared LLC in order to concurrently execute multiple applications. Zhang *et al.* [24] survey the design of cache and memory architectures for 3D CMPs. However, previous studies have excessively concentrated on minimizing the cache access latency, which may negatively impact cache access fairness, particularly in the context of 3D architecture. In contrast, this paper not only concerns about the cache access latency, but also pays more attention to the problems inherent in NUCA, and focuses on offering more uniform cache access latency.

The contentions and interferences on shared resources lead to high variances across request latencies, which can hurt the fairness and worsen the system performance. Therefore, some researchers explore how to balance the latencies of requests and improve the fairness [25], [26]. Das *et al.* [25] propose novel router prioritization policies to improve the system fairness by exploiting interfering packets' available slack. Sharifi *et al.* [26] point out that the variances of memory access latencies degrade the overall system performance, so it is crucial to balance latencies of memory accesses and reduce the number of high-latency memory accesses. Some researchers find the unbalanced buffer utilization may cause degradation in performance [27], [28]. Gorgues *et al.* [27] describe a new flow control and routing algorithm in order to achieve balanced buffer utilization. Lu and Yao [28] propose three coherent mechanisms to facilitate and realize unbiased workload-adaptive resource allocation. To the best of our knowledge, there is minimal related work studying cache access fairness in the context of 3D NUCA-based CMPs; accordingly, this paper can provide a new horizon in the design of 3D architectures.

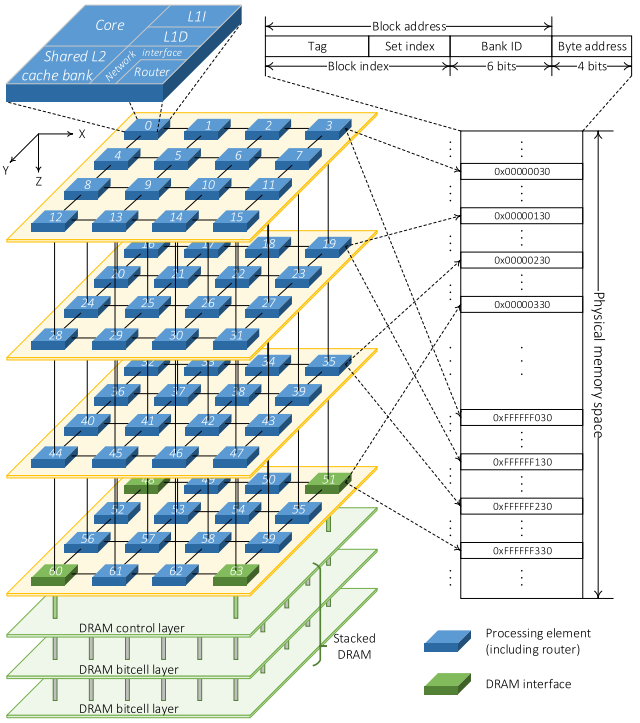

FIGURE 2. The baseline architecture, which includes four processor layers and three DRAM layers. The LLC (i.e., shared L2 cache) is distributed on each node. The physical memory address includes two parts: byte address and block address. The bank address field is taken from the LSBs of the block address (assume that the memory address width is 32 bits, the block size is 16 bytes[1], and the bank address field occupies 6 bits). Hence, the cache system applies block-interleaving addressing to distribute the memory blocks uniformly on each LLC bank. The number label on each node denotes the node (core/bank) ID.

III. PRELIMINARY: BASELINE 3D NUCA

Our baseline architecture is as shown in Figure 2. Such architecture is referred to as a *3D NoC-based many-core system*, which is widely used in 3D many-core system studies and has been illustrated in [29]–[32]. The architecture includes both processor layers and DRAM layers. Processor layers integrate processing elements as computation resources and caches as on-chip storage resources. Our system is made up of 64 ($4 \times 4 \times 4$) nodes, which are distributed in four processor layers and interconnected via a 3D mesh network. Each node includes a core, a private level one (L1) instruction/data cache, a shared level two (L2) cache bank, and a network interface that connects the core and the caches to the router. The routing algorithm employed is XYZ deterministic dimension-order routing (XYZ-DOR). All the shared L2 cache banks collectively constitute the distributed LLC, which is organized according to S-NUCA principles.

[1]In reality, the block size in the experimental configuration is 64 bytes as shown in Table 1. Here we assume it is 16 bytes to enable a clear demonstration of the mapping scheme.

In S-NUCA, each memory block is statically mapped to a unique bank based on the bank ID field in the block address; therefore, the blocks in memory are mapped to each L2 cache bank sequentially. In other words, the distributed L2 cache banks adopt block-interleaving addressing. This block-interleaving used for the cache system can destroy the data locality when mapping from the memory to the LLC, and the accesses to LLC are distributed on each L2 cache bank. In this paper, a balanced workload for general-purpose applications will be considered, because the balanced application mapping mechanism is often employed to avoid network hotspots [33], [34], and the balanced workload can be a good generalization to different kinds of applications. For these balanced workloads, the traffic pattern can to some extent be uniformly distributed on each bank due to the uniform memory-to-LLC mapping scheme. As a result, the calculation in the next section is on the basis of uniform traffic pattern.

IV. FAIR CACHE ACCESS DESIGN

A. LOCATION-AWARE MEMORY MAPPING

In the traditional 3D NUCA, the memory blocks are uniformly distributed on each bank; in other words, the memory mapping scheme is location-agnostic. In order to equalize the access latency of each LLC bank, we hope to change the bank access probability according to the bank's location in network by affecting the memory-to-LLC mapping scheme. This necessitates amending the traditional uniform mapping scheme to make it non-uniform. In the following discussion, the location-aware distribution of memory-to-LLC mapping will be calculated.

1) SCHEME

Assume that the proportion of mapping blocks for the ith bank is p_i. From the perspective of cache access, p_i can also be regarded as the probability of a bank being accessed. The larger the p_i, the more blocks are mapped to the ith bank and the greater the probability of the bank being accessed. To fairly treat the access pattern of each core, consider a large contiguous memory area including M cache blocks in a case where all cores issue one access request for each block in this area (i.e., M access requests per core). Further, suppose that the 3D mesh-based CMPs have N nodes. If the ith bank is accessed by the jth core, then the cache access non-contention latency between the ith bank and jth core can be calculated by (2), as follows:

$$t_{i,j} = h_{i,j}\tau_{1hop} \tag{2}$$

In (2), $t_{i,j}$ denotes the non-contention latency of one cache access request which is toward to the ith bank and issued by the jth core, while $h_{i,j}$ represents the hop count between the ith bank and jth core. Assuming that there are m_i blocks mapped to the ith bank in the aforementioned memory area, each core will thus issue m_i requests for the ith bank. The total cache access non-contention latencies of the ith bank is the sum of the total access latencies caused by each core, and can be calculated by (3):

$$T_i = \sum_{j=0}^{N-1} m_i t_{i,j} = \sum_{j=0}^{N-1} m_i h_{i,j}\tau_{1hop} \tag{3}$$

To normalize T_i, we divide it by the constant $M \cdot \tau_{1hop}$, and define it as the cache access cost of the ith bank, as per (4):

$$c_i = \frac{T_i}{M \cdot \tau_{1hop}} = \sum_{j=0}^{N-1} \frac{m_i}{M} h_{i,j} \tag{4}$$

It should be noted that m_i/M is the proportion of mapping blocks for the ith bank, meaning that we can replace it with p_i. Accordingly, the cache access cost of the ith bank can be represented by (5):

$$c_i = \sum_{j=0}^{N-1} \frac{m_i}{M} h_{i,j} = \sum_{j=0}^{N-1} p_i h_{i,j} \tag{5}$$

We use the vector $\boldsymbol{C}$ to represent the distribution of each bank's cache access cost. We can get the average value of $\boldsymbol{C}$:

$$\mu(\boldsymbol{C}) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_i h_{i,j} \tag{6}$$

To equalize the cache access cost for each node's bank, it is expected that the difference of the elements in $\boldsymbol{C}$ should be as small as possible. Therefore, we set the standard deviation of $\boldsymbol{C}$ as the objective function:

$$\sigma(\boldsymbol{C}) = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (c_i - \mu(\boldsymbol{C}))^2} \tag{7}$$

Note that the sum of the probability of each bank being visited should be 1, i.e., satisfy the following constraint:

$$\sum_{i=0}^{N-1} p_i = 1, \quad p_i \geq 0 \quad (0 \leq i \leq N-1) \tag{8}$$

With the objective function and constraints summarized above, the optimization can be abstracted as the following nonlinear programming problem:

$$\begin{aligned} &\textbf{\textit{minimize}}\ \sigma(\boldsymbol{C}) \\ &\textbf{\textit{subject to}}\ \sum_{i=0}^{N-1} p_i = 1 \\ &\qquad\qquad p_i \geq 0 \quad (0 \leq i \leq N-1) \end{aligned} \tag{9}$$

We next model this nonlinear programming problem in MATLAB and calculate the probability distribution $\boldsymbol{P}$ with the help of MATLAB's optimization toolbox. Without loss of generality, we calculate the distribution $\boldsymbol{P}$ for a $4 \times 4 \times 4$ mesh network ($N = 64$), and resize the vector $\boldsymbol{P}$ to a $4 \times 4 \times 4$ matrix. We use $\boldsymbol{P}_{..i}$ to represent the ith component of $\boldsymbol{P}$ in the third dimension. Due to the symmetrical nature of a $4 \times 4 \times 4$ 3D mesh network, the first and fourth layer have the same probability distribution, as do the second and third layer.

In short, $\boldsymbol{P}_{..0}$ equals to $\boldsymbol{P}_{..3}$, and $\boldsymbol{P}_{..1}$ equals to $\boldsymbol{P}_{..2}$. As shown in (10), the four components of $\boldsymbol{P}$ can be represented by $\boldsymbol{P_0}$ and $\boldsymbol{P_1}$ which are illustrated in (11) and (12) respectively.

$$\begin{aligned} \boldsymbol{P}_{..0} &= \boldsymbol{P_0} \\ \boldsymbol{P}_{..1} &= \boldsymbol{P_1} \\ \boldsymbol{P}_{..2} &= \boldsymbol{P_1} \\ \boldsymbol{P}_{..3} &= \boldsymbol{P_0} \end{aligned} \tag{10}$$

$$\boldsymbol{P_0} = \begin{bmatrix} 0.0128 & 0.0145 & 0.0145 & 0.0128 \\ 0.0145 & 0.0165 & 0.0165 & 0.0145 \\ 0.0145 & 0.0165 & 0.0165 & 0.0145 \\ 0.0128 & 0.0145 & 0.0145 & 0.0128 \end{bmatrix} \tag{11}$$

$$\boldsymbol{P_1} = \begin{bmatrix} 0.0145 & 0.0165 & 0.0165 & 0.0145 \\ 0.0165 & 0.0192 & 0.0192 & 0.0165 \\ 0.0165 & 0.0192 & 0.0192 & 0.0165 \\ 0.0145 & 0.0165 & 0.0165 & 0.0145 \end{bmatrix} \tag{12}$$

Using the distribution of p_i, we can calculate the proportion of blocks mapped from memory for each bank. For a $4\times4\times4$ mesh network, the bank address field in the physical memory address requires 6 ($log_2 64$) bits in order to map the block directly to the corresponding bank. In other words, memory-to-LLC mapping is interleaved with 64 blocks. In order to map blocks in proportion to the probability distribution $\boldsymbol{P}$, it is necessary to expand the width of the bank address field to rearrange the memory-to-LLC mapping. The longer the bank address field, the closer to the ideal distribution it is possible to arrange the practical mapping (at the cost of an increase in complexity). By trading off between accuracy and complexity, we expand the bank address field to 10 bits, which can interleave the mapping with 1024 blocks. We then multiply the probability distribution matrix $\boldsymbol{P}$ by the number of blocks in a given mapping interval and obtain the distribution of the number of blocks in said interval. We use matrix $\boldsymbol{B}$ to denote the distribution of the number of blocks such that $\boldsymbol{B}_{..i}$ represents the ith component of $\boldsymbol{B}$ in the third dimension. As shown in (13), $\boldsymbol{B}$ can be calculated from $\boldsymbol{P}$, and the four components of $\boldsymbol{B}$ can be represented by $\boldsymbol{B_0}$ and $\boldsymbol{B_1}$, which are in turn illustrated in (14) and (15) respectively.

$$\boldsymbol{B} = \boldsymbol{P} \times 2^{10} \qquad \begin{aligned} \boldsymbol{B}_{..0} &= \boldsymbol{B_0} \\ \boldsymbol{B}_{..1} &= \boldsymbol{B_1} \\ \boldsymbol{B}_{..2} &= \boldsymbol{B_1} \\ \boldsymbol{B}_{..3} &= \boldsymbol{B_0} \end{aligned} \tag{13}$$

$$\boldsymbol{B_0} = \begin{bmatrix} 12 & 15 & 15 & 12 \\ 15 & 17 & 17 & 15 \\ 15 & 17 & 17 & 15 \\ 12 & 15 & 15 & 12 \end{bmatrix} \tag{14}$$

$$\boldsymbol{B_1} = \begin{bmatrix} 15 & 17 & 17 & 15 \\ 17 & 20 & 20 & 17 \\ 17 & 20 & 20 & 17 \\ 15 & 17 & 17 & 15 \end{bmatrix} \tag{15}$$

The block distribution $\boldsymbol{B}$ for a $4\times4\times4$ mesh network is visualized in Figure 3. due to the symmetry of a $4\times4\times4$ 3D mesh cube, it is possible to divide the network into eight parts,

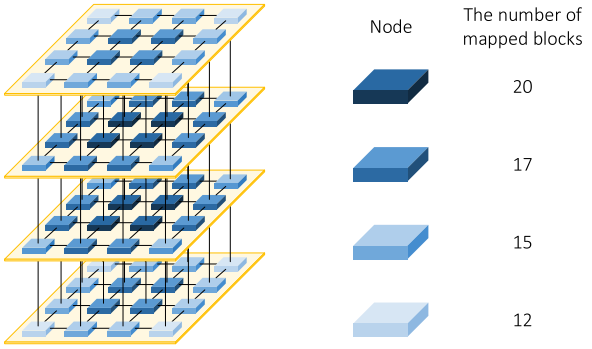

FIGURE 3. The block distribution for a 4 × 4 × 4 mesh network.

each of which is a $2\times2\times2$ mesh subcube including eight nodes (as shown in Figure 4a). Observing the distribution $\boldsymbol{P}$ and $\boldsymbol{B}$, we can determine that each subcube follows the same block distribution pattern: the closer to the central region of the network, the more blocks that should be mapped to the bank. Accordingly, we can perform a similar block remapping scheme on each part. Considering the distribution of the number of blocks for S-NUCA, 1024 blocks should be uniformly distributed on 64 banks by block-interleaving, and each bank is mapped with 16 blocks. Compared with the distribution in S-NUCA, we can remap some blocks from peripheral banks to central banks and keep the majority of blocks unchanged. This remapping scheme is able to maintain the minimum extra costs in hardware implementation. The following analysis will concentrate on one part of the network (the selected subcube shown in Figure 4a) in order to concisely explain the block remapping scheme. The remapping scheme performed on the other seven subcubes is similar.

Figure 4b shows the bank ID of each node in the subcube, while Figure 4c depicts the corresponding number of mapped blocks for each bank and the block remapping scheme. In Figure 4c, an arrowed line denotes that a block originally mapped to the source bank in the mapping scheme of S-NUCA should be remapped to the destination bank. Because each bank is mapped with 16 blocks, banks 6/18/22/23 should be mapped with more blocks, and banks 2/3/7/19 with fewer blocks. Therefore, we move some blocks from banks 2/3/7/19 to banks 6/18/22/23. More specifically, the additional four blocks mapped to bank 22 consist of one block from banks 2/3/7/19 respectively, while the additional one block mapped to banks 6/18/23 comes from bank 3. Thus, every bank in the subcube is mapped with the exact number of blocks.

We will next consider how the remapping scheme should be implemented. The 1024 blocks within a mapping interval can be divided sequentially into 16 groups, each of which includes 64 blocks. Because the bank address has been expanded to 10 bits, we divide the bank address into two parts, i.e., *bank index* and *bank tag*, which occupy 6 bits and 4 bits respectively as shown in Figure 4d. The bank index indicates the original bank ID in the mapping scheme of S-NUCA, and

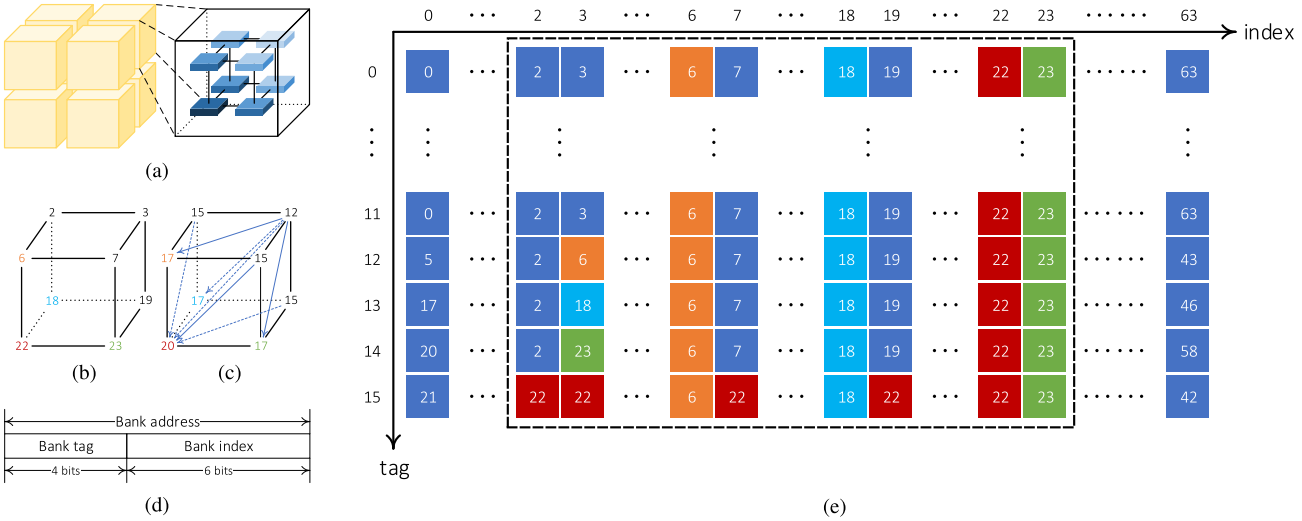

FIGURE 4. The remapping scheme for a 4 × 4 × 4 mesh network. (a) The 4 × 4 × 4 mesh network can be divided into eight parts, each of which is a 2 × 2 × 2 subcube. (b) The bank ID of each node in the subcube. The colored nodes indicate that the banks should be mapped with more blocks. (c) The number of mapped blocks for each bank and the block remapping scheme. An arrowed line denotes that a block that was originally mapped to the source bank in the S-NUCA mapping scheme of should be remapped to the destination bank. (d) The bank address field under the remapping scheme. (e) The detailed remapping scheme design in a mapping interval (i.e., 1024 blocks). The horizontal and vertical axes indicate the bank index and bank tag respectively, and the part in the dashed box shows the mapping design for the subcube. The number label on each block represents the actual mapped bank ID.

the bank tag stands for the group number. In order to maintain accordance with the block remapping scheme, we remap some blocks from peripheral banks (banks 2/3/7/19) to central banks (banks 6/18/22/23). The detailed remapping design in a mapping interval is illustrated in Figure 4e. In Figure 4e, the horizontal and vertical axes indicate the bank index and bank tag respectively, and the part in the dashed box shows the mapping design for the subcube. To move the additional four blocks to bank 22, we remap the blocks that originally belonged to banks 2/3/7/19 to bank 22 in the 16th group (i.e., the tag is equal to 15). In addition, we remap the blocks that originally belonged to bank 3 to banks 6/18/23 respectively in the 13/14/15th group (i.e., the tag is equal to 12/13/14).

Although the above analysis and results are derived for a $4 \times 4 \times 4$ mesh network, it is also possible to obtain a similar conclusion and corresponding probability distribution $\boldsymbol{P}$ and block distribution $\boldsymbol{B}$ for networks of other sizes. In general, the trend of distribution is the same, i.e., the central banks should be mapped with more blocks, which can be remapped from the peripheral banks. This enables us to design and implement a similar remapping scheme.

2) IMPACT ON CACHE CONTROLLER

Since we change the uniform mapping scheme, the bank ID cannot directly be obtained from the bank address field. Instead, as shown in Figure 5, it is required to add an address mapping module (*addr_map*) inside each L1 cache controller, which is responsible for mapping the bank address to the corresponding bank ID. Every time the L1 cache controller issues a coherence request (e.g. *GetS*, *GetM*, etc.), the module will calculate the bank ID that the cache block belongs, and thus the controller can delivery the correct node address to the network interface. This module is a simple logic unit and can be easily implemented by a lookup table.

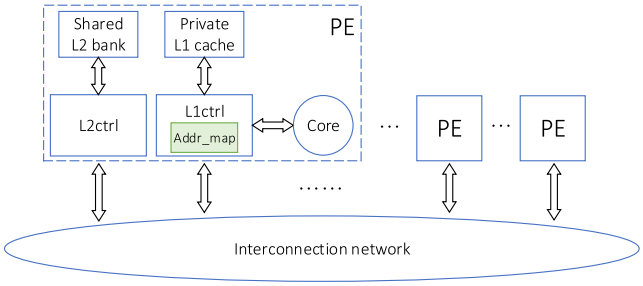

FIGURE 5. The address mapping module in each L1 cache controller.

B. NON-UNIFORM LINK DISTRIBUTION

As stated in the previous subsection, the memory mapping scheme for 3D mesh-based NUCA is redesigned to equalize the non-contention latencies of cache access. To further improve the cache access fairness, it is necessary to consider how the impact of contention latency on cache access fairness might be reduced. A major source of contention latency is link competition. However, in a 3D mesh-based network, the degrees of competition differ for each link due to the different positions occupied by links. In [35], the traffic requirements per link for the mesh network are analyzed, and it is found that traffic load on the link increases with proximity to the central region of the mesh network. An unbalanced traffic load can aggravate the variance of contention latencies of cache access. Hence, redesigning the link distribution and

increasing the communication capacity of those links with heavier traffic load can alleviate the degree of variance for contention latencies. However, it should be noted that the vertical links impose a larger area overhead than the horizontal links due to the bonding pad requirements [5], [36], meaning that it is not cost-effective to intensify the communication capacity of links in the vertical direction. Taking this into consideration, it is preferable to redesign the link distribution in the horizontal direction. In the following discussion, the non-uniform link distribution will be presented.

1) SCHEME

We will first need to analyze the traffic loads distributed on each link. Although the related analysis has been represented in [35], the results of this work are based on the original S-NUCA memory mapping for a 2D mesh network. Because we have changed the memory mapping, it is therefore necessary to recalculate the traffic load based on the proposed memory mapping scheme for a 3D mesh network described in Section IV-A.

Suppose the size of the 3D mesh network is $M \times N \times L$, which corresponds to the X-, Y-, and Z-axes respectively as shown in Figure 2. Here we are concerned with the links in the horizontal direction, and these horizontal links can be divided into two groups, i.e., links in the X- and Y-dimensions. The sizes of the matrices to represent the links in the X- and Y-dimensions are $(M-1) \times N \times L$ and $M \times (N-1) \times L$, respectively. To simulate the memory mapping scheme for cache access fairness, assume that each core sends 1024 messages to all banks, corresponding to the distribution of the number of blocks $\boldsymbol{B}$ as shown in (13). We then need to count the number of messages through each link to represent the traffic load. Algorithm 1 illustrates the procedure for calculating the traffic load distributed on each link. It quantitatively analyzes the traffic distributions (represented by $\boldsymbol{X}$ and $\boldsymbol{Y}$) on each link under the specific memory-to-LLC mapping scheme. In Algorithm 1, line 7 to 18 is the main part for simulating the all to all communication corresponding to the distribution $\boldsymbol{B}$, and each iteration represents that the core (i, j, k) issues $b_{m,n,l}$ messages to the bank (m, n, l). Line 9 to 12 and line 13 to 16 are responsible for calculating the traffic load in the X- and Y-dimensions respectively. The *minmax*[2] function is used to locate the range of links.

According to Algorithm 1, we first calculate the traffic load distribution in the X- and Y-dimensions for a $4 \times 4 \times 4$ mesh network, as illustrated in (16) and (17) respectively. Two points of interest can be observed here. Firstly, the four components of $\boldsymbol{X}$ ($\boldsymbol{Y}$) are the same; in other words, the traffic load distribution on each layer of the network follows the same pattern. This is because the routing algorithm is XYZ-DOR, meaning that the messages always traverse along the X- and Y-dimensions first and get through the vertical links last. Therefore, the vertical traffic does not affect the horizontal traffic pattern. Secondly, links closer to the central region on each layer are utilized more frequently. This is due to the additional traffic introduced by the remapping scheme, which is biased toward the central region of the network. As a result, in order to alleviate the degree of variance for contention latencies, we should intensify the communication capacity of the more highly utilized links and implement the same link distribution on each layer.

[2]The *minmax*(x, y) function that appears in the algorithm returns a vector of minimum and maximum values between the inputs x and y.

Algorithm 1 Calculate the Traffic Load Distributed on Each Link

Input:
The size of the mesh network, $M \times N \times L$;
The distribution of the number of blocks, $\boldsymbol{B} = [b_{i,j,k}]_{M \times N \times L}$.

Output:
The traffic load distribution of the links in the X-dimension, $\boldsymbol{X} = [x_{i,j,k}]_{(M-1) \times N \times L}$;
The traffic load distribution of the links in the Y-dimension, $\boldsymbol{Y} = [y_{i,j,k}]_{M \times (N-1) \times L}$.

1: **for** $(i, j, k) \Leftarrow (0, 0, 0) \rightarrow (M-2, N-1, L-1)$ **do**
2: $\quad x_{i,j,k} \Leftarrow 0$
3: **end for**
4: **for** $(i, j, k) \Leftarrow (0, 0, 0) \rightarrow (M-1, N-2, L-1)$ **do**
5: $\quad y_{i,j,k} \Leftarrow 0$
6: **end for**
7: **for** $(i, j, k) \Leftarrow (0, 0, 0) \rightarrow (M-1, N-1, L-1)$ **do**
8: $\quad$ **for** $(m, n, l) \Leftarrow (0, 0, 0) \rightarrow (M-1, N-1, L-1)$ **do**
9: $\quad\quad (min, max) \Leftarrow minmax(i, m)$
10: $\quad\quad$ **for** $q \Leftarrow min \rightarrow max - 1$ **do**
11: $\quad\quad\quad x_{q,j,k} \Leftarrow x_{q,j,k} + b_{m,n,l}$
12: $\quad\quad$ **end for**
13: $\quad\quad (min, max) \Leftarrow minmax(j, n)$
14: $\quad\quad$ **for** $q \Leftarrow min \rightarrow max - 1$ **do**
15: $\quad\quad\quad y_{m,q,k} \Leftarrow y_{m,q,k} + b_{m,n,l}$
16: $\quad\quad$ **end for**
17: $\quad$ **end for**
18: **end for**
19: **return** $\boldsymbol{X}, \boldsymbol{Y}$

$$\boldsymbol{X}_{..i} = \begin{bmatrix} 1496 & 2048 & 1496 \\ 1496 & 2048 & 1496 \\ 1496 & 2048 & 1496 \\ 1496 & 2048 & 1496 \end{bmatrix} \quad (0 \le i \le 3) \tag{16}$$

$$\boldsymbol{Y}_{..i} = \begin{bmatrix} 1376 & 1616 & 1616 & 1376 \\ 1888 & 2208 & 2208 & 1888 \\ 1376 & 1616 & 1616 & 1376 \end{bmatrix} \quad (0 \le i \le 3) \tag{17}$$

The traffic load distributions $\boldsymbol{X}$ and $\boldsymbol{Y}$ reflect the degree of contention of each link under the location-aware memory mapping calculated in Section IV-A. It is possible to reduce the contention latencies for these highly utilized links by increasing the channel width. By referring to the width of those links with minimum traffic load in the distribution

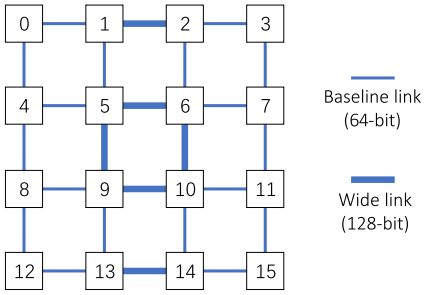

FIGURE 6. The channel width distribution on each layer for a 4 × 4 × 4 mesh network. Here, take the top layer as an example.

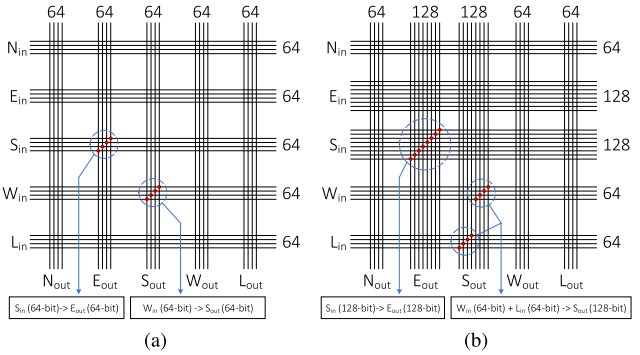

FIGURE 7. The difference between the crossbar architecture of the baseline and F-NUCA. (a) Baseline design. (b) Crossbar design in F-NUCA.

$\boldsymbol{X}$ and $\boldsymbol{Y}$, each link is given an appropriate *relative width* in proportion to the traffic load. The minimum traffic load within the distribution $\boldsymbol{X}$ and $\boldsymbol{Y}$ is ℓ:

$$\ell = min\{\boldsymbol{X}, \boldsymbol{Y}\} \tag{18}$$

Suppose the basic channel width (i.e., the width of the link with the minimum traffic load) is ω. The relative width of each link can thus be derived from the corresponding traffic load divided by ℓ. Further, we can now obtain the actual channel width, which is the product of relative width and ω:

$$\begin{aligned} \boldsymbol{X_w} &= \frac{\boldsymbol{X}}{\ell} \cdot \omega = [round(\frac{x_{i,j,k}}{\ell}) \cdot \omega]_{(M-1)\times N\times L} \\ \boldsymbol{Y_w} &= \frac{\boldsymbol{Y}}{\ell} \cdot \omega = [round(\frac{y_{i,j,k}}{\ell}) \cdot \omega]_{M\times (N-1)\times L} \end{aligned} \tag{19}$$

Note that the channel width of each link is a multiple of the baseline channel width ω. Multiple flits can be sent over a wide link in parallel. More specifically, assume that the baseline channel width is 64-bit; as such, the channel width distribution on each layer can be calculated. Example calculations based on the traffic load distribution $\boldsymbol{X}$ and $\boldsymbol{Y}$ are presented in Figure 6.

2) IMPACT ON ROUTER

Utilization of the wide links requires minor modifications to the crossbar and switch allocator (SA) of the routers. Our design is inspired by the XShare and HeteroNoC techniques, which are proposed in [37] and [38] respectively, but improved in transmission efficiency and hardware implementation. Here, we will talk about the impacts on crossbar and SA of the router.

When communication takes place between two routers connected by a wide link, two flits can be simultaneously sent over the wide link. Such a case is depicted in Figure 7, where we illustrate the difference between the crossbar design of the baseline and F-NUCA[3]. Take router 5 as an example in this case. In Figure 7a, the baseline crossbar can allow that one flit in input port S goes to output port E as well as one flit in input port W goes to output port S. One output port can only be mapped to one input port, and vice versa. In contrast, because the channel width is doubled in the east and south in F-NUCA (as shown in Figure 6), the corresponding width is also doubled in the crossbar (E_{in}/S_{in}/E_{out}/S_{out}). Hence, two flits from different VCs within a single input port ($S_{in} \rightarrow E_{out}$) or two flits from different input ports (W_{in}+$L_{in} \rightarrow S_{out}$) can be sent together. Different from XShare and HeteroNoC, it is also supported that two flits from different VCs within a single input port respectively go to two different output ports (e.g. $E_{in} \rightarrow N_{out} + W_{out}$), and such a case benefits from the improved SA design in F-NUCA.

Figure 8a shows the switch allocator of the baseline router. Suppose that the baseline router has V virtual channels, P input/output ports. $r_i^v[]$ represents VC request signals from ith input port. The allocation includes two stages, i.e., input and output arbitrations. The input arbitrations are performed in parallel by P input arbiters to select one input VC of each arbiter. $g_i^v[]$ represents VC grant signals from ith input port to indicate which VC is granted. Next, the decoder unit translates the granted VC request into the corresponding output port request, which is propagated to output arbiters. Then the output arbitrations are performed in parallel by the P output arbiters to select one input port for each arbiter. $g_i^p[]$ represents the input port grant signals to ith output port to indicate which input port is granted. By combining $g_i^v[]$ and $g_i^p[]$ together, the switch allocator accomplishes the matching of the input ports to output ports.

In F-NUCA, the ports connected by wide links need additional arbiters to allow more flits to cross the ports. Figure 8b shows the SA design in F-NUCA. Take a router with one port (Pth port) connected by the wide link as an example. Compared with the baseline design, an additional $V : 1$ is used in the input arbiters of Pth input port. As for the output arbiters, the size of each arbiter is increased to $(P + 1) : 1$, and an additional $(P + 1) : 1$ arbiter is also required for the Pth output port. Actually, the two input/output arbiters of Pth port can be seen as an $N : 2$ arbiter ($N = V$ õr $P + 1$) in combination. In such an $N : 2$ arbiter, the second arbiter receives the *mask* signals (indicating which request has been granted) directly from the interior of the first arbiter to avoid

[3]We only show the five horizontal ports (North/South/East/West/Local) in the crossbar in the interests of clear demonstration. In fact, the crossbar should have seven ports in the symmetric 3D NoC, including the Up/Down ports.

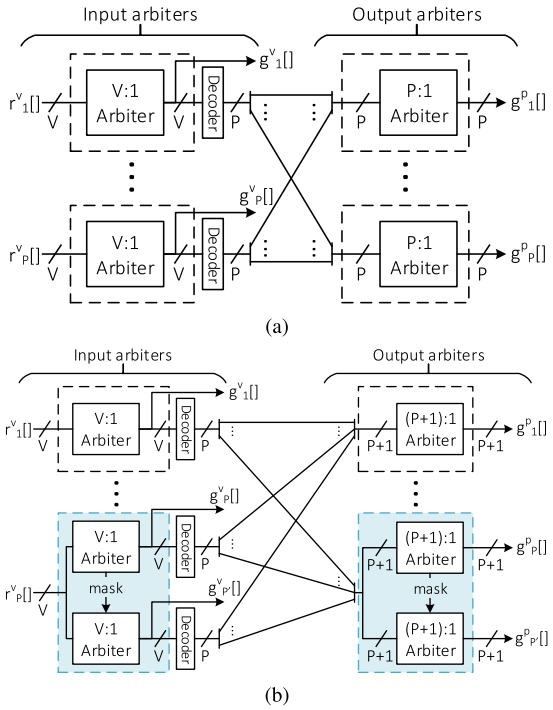

FIGURE 8. The SA organization in baseline and F-NUCA. (a) Baseline design. (b) SA design in F-NUCA.

picking the same flit. Thus the $N : 2$ arbiter can grant two requests simultaneously.

Compared with the SA design in XShare or HeteroNoC, the simultaneous transmission from a single input port does not require combining the two flits toward the same output port. In contrast, they can be sent to different output ports due to the independence in input arbitration. Besides, the $N : 2$ arbiter can avoid the conflict of arbitration between the two sub-arbiters, and thus further improve the efficiency of allocation.

C. HARDWARE COST

The hardware overhead of F-NUCA is primarily due to the larger crossbar and the extra arbiters. The synthesis is based on the open-source NoC router RTL code [39] using Synopsys Design Compiler with SMIC's 130nm library under typical-case voltage and temperature conditions (1.2V and 25 °C, respectively). Compared with the baseline, F-NUCA extends the critical path, mainly due to the increased complexity of arbiters. As a result, the synthesis results demonstrate that the baseline router can be run up to 500MHz and F-NUCA up to 462MHz, which is operated by the network of S-NUCA and F-NUCA in the simulation experiments. The synthesis results demonstrate that the area of the router connected by one wide link increases by 2.9%, and the the area of the router connected by two wide links increases by 5.7%. Hence, the baseline total area of 64 routers ($4\times4\times4$ mesh network) is $17.7245mm^2$ ($= 64 \times 0.2769mm^2$), and the total area for F-NUCA is $18.1110mm^2$ ($= 32\times0.2769mm^2 + 16 \times 0.2851mm^2 + 16 \times 0.2928mm^2$), which is increased by 2.1%.

TABLE 1. The baseline simulation configuration.

Simulation configuration	
ISA	Alpha
Number of cores	32/64
Cache coherency protocol	MESI
Cacheline size	64 bytes
L1 instruction cache	32 kB, 2-way associative
L1 data cache	64 kB, 2-way associative
L2 cache bank	1 MB, 8-way associative
Network topology	3D symmetric mesh
Router	2-stage wormhole flit-switched
Number of VCs	4
Routing algorithm	XYZ-DOR
Flit size	64 bits
Control packet size	8 bytes
Data packet size	72 bytes
Channel width	64 bits (baseline) 128 bits (wide)
Main memory	4GB DRAM, DDR3-1600
DRAM interface	Corner placement (bottom layer)

V. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETUP

In order to conduct an evaluation of F-NUCA's performance compared with the traditional S-NUCA, we build the proposed architecture based on *gem5* [40], a popular full-system simulator enabled with *Garnet2.0* [41] (to model the interconnection network) and *Ruby* (to model the memory subsystem). The performance simulations include two phases. First, we switch gem5 to network-only mode in order to perform a network simulation with synthetic traffic, then evaluate the network performance across three different network scales ($4\times4\times2/4\times4\times4/6\times6\times4$). Second, we switch gem5 to full-system mode to perform a full simulation with real benchmarks and observe the system performance when running different programs across two different network scales ($4\times4\times2/4\times4\times4$). The simulation configuration is illustrated in Table 1.

B. SIMULATION WITH SYNTHETIC TRAFFICS

Simulation with synthetic traffics is performed using gem5's network-only mode. In this mode, the simulated cores do not execute real instructions and instead act simply as traffic injectors that inject packets into the network. Therefore, the system is ISA-agnostic, and the memory subsystem is non-functional. This mode focuses on evaluating the network performance of the simulated system. The baseline network simulation configuration is illustrated by the network-related portion of Table 1.

We investigate the simulated systems (F-NUCA and S-NUCA) under different injection rates at a fixed interval (0.005 packets/cycle) to observe the network performance, Although a simulation utilizing synthetic traffics does not include the memory-to-LLC mapping mechanism, we can build the specific traffic distribution corresponding to the bank by accessing probability distribution $\boldsymbol{P}$ (as shown in (10)) rather than the non-uniform mapping scheme.

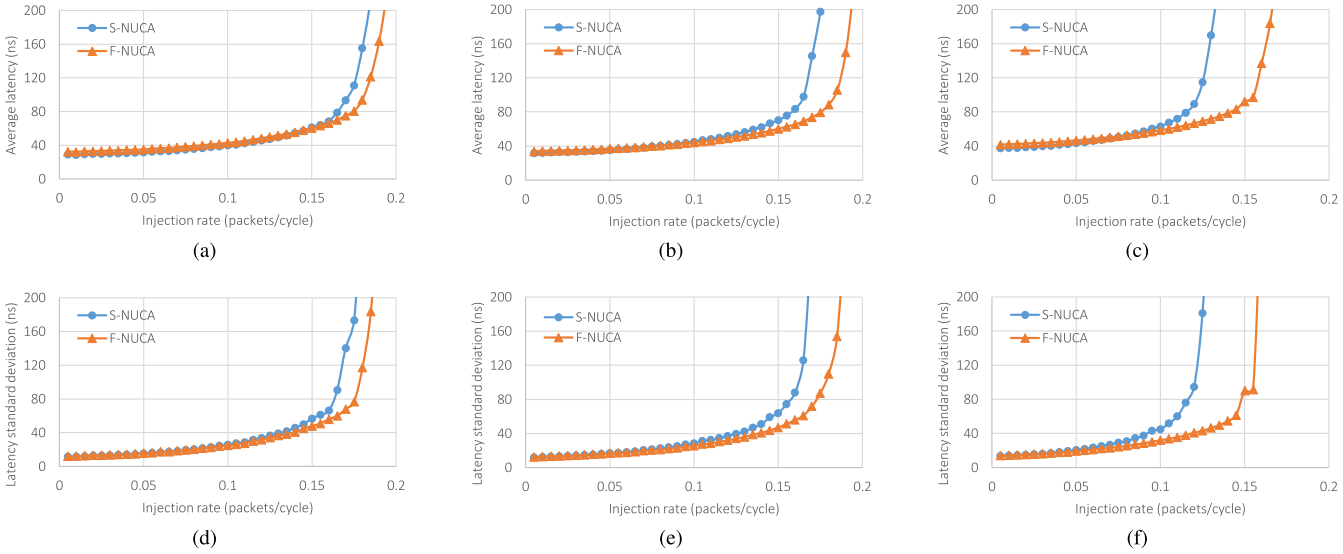

FIGURE 9. Network performance under different injection rates for 4 × 4 × 2, 4 × 4 × 4, and 6 × 6 × 4 mesh networks. (a) AL(4 × 4 × 2). (b) AL(4 × 4 × 4) (c) AL(6 × 6 × 4) (d) LSD(4 × 4 × 2). (e) LSD(4 × 4 × 4). (f) LSD(6 × 6 × 4).

In addition, because the network latency dominates over the cache access latency, the degree of equalization in network latency can largely reflect the cache access fairness. Therefore, we can evaluate the fairness by observing the network latency. The evaluation is performed across three different network scales, namely $4 \times 4 \times 2$, $4 \times 4 \times 4$, and $6 \times 6 \times 4$.

The experimental results are presented in Figure 9. The network performance is expressed in terms of two metrics: *average latency* (AL) and *latency standard deviation* (LSD). LSD measures the amount of variation from the average latency. A lower standard deviation indicates that the latencies tend to be closer to the mean; therefore, LSD can be used to evaluate fairness. In Figure 9, F-NUCA exhibits 2.95%/21.55%/25.43% lower average latencies compared with S-NUCA at the saturation point (0.16/0.16/0.12) for $4\times4\times2/4\times4\times4/6\times6\times4$ networks respectively. In addition, F-NUCA also outperforms S-NUCA in terms of fairness, as the LSD is reduced by 15.26%/36.56%/56.89% at the saturation point. From this we can see that the improvement in LSD is more pronounced than AL. Even though S-NUCA achieves a little better than F-NUCA in AL when the injection rate is very low, F-NUCA also outperforms S-NUCA in LSD, which indicates that F-NUCA can effectively reduce variation and improve fairness.

Note that two trends can be observed in Figure 9. First, the performance improvement becomes more significant as the injection rate increases. This is because contentions in network are exacerbated under higher injection rate conditions, which has a detrimental effect on fairness, an issue that can be better alleviated by our approach. Second, performance improvements are seen to grow more pronounced as the network size is scaled up; this is due to the fact that the fairness problem is more serious for large-scale networks.

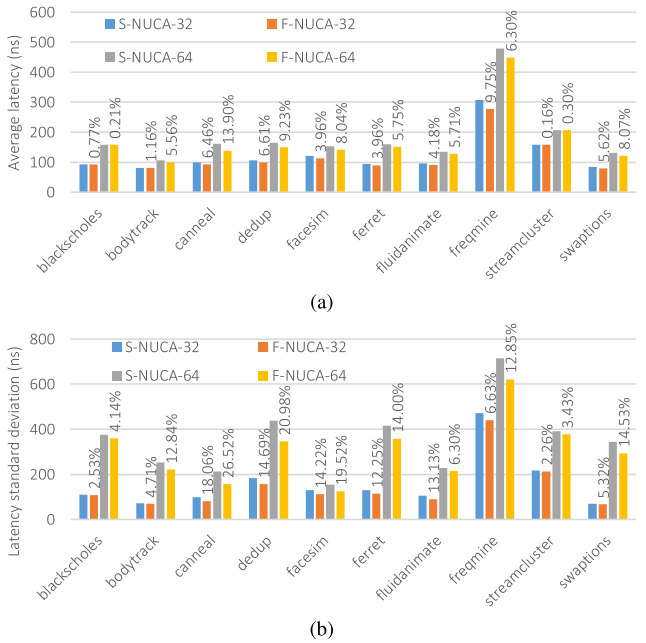

FIGURE 10. L2 cache access performance comparison between S-NUCA and F-NUCA for 4 × 4 × 2 (32) and 4 × 4 × 4 (64) mesh networks. (a) AL. (b) LSD.

C. SIMULATION WITH BENCHMARKS

We choose ten programs from the available *PARSEC 2.1* [42] benchmarks to perform our experiments. The simulation is performed under gem5's full-system mode. In this mode, the simulator first boots up a Linux kernel, then loads the program's execution script after entering the system. For each PARSEC program, the simulation is divided into three phases: an initial serial phase, a parallel phase, and a final serial phase. The parallel phase, also called the *region of*

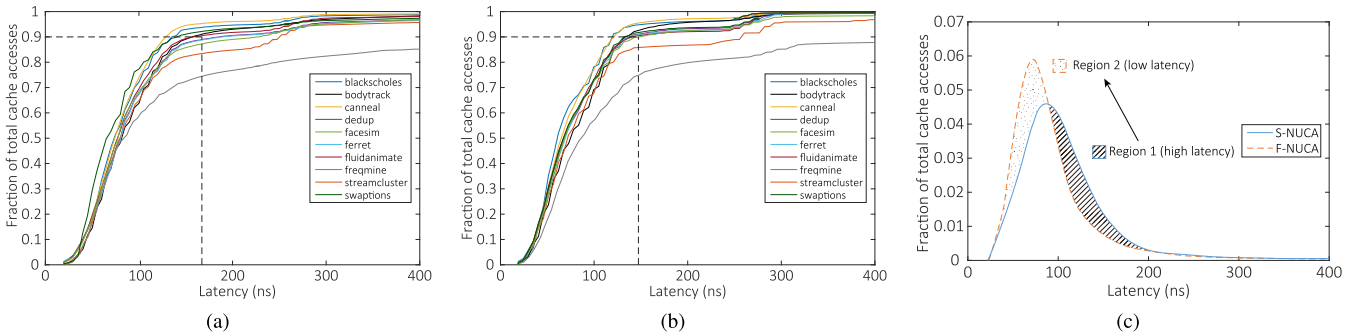

FIGURE 11. (a) The CDFs of the cache accesses for 4 × 4 × 4 mesh network in S-NUCA. The values on the x-axis are the cache access latencies (in nanoseconds), and the y-axis denotes fraction of the total number of cache accesses. (b) The CDFs of the cache accesses for 4 × 4 × 4 mesh network in F-NUCA. (c) The latency distribution of the cache accesses for one of the programs (*canneal*). (a) CDFs in S-NUCA. (b) CDFs in F-NUCA. (c) PDF for *canneal*.

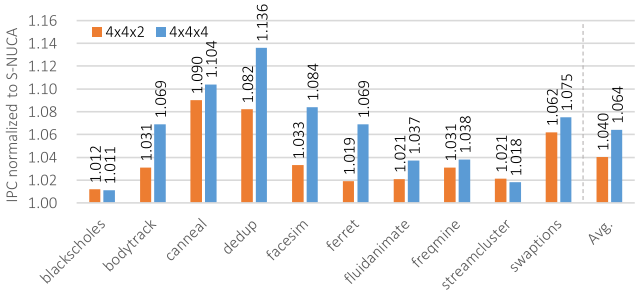

FIGURE 12. The IPC performance normalized to S-NUCA for 4 × 4 × 2 and 4 × 4 × 4 mesh networks.

interest (ROI), is the phase that truly reflects the characteristics of the program; therefore, the simulation results we collected are based on the ROI part of each program. Table 1 shows the configuration for the full-system simulation. The evaluation is performed across two different network scales, namely 4 × 4 × 2 (32 cores) and 4 × 4 × 4 (64 cores). Each core is alpha-ISA, in-order and single-thread architecture.

Figure 10 presents the cache access (L2) performance for each program's simulation. We can observe that F-NUCA outperforms S-NUCA with each program. The average reductions in AL/LSD across all ten programs are 4.64%/9.38% and 6.31%/13.51% for 4 × 4 × 2 and 4 × 4 × 4 networks respectively. It can be seen that F-NUCA achieves notable improvements in cache access performance, particularly in LSD, which benefits more from the fairness-oriented design. However, performance improvement in this simulation is not significant as the experimental results of simulation with synthetic traffics. This is because that F-NUCA is mainly optimized for those workloads with more balanced communications, and the communications in some PARSEC programs are not balanced [43], such as *blackscholes*/*streamcluster* which show strong communication between a few cores while revealing weak communication between the rest. Hence, the improvements for these programs are less obvious.

To further observe the effect on cache access fairness, we collect 1M cache access (L2) latencies with each program for the 4 × 4 × 4 mesh network. Based on the collected latencies, we plot the cumulative distribution function (CDF), as shown in Figure 11a and Figure 11b, which reveals the cumulative distribution of cache access latencies for S-NUCA and F-NUCA respectively. Only the main parts of the curves are presented here, due to the fact that most cache access latencies are less than 400ns. The x-axis represents the cache access latency (in nanoseconds), while the y-axis denotes the fraction of the total number of cache accesses. In more detail, one point $(x, F(x))$ on curve F signifies that the fraction of the total number of cache accesses with latencies less than x is $F(x)$. From Figure 11a, we can observe that the latency of 90% of cache accesses for S-NUCA is less than 170ns for the majority of programs. By contrast, results for F-NUCA show that the same proportion of cache accesses (90%) are less than 150ns, as shown in Figure 11b. In short, F-NUCA successfully produces fewer high-latency cache accesses than S-NUCA.

Figure 11c plots the probability density function (PDF) of the cache accesses issued by one of the programs (*canneal*) for both S-NUCA and F-NUCA. As can be seen from this figure, F-NUCA prompts the distribution of high latency (Region 1) to transfer to the new distribution of low latency (Region 2). Therefore, F-NUCA makes cache access latencies more concentrated, thus achieving the stated goal of equalizing cache access latencies.

Figure 12 illustrates the IPC (instructions per cycle) normalized to S-NUCA. Here we can see that F-NUCA outperforms S-NUCA on the whole, and improves by about 4.0%/6.4% on average for 4 × 4 × 2 and 4 × 4 × 4 networks compared with S-NUCA. Similar to the performance analysis presented in the previous subsection, it can also be observed that the improvement is more pronounced for larger-scale network; again, this is because the exacerbation of cache access fairness issues are more serious for large-scale networks, and our fairness-oriented approach can thus play a more significant role in reducing the impact of overhigh cache accesses on system performance. Another observation is that the performance improvements are not balanced for each program. The reason lies in the communication pattern

of each program [43]. Because our design is mainly optimized for the balanced workload, the programs with strong all to all communication (*canneal*/*dedup*) can be achieved more benefits from the optimization. By contrast, some programs like *blackscholes*/*streamcluster* shows less balanced communication pattern but strong communication between neighboring cores are improved less in performance.

VI. CONCLUSION

The aggravation of cache access fairness issues causes an increased number of cache accesses with overhigh latencies. These can cause a bottleneck in system performance, especially for large-scale 3D architecture. In this context, cache access fairness is expected to be one of the major factors in optimizing communication overheads and thereby improving system performance. In this work, we study the factors affecting cache access fairness in the context of 3D NUCA-based CMPs and attempt to equalize the cache access latencies on two fronts, i.e., non-contention and contention latencies. The proposed location-aware memory mapping scheme can equalize the non-contention latencies, while the redistribution of channel width alleviates the contention latency variance. This fairness-oriented co-design is called *fair-NUCA* (F-NUCA). We evaluate the hardware cost and perform the simulation experiments for F-NUCA. Our simulation results indicate that F-NUCA is more effective than the traditional architecture. Evaluation using ten diverse programs demonstrates that F-NUCA can effectively improve cache access fairness. The reductions in AL/LSD are up to 4.64%/9.38% and 6.31%/13.51% on $4 \times 4 \times 2$ and $4 \times 4 \times 4$ 3D mesh networks respectively. As for system throughput, F-NUCA achieves 4.0%/6.4% improvements.

In conclusion, this paper demonstrates the importance of cache access fairness for system performance and makes a case for the design of a non-uniform 3D architecture that combines the designs of memory mapping and NoC. In the anticipated future, we will study how the cache protocol and operation (e.g. update, invalidate, etc.) have an impact on the cache access fairness.

REFERENCES

[1] N. E. Jerger, T. Krishna, and L.-S. Peh, *On-Chip Networks* (Synthesis Lectures on Computer Architecture), 2nd ed. San Rafael, CA, USA: Morgan & Claypool, 2017.

[2] K. Manna, S. Swami, S. Chattopadhyay, and I. Sengupta, "Integrated through-silicon via placement and application mapping for 3D mesh-based NoC design," *ACM Trans. Embedded Comput. Syst.*, vol. 16, Nov. 2016, Art. no. 24.

[3] S. J. Souri, K. Banerjee, A. Mehrotra, and K. C. Saraswat, "Multiple Si layer ICs: Motivation, performance analysis, and design implications," in *Proc. 37th Annu. Design Automat. Conf. (DAC)*, New York, NY, USA, 2000, pp. 213–220.

[4] S. Das, A. Fan, K.-N. Chen, C. S. Tan, N. Checka, and R. Reif, "Technology, performance, and computer-aided design of three-dimensional integrated circuits," in *Proc. Int. Symp. Phys. Design (ISPD)*, New York, NY, USA, 2004, pp. 108–115.

[5] J. Kim *et al.*, "A novel dimensionally-decomposed router for on-chip communication in 3D architectures," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2007, pp. 138–149.

[6] W. R. Davis *et al.*, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Design Test Comput.*, vol. 22, no. 6, pp. 498–510, Nov. 2005.

[7] Y. J. Hwang, J. H. Lee, and T. H. Han, "3D network-on-chip system communication using minimum number of TSVs," in *Proc. ICTC*, Sep. 2011, pp. 517–522.

[8] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," *SIGARCH Comput. Archit. News*, vol. 30, pp. 211–222, Oct. 2002.

[9] Y. Xie, G. H. Loh, B. Black, and K. Bernstein, "Design space exploration for 3D architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 2, no. 2, pp. 65–103, Apr. 2006.

[10] B. Black *et al.*, "Die stacking (3D) microarchitecture," in *Proc. 39th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2006, pp. 469–479.

[11] G. H. Loh, Y. Xie, and B. Black, "Processor design in 3D die-stacking technologies," *IEEE Micro*, vol. 27, no. 3, pp. 31–48, May/Jun. 2007.

[12] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 32–45, Jan. 2009.

[13] V. F. Pavlidis and E. G. Friedman, "3-D topologies for networks-on-chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 10, pp. 1081–1090, Oct. 2007.

[14] A. W. Yin, T. C. Xu, P. Liljeberg, and H. Tenhunen, "Explorations of honeycomb topologies for network-on-chip," in *Proc. 6th IFIP Int. Conf. Netw. Parallel Comput.*, Oct. 2009, pp. 73–79.

[15] S. Akbari, A. Shafiee, M. Fathy, and R. Berangi, "AFRA: A low cost high performance reliable routing for 3D mesh NoCs," in *Proc. Design, Automat. Test Eur. (DATE)*, San Jose, CA, USA, 2012, pp. 332–337.

[16] A. B. Ahmed and A. B. Abdallah, "Low-overhead routing algorithm for 3D network-on-chip," in *Proc. 3rd Int. Conf. Netw. Comput.*, Dec. 2012, pp. 23–32.

[17] A. B. Ahmed and A. B. Abdallah, "Graceful deadlock-free fault-tolerant routing algorithm for 3D network-on-chip architectures," *J. Parallel Distrib. Comput.*, vol. 74, no. 4, pp. 2229–2240, 2014.

[18] J. Liu, J. Harkin, Y. Li, and L. P. Maguire, "Fault-tolerant networks-on-chip routing with coarse and fine-grained look-ahead," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 2, pp. 260–273, Feb. 2016.

[19] A. Arora, M. Harne, H. Sultan, A. Bagaria, and S. R. Sarangi, "FP-NUCA: A fast NoC layer for implementing large NUCA caches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2465–2478, Sep. 2015.

[20] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches," in *Proc. 37th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Washington, DC, USA, Dec. 2004, pp. 319–330.

[21] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir, "Design and management of 3D chip multiprocessors using network-in-memory," in *Proc. 33rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Washington, DC, USA, 2006, pp. 130–141.

[22] N. Madan *et al.*, "Optimizing communication and capacity in a 3D stacked reconfigurable cache hierarchy," in *Proc. IEEE 15th Int. Symp. High Perform. Comput. Archit.*, Feb. 2009, pp. 262–274.

[23] J. Jung, K. Kang, and C.-M. Kyung, "Design and management of 3D-stacked NUCA cache for chip multiprocessors," in *Proc. 21st Ed. Great Lakes Symp. Great Lakes Symp. VLSI (GLSVLSI)*, New York, NY, USA, 2011, pp. 91–96.

[24] Y. Zhang *et al.*, "A survey of memory architecture for 3D chip multiprocessors," *Microprocess. Microsyst.*, vol. 38, no. 5, pp. 415–430, 2014.

[25] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aérgia: Exploiting packet latency slack in on-chip networks," in *Proc. 37th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2010, pp. 106–116.

[26] A. Sharifi, E. Kultursay, M. Kandemir, and C. R. Das, "Addressing end-to-end memory access latency in NoC-based multicores," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp. 294–304.

[27] M. Gorgues, D. Xiang, J. Flich, Z. Yu, and J. Duato, "Achieving balanced buffer utilization with a proper co-design of flow control and routing algorithm," in *Proc. 8th IEEE/ACM Int. Symp. Netw.-Chip (NoCS)*, Sep. 2014, pp. 25–32.

[28] Z. Lu and Y. Yao, "Aggregate flow-based performance fairness in CMPs," *ACM Trans. Archit. Code Optim.*, vol. 13, Dec. 2016, Art. no. 53.

[29] D. H. Kim *et al.*, "Design and analysis of 3D-MAPS (3D massively parallel processor with stacked memory)," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 112–125, Jan. 2015.

[30] M. Wordeman, J. Silberman, G. Maier, and M. Scheuermann, "A 3D system prototype of an eDRAM cache stacked over processor-like logic using through-silicon vias," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2012, pp. 186–187.
[31] D. Fick *et al.*, "Centip3De: A 3930 DMIPS/W configurable near-threshold 3D stacked system with 64 ARM Cortex-M3 cores," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2012, pp. 190–192.
[32] R. G. Dreslinski *et al.*, "Centip3De: A 64-core, 3D stacked, near-threshold system," in *Proc. IEEE Hot Chips 24 Symp. (HCS)*, Aug. 2012, pp. 1–30.
[33] C.-L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2008, pp. 164–169.
[34] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, 2013.
[35] Y. Zhang and A. K. Jones, "Non-uniform fat-meshes for chip multiprocessors," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, May 2009, pp. 1–8.
[36] S. Pasricha, "Exploring serial vertical interconnects for 3D ICs," in *Proc. 46th Annu. Design Automat. Conf. (DAC)*, New York, NY, USA, 2009, pp. 581–586.
[37] R. Das, S. Eachempati, A. K. Mishra, V. Narayanan, and C. R. Das, "Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs," in *Proc. IEEE 15th Int. Symp. High Perform. Comput. Archit.*, Feb. 2009, pp. 175–186.
[38] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A case for heterogeneous on-chip interconnects for CMPs," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2011, pp. 389–400.
[39] D. U. Becker, "Efficient microarchitecture for network-on-chip routers," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, Jul. 2012.
[40] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.
[41] 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)*, Apr. 2009, pp. 33–42.
[42] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. Int. Conf. Parallel Archit. Compil. Techn. (PACT)*, Oct. 2008, pp. 72–81.
[43] N. Barrow-Williams, C. Fensch, and S. Moore, "A communication characterisation of Splash-2 and Parsec," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2009, pp. 86–97.

ZICONG WANG was born in 1989. He received the B.S. degree in computer science and the M.S. degree in microelectronics from the National University of Defense Technology, Hunan, China, in 2008 and 2012, respectively, where he is currently pursuing the Ph.D. degree in microelectronics. His main research interests include networks-on-chip design and computer architecture.

XIAOWEN CHEN was born in 1982. He received the B.S. degree in microelectronics and the B.S. degree in computer science from the University of Electronic Science and Technology of China in 2005, and the Ph.D. degree in microelectronics from the National University of Defense Technology (NUDT), Hunan, China, in 2011. He is currently an Assistant Researcher in microprocessor design with NUDT. His research interests include computer architecture, microprocessor and DSP design, interconnection networks, and memory.

ZHONGHAI LU 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 was an Engineer in the area of electronic and embedded systems from 1989 to 2000. He is currently an Associate Professor with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology. He has authored over 130 peer-reviewed papers. His current research interests include interconnection networks, computer architecture, and real-time cyber-physical systems.

YANG GUO received the Ph.D. degree from the National University of Defense Technology, Hunan, China, in 1999. He is currently a Professor with the National University of Defense Technology, where he leads the Digital Signal Processor Group, and also the Director of the integrated circuits. He has authored or co-authored over 50 publications in journals and conference proceedings. His primary research interests include low-power VLSI circuits, microprocessor design and verification, and electronic design automation techniques for VLSI circuits.