A Lifetime Reliability-Constrained Runtime Mapping for Throughput Optimization in Many-Core Systems

Liang Wang, Ping Lv, Leibo Liu, Jie Han, Ho-fung Leung, Xiaohang Wang, Shouyi Yin, Shaojun Wei, Terrence Mak

***Abstract*—Due to technology scaling, lifetime reliability is becoming one of the major design constraints in the performance optimization of future many-core systems. Given a lifetime reliability constraint, the existing lifetime-constrained runtime mapping schemes often lead to low throughput because of the requirement to map all applications to compact regions. In this paper, we propose a runtime application mapping scheme, *LBRM*, that exploits a *borrowing strategy* to improve the throughput of many-core systems given a lifetime constraint. First, we propose using different strategies for mapping communication-intensive applications and computation-intensive applications. The lifetime reliability constraint can be relaxed in the local time scale when the communication requirement is high. The throughput is improved because the communication distance of communication-intensive applications is optimized while the waiting time of computation-intensive application is reduced. Then, we propose a method to effectively classify applications depending on the communication-to-computation ratio. A dynamic threshold is determined according to the current locations of available cores. Finally, we propose an improved neighborhood allocation scheme to reduce the communication cost in the task mapping. The experimental results show that compared to the state-of-the-art lifetime-constrained mapping, the proposed mapping scheme improves the throughput of many-core systems by 26% on average for synthetic task graphs and by 20% on average for realistic task graphs while the lifetime reliability is maintained within a constraint.**

***Index Terms*—Lifetime reliability, runtime mapping, many-core, throughput**

Manuscript received 19 Jan. 2018; revised 4 Apr. 2018; accepted 24 Jun. 2018. Date of publication XX, XX, XXXX; date of current version XX, XX, XXXX. This work was supported in part by the National Natural Science Foundation of China under Grant 61672317, and in part by National Science and Technology Major Project of the Ministry of Science and Technology of China under grant 2016ZX01012101, and in part by Natural Science Foundation of Guangdong Province 2018A030313166, in part by the Science and Technology Research Grant of Guangdong Province No. 2017A050501003, in part by Pearl River S&T Nova Program of Guangzhou No. 201806010038. This paper was recommended by Associate Editor Y. Shi. (*Corresponding author: Leibo Liu.*)

Liang Wang, Leibo Liu, Shouyi Yin and Shaojun Wei are with the Institute of Microelectronics, Tsinghua University, Beijing, China. Email: {lwang_cuhk, liulb, yinsy, wsj}@tsinghua.edu.cn

Ping Lv is with National Digital Switching System Engineering and Technological Research Center, China. Email: lp@ndsc.com.cn

Jie Han is with the Department of Electrical and Computer Engineering, University of Alberta, Canada. Email: jhan8@ualberta.ca

Ho-fung Leung is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China. Email: lhf@cuhk.edu.hk

Xiaohang Wang is with the School of Software Engineering, South China University of Technology, and Guangzhou Institute of Advanced Technology, Guangzhou, China. Email: xiaohangwang@scut.edu.cn

Terrence Mak is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, United Kingdom. Email: tmak@ecs.soton.ac.uk

I. Introduction

Aggressive technology scaling has enabled the integration of tens to hundreds of cores on a single chip [1], [2], which is known as many-core processor. The workloads on many-core systems generally exhibit a dynamic nature, i.e. applications with different characteristics arrive and leave the system dynamically. To better utilize the many-core resources, the resource manager requires an efficient runtime application mapping scheme, which should be lightweight and provide high performance under various resource constraints, such as thermal, power, lifetime reliability, and so forth. Application runtime mapping on many-core systems is a challenging problem because it is time consuming to find the optimal solution given various constraints while the information is available only when the applications arrive [3]. In addition to resource constraints, the performance requirements [4], internal and external congestion [5], [6] and fragmentation issues [7], [8] also complicate the design of runtime mapping.

Meanwhile, lifetime reliability is emerging as a major design concern for many-core systems [9]–[12]. The transistors are affected by various aging mechanisms, which can eventually result in permanent failures of cores. Mean time to failure (MTTF), a metric for lifetime reliability, is used for the expected lifetime of transistors. Dynamic reliability management (DRM), first proposed in [9], ensures a target lifetime reliability by controlling the system knobs such as voltage/frequency [13]–[15], power gating [16], task-to-core mapping [17], [18], and so forth. One specific feature of lifetime-constrained design is the *borrowing strategy* [13], which allows the lifetime constraint to be locally relaxed for better performance in the local time window while still meeting the constraint over the long-term scale. However, most prior studies exploit the *borrowing strategy* by DVFS in single-core or multi-/many-core processors [13]–[15]. Additionally, the *borrowing strategy* is less exploited for managing lifetime reliability through runtime application mapping. Since runtime application mapping is a different knob from DVFS, if the *borrowing strategy* is adopted, the design challenge of the application mapping is how to consider short-term relaxation of lifetime constraint when selecting a set of cores to map an application.

In this paper, we show that it is possible to improve the throughput by exploiting a *borrowing strategy* in the lifetime-

constrained design of many-core systems. We propose a runtime application mapping scheme named **L**ifetime **B**udgeting **R**eliability **M**anagement (*LBRM*), which aims to improve the global throughput of the system given a lifetime reliability constraint. The key idea of this mapping scheme is that a *borrowing strategy* is adopted to satisfy the lifetime reliability constraint in a long-term scale, while the lifetime constraint can be relaxed in the local time scale when the communication requirement is high. First, the applications are classified as communication-intensive applications and computation-intensive applications. The mapping schemes include two subroutines: (1) mapping for communication-intensive applications and (2) mapping for computation-intensive applications. The communication-intensive applications are mapped to near-square regions for a lower communication cost. In contrast, the computation-intensive applications are allowed to map to dispersed regions to avoid over-stressed cores. The global throughput is improved because the communication performance of communication-intensive applications is optimized while the waiting time of computation-intensive applications is reduced. Second, to satisfy the lifetime constraint, we propose a two-level controller based on runtime application mapping strategies. The proposed two-level controller includes a long-term controller and a short-term controller. The long-term controller is responsible for managing the lifetime reliability in a long-term scale while the short-term controller can manage the performance in a short time scale. Note that the *borrowing strategy* and the two-level controller are different from that in prior work [13] because in this work task mapping is used as the knob, whereas in [13] per-core DVFS is used to control the system. Different from per-DVFS which can control the lifetime reliability of cores separately, the application mapping has to select a set of cores with various aging statuses each time. Third, we propose a method to effectively classify the applications depending on the communication-to-computation ratio. To improve the throughput, a dynamic threshold is determined according to the current locations of available cores with the objective of achieving a balance between the communication time and computation time for the incoming application. Finally, to further reduce the communication cost, we propose an improved neighborhood allocation scheme, which considers the communication distance with predecessors when allocating the task graphs on the cores. A preliminary work has been presented in [19]. This paper extends the preliminary work by adding substantial analyses on the mapping scheme, a novel method to classify applications, a mapping-based two-level controller to manage the lifetime reliability and performance, and more experimental results including evaluations on realistic task graphs. The novel contributions of this paper include the following:

(1) A runtime mapping scheme exploiting a *borrowing strategy* for dynamic reliability management in many-core systems. The mapping scheme can improve the throughput because the lifetime reliability constraint can be locally relaxed when the communication requirement is high while the lifetime reliability constraint is satisfied in the long time scale.

(2) A mapping-based two-level controller that includes a long-term controller and a short-term controller to manage the lifetime reliability and performance, which are controlled by runtime application mapping.

(3) An effective method to classify the applications as communication-intensive applications and computation-intensive applications. The threshold is determined dynamically according to the current locations of available cores to achieve a balance between the communication time and computation time.

(4) An improved neighborhood allocation scheme that maps the next task considering the communication distance with all predecessors. Thus, the communication cost can be effectively reduced.

The remainder of this paper is organized as follows. Section II briefly introduces the related work. Section III discusses the motivation and design overview. Section IV presents the proposed runtime mapping scheme. Section V analyzes the experimental results, and Section VII concludes this paper.

II. Related Work

A. Dynamic Reliability Management Approaches

DRM, first proposed in [9], ensures a target lifetime reliability by adapting the processor to dynamic workloads for higher performance. DVFS has been widely used for lifetime reliability management by dynamically controlling the voltage and frequency of transistors. In [15], the researchers implement a DRM system that includes a lifetime prediction model and a control system to make a trade-off between performance and remaining lifetime reliability. The supply voltage is allowed to boost beyond the nominal voltage for critical computational demands, but the lifetime budget is also satisfied. In [13], a DRM policy is proposed for multicore platforms considering lifetime reliability and user experience. The DRM policy incorporates a two-level controller, which consists of a long-term controller for lifetime management in a long time scale and a short-term controller for satisfying performance requirements in a short time scale. They propose a *borrowing strategy*, which locally relaxes reliability-induced operating point constraints while still satisfying them over the large time windows. In [16], the researchers propose a DRM by exploiting dark silicon in many-core processors. In addition to DVFS, power gating of cores is also used as a knob to control the lifetime reliability, energy and performance.

In summary, DRM allows increasing performance in a short term scale while meeting the lifetime constraint in a long term scale. This strategy is also named the *borrowing strategy*. Most prior studies exploit the *borrowing strategy* by DVFS. However, the *borrowing strategy* is less exploited for managing lifetime reliability by runtime application mapping.

B. Runtime Mapping

Runtime mapping is very challenging since the arrival time and execution time of the applications are unknown at design time [3], [20]. The mapping algorithm should be lightweight to immediately react to the varying requests for resource allocation. Researchers have proposed various efficient heuristics to assign incoming applications to platform resources. Chou et al. [20] propose an incremental mapping

to minimize the communication energy consumption. A near-convex region is first selected as the mapping area to minimize the average Manhattan distance regardless of the application. Then, a heuristic is proposed to map the application to the selected area. In contrast to [20], the recent runtime application mapping algorithms proposed in [5], [6], [21] consist of two steps: (1) the first node is selected considering fragmentation, spatial availability and congestion, and (2) the application is mapped to a near-convex continuous region around the first node. In [21], two metrics AMD and mapped region dispersion (MRD) are defined to measure the communication cost and compactness of a mapping area. In [5], it is stated that a near-square area is preferred due to fragmentation. In [6], the researchers propose a more advanced first node selection method, which quantitatively incorporates spatial availability and dispersion into a single value. Then the neighborhood allocation method in [21] is employed to select a near-convex area to minimize dispersion and external congestion. Fattah et al. [22] propose an approach for non-continuous mapping, in which a parameter α named *allowed level of dispersion* is defined. They show that when $0 < \alpha < 1$, higher throughput can be achieved because of less waiting time. However, the *allowed level of dispersion* remains constant for all applications. All these runtime application mapping approaches are lifetime agnostic. It was shown in [18] that lifetime-agnostic approaches can aggravate lifetime reliability issues despite performance improvement and energy savings.

C. Lifetime-Aware Mapping

Lifetime-aware mapping problems in multi-/many-core systems have been extensively studied. Huang et al. [11] define an aging effect for multi-core system, and they propose a task mapping algorithm to maximize MTTF of multi-core system given performance constraints. Hartman et al. [23] propose a lifetime-aware runtime mapping heuristic for NoC-based chip multiprocessors. Their heuristic minimizes the accumulated aging effects of the cores and extends the lifetime. Sun et al. [24] propose a workload balancing scheme for multi-core systems to mitigate NBTI-induced aging issues. All these approaches address the lifetime reliability issues by optimizing the MTTF given performance constraints.

With a much shorter MTTF due to technology scaling, an increasing number of studies on many-core systems focus on optimizing the performance given a lifetime reliability constraint. Huang et al. [17] propose a lifetime-constrained mapping algorithm to minimize the energy consumption under a lifetime constraint on multi-core system. The optimization problem is solved by simulated annealing at design time. Liu et al. [25] propose a design-time mapping and scheduling approach for multi-core systems given lifetime reliability and thermal constraints. Incorporated with DVFS, the mapping and scheduling problem is formulated as a mixed integer linear programming (MILP) problem, which minimizes the total energy consumption with both thermal threshold constraint and system lifetime reliability constraint satisfied. Gnad et al. [26] propose a runtime lifetime management system that harnesses dark silicon to decelerate aging and improves the overall system performance for a given MTTF. However, their method targets malleable applications that have varying degrees of parallelism [27], i.e., the applications can expand or shrink the mapping regions at runtime according to the performance requirements. The runtime mapping in [18] extends the method in [6] to take the aging information into consideration when selecting the first node. Based on the first node selection method, a runtime mapping given a lifetime constraint is proposed. However, the method does not exploit the *borrowing strategy* feature of DRM. In other words, the fact that the lifetime constraint can be temporally locally relaxed for higher performance requirements is neglected.

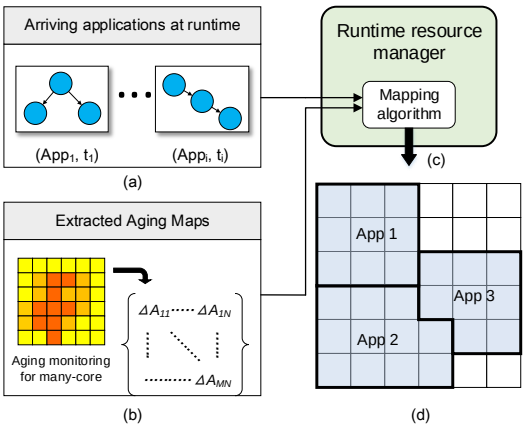


Fig. 1. Application mapping overview. (a) Applications are represented as task graphs. (b) The aging status of the cores is monitored and stored in an aging map. (c) The resource manager is responsible for the application mapping. (d) The applications are mapped on the many-core platform.

III. Motivation and Design Overview

A. Design Overview

Fig. 1 presents an overview of the runtime mapping design. This paper considers multiple applications coexisting on the many-core system. The arrival time and leaving time of the applications are unknown in advance. At runtime, when there is an incoming application, the application mapping scheme needs to determine a set of cores for the application considering the lifetime reliability of the cores, performance requirements of the application, internal and external congestion of the mapping region, and so forth. The application mapping and resource management are controlled by a centralized resource manager. The resource manager takes the application information, platform aging map and current resource status as the inputs of the mapping algorithm. The output of the algorithm specifies how the tasks of the incoming applications are mapped to the many-core system. The models of lifetime reliability, applications and many-core platform are presented in detail in the following section.

B. Models

1) Lifetime Reliability: The aging mechanisms for lifetime reliability include electro-migration (EM), time-dependent dielectric breakdown (TDDB), stress migration (SM), negative bias temperature instability (NBTI), hot carrier injection

(HCI), and so forth. In this paper, we only consider the EM-induced aging mechanism. Other aging mechanisms can easily be incorporated using the Sum-of-Failure-Rate (SOFR) model [17].

The lifetime reliability can be modeled according to Weibull distribution [11]. The MTTF is given as

$$MTTF = \alpha\Gamma(1 + \frac{1}{\beta}) \tag{1}$$

where α is related to operating conditions such as temperature, switching activity, frequency, etc. Γ is the gamma function. β is the slope parameter for the Weibull distribution. The scale parameter is

$$\alpha = \frac{M_0(J - J_{crit})^{-n} e^{\frac{E_a}{kT}}}{\Gamma(1 + \frac{1}{\beta})} \tag{2}$$

where M_0 is a material-related constant; J is the current density which is related to voltage, frequency and switching activity; J_{crit} is the critical current density; and T is the operating temperature.

Considering the runtime variation in the operating conditions, an aging effect for a processor is also defined in [11], as shown in Equation 3.

$$A = \sum_i \frac{\Delta t_i}{\alpha_i} \tag{3}$$

where Δt_i is the duration of the i-th time interval. α_i is the scale parameter in the i-th time interval. The aging effect represents aggregate operating conditions including temperature, voltage, frequency and utilization.

To satisfy the predefined lifetime constraint, the aging effect should be less than the target aging effect. We define a lifetime budget in Equation 4.

$$\Delta A(t) = A^{target}(t) - A(t) > 0 \tag{4}$$

A positive lifetime budget value means that the lifetime constraint is satisfied and vice versa. For lifetime reliability management, the lifetime can be regarded as a resource consumed over time. $A_{target}(t)$ indicates the given resource, and $A(t)$ indicates the consumed resource. As soon as the consumed resource is less than the given resource, the lifetime constraint is satisfied. Equation 3 and Equation 4 also indicate that if a healthy core is over-stressed for a period, then the lifetime budget decreases and even the lifetime constraint is violated. Conversely, if a high-aged core ($\Delta A < 0$) is less stressed for a period, then it can recover from its high-aged status.

2) Application Model: Each application can be modeled as a directed task graph $AG = (T, E)$, where T is the set of tasks and E is the set of directed edges. Each task $t_i \in T$ has a worst-case execution time when mapped to a core. Each edge $e_{i,j} \in E$ represents data dependency between the source task t_i and destination task t_j. The weight $w_{i,j}$ represents the communication volume from task t_i to task t_j. The execution time of an application or the makespan of a task graph is determined by both task execution time and communication time. Fig. 1(a) shows a set of applications. At runtime, the applications arrive and leave the system dynamically. The arrival time and execution time are unknown in advance. Therefore, the information of an application and the resource status are available when the application arrives. This requires the design of a low-complexity and highly efficient runtime mapping algorithm.

3) Platform Description: The many-core system consists of a set of homogeneous cores that are connected by a 2D-mesh on-chip interconnect network. The platform is represented as a directed graph $G(P, L)$, where P represents the set of cores n_i and $l_{i,j} \in L$ represents the physical channel between two cores n_i and n_j. Fig. 1(d) shows the many cores that are interconnected by the on-chip network.

Each core is associated with a lifetime budget ΔA. It is assumed that a software-based aging monitoring framework is adopted. The framework estimates the aging status of each core periodically according to the operating conditions. The resource management manager maintains an aging map, which contains the ΔA values of all cores. If the size of the many-core system is $M \times N$, then the aging map can be represented by a matrix in Equation 5.

$$\boldsymbol{AM} = \begin{Bmatrix} \Delta A_{1,1} & \cdots & \Delta A_{1,N} \\ \vdots & \ddots & \vdots \\ \Delta A_{M,1} & \cdots & \Delta A_{M,N} \end{Bmatrix} \tag{5}$$

As demonstrated in Fig. 1(b), an aging map is used to keep the lifetime budgets ΔA of the cores in the many-core system.

C. Motivation Example

We observe that state-of-the-art lifetime-constrained mapping [18] can possibly lead to low throughput of many-core systems if all applications are required to map to near-square regions. Fig. 2 presents an example of an application with 4 tasks that are mapped to a 3×3 many-core system[1], in which the core in the central region is a high-aged core and the others are less-aged cores. It is assumed that the high-aged core has violated the predefined lifetime constraint. If the application is mapped to a near-square region as in Fig. 2(a), then the application has to wait until the core recovers from the high-aged status. This situation would possibly lead to low throughput due to a long application waiting time. However, the applications can have various communication characteristics, i.e., some applications are more computation-intensive than others. For those applications, it is not necessary to map to a near-square region because the communication distance is less important. As shown in Fig. 2(b), the application is mapped to a non-square region without occupying the high-aged core. Despite a slight increase in the average communication distance, the system throughput can be improved because of a shorter waiting time. Therefore, Fig. 2 indicates that higher throughput can be achieved by mapping the applications with different strategies. In Section IV, a detailed runtime mapping scheme that includes two different mapping strategies is presented to improve the throughput.

[1]It is assumed that the maximum number of tasks running on each core is one.

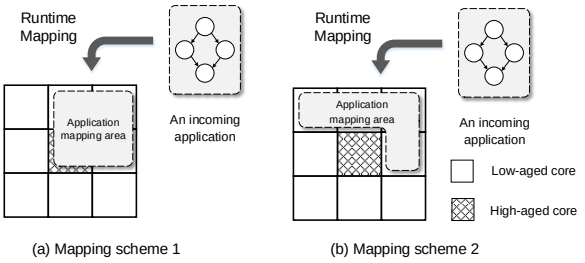


Fig. 2. Motivation example. (a) The application is mapped to a square region for low communication cost, but the aging effect of the high-aged core is aggravated. (b) The application is mapped to a non-square region for a low waiting time, and the lifetime constraint of the high-aged core is satisfied.

D. Problem Statement

At runtime, N applications arrive. To improve the throughput, the resource manager should allocate resources to the N applications such that all N applications can finish as early as possible. The constraint is the lifetime reliability constraint. The decision variables are (1) choosing if mapping or waiting for the next application and (2) task-to-core mapping of each application.

$$\text{maximize} \quad \text{Throughput}(t_s, t_f) = \frac{N}{t_f - t_s} \tag{6}$$

subject to

$$A_j^{target}(t_f) - A_j(t_f) \geq 0, \quad \forall n_j \in P \tag{7}$$

$$A_j(t_f) = A_j(t_s) + \int_{t_s}^{t_f} \frac{1}{\alpha_j(t)} dt, \quad \forall n_j \in P \tag{8}$$

$$A_j^{target} = \int_0^{t_f} \frac{1}{\alpha_j^{nominal}} dt \tag{9}$$

where t_s is the arrival time of the first application and t_f is the completion time of all N applications. At time t_f, the lifetime reliability constraint should be met. In this paper, we use the aging effect as the lifetime constraint.

Equation 6 defines the objective as maximizing the throughput. Since N is constant, the objective is to minimize t_f such that at time t_f all applications have finished and the lifetime budgets of all cores are positive (Equation 7). The total time includes waiting time and running time. The running time of each application consists of the execution time of the tasks and the communication time, which is related to the communication distance of the mapped tasks. Therefore, the mapping decisions should attempt to minimize the waiting time of the applications and the communication distance of the mapping region. Equation 8 and Equation 9 present the calculation of the aging effect at time t_f and the target aging effect according to Equation 3. The aging effect at time t_f is associated with $\alpha(t)$, which integrates the operating conditions between t_s and t_f. If a core is allocated a task, then the core will experience higher workload and temperature, causing a higher aging effect during this period. The target aging effect is associated with the accumulated nominal operating condition which is constant and represented as $\alpha_{nominal}$. This comes from the user-defined lifetime constraint.

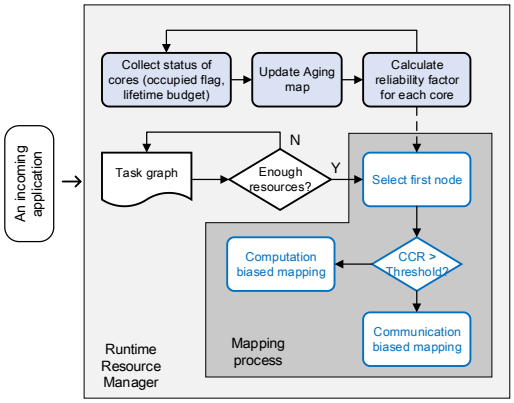


Fig. 3. Flow of the proposed application mapping scheme.

IV. Proposed Approach for Lifetime Constraint Mapping

In this section, we propose a lifetime-constrained mapping scheme. Fig. 3 presents the flow for the application mapping scheme. First, the centralized manager periodically updates the aging map of the many-core system. When a new application arrives, the mapping algorithm is called to map the application onto the many-core system. The mapping algorithm includes two steps: (1) first node selection and (2) mapping other tasks around the first node. To satisfy the lifetime constraint, an application mapping-based two-level controller is presented to control if the incoming application is mapped or placed in the waiting queue. Finally, we present a method to determine the threshold to efficiently classify the applications for higher throughput. To incorporate the *borrowing strategy* in the application mapping, two challenges need to be addressed: (1) how to maintain the target lifetime constraint in a long-term scale and (2) when and how to relax the lifetime constraint in a short-term scale. These two challenges are effectively addressed in Section IV-C and Section IV-D.

A. Runtime Aging Monitoring and Aging Map Update

For runtime lifetime management, a runtime aging monitoring framework is required to make decisions based on the runtime aging status. In this paper, we adopt a similar method as [28], in which the aging effect is estimated based on aggregate operating conditions, including the temperature, switching activity, voltage, frequency, and so forth.

Given a target aging effect, the lifetime budget ΔA is calculated for each core. If $\Delta A > 0$, then it indicates that the aging effect is less than the target aging effect, i.e. the lifetime constraint of the core is satisfied, and vice versa. The aging map is updated periodically.

B. First Node Selection

In this paper, we adopt the state-of-the-art approach [6], [18] for first node selection. The reason is that the first node selection in the mapping framework considers external congestion and fragmentation issues, which are two important factors that affect the mapping performance. Thus external

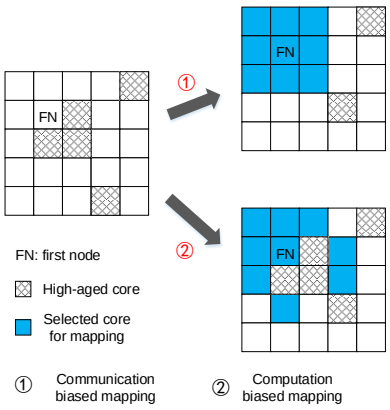


Fig. 4. Diagram for neighborhood node allocation, which includes two subroutines: (1) communication-biased mapping for communication-intensive applications and (2) computation-biased mapping for computation-intensive applications.

congestion and fragmentation are beyond the scope of this paper. Moreover, the first node selection in [18] can also balance the aging effect of cores. In this paper, we use the lifetime budget ΔA as the lifetime reliability metric in the first node selection to balance the aging effect. For each core (m, n) in the many-core system, a reliability factor $RF_{m,n}$ is evaluated based on the status of the neighborhood cores. The reliability factor is represented as follows,

$$RF_{m,n} = \sum_{i=m-r}^{m+r} \sum_{j=n-r}^{n+r} W_{i,j} \times \Delta A_{i,j} \times (r - d + 1) \quad (10)$$

where $W_{i,j} = 0$ if (i, j) is occupied and $W_{i,j} = 1$ if core (i, j) is unoccupied. r is the radius of the square and d is the distance from the center. $RF_{m,n}$ incorporates lifetime reliability metric, status of the cores and the range of the square into a single value. Additionally, the closer core around (m, n) is given a higher weight. A higher $RF_{m,n}$ indicates that there are more free cores and more lifetime budgets around the core (m, n). The first node is the core with the highest reliability factor. Note that the reliability factor values of all cores are calculated at runtime proactively. Thus, the calculation of the reliability factor does not affect the running time of the mapping algorithm.

C. Neighborhood Node Allocation

After the first node is selected, the tasks of the application are allocated to the first node and the neighborhood cores around the first node. Because the applications have various characteristics, it poses different requirements of communication performance on the resource allocation. In this paper, we classify applications as communication-intensive applications and computation-intensive applications. Because communication is more dominant in communication-intensive applications, the communication-intensive applications are mapped to more compact regions. Meanwhile, the computation-intensive applications are mapped to more dispersed regions.

In this paper, the applications are classified according to their communication-to-computation ratio (CCR), which is defined as the average communication time to be sent in one application divided by the average computation time. Therefore, a higher CCR value indicates that the application performance is more dominated by the communication, whereas a lower CCR value indicates that the application performance is more dominated by the computation. The method to classify the applications is detailed in Section IV-E.

Algorithm 1 Pseudocode of Lifetime-aware Neighborhood Allocation (LNA)

Input n_f: First node;
$\Delta A_{i,j}, i \in [1, N], j \in [1, M]$: Aging map;
$AG(T, E)$: the application to be mapped.
Output Mapping function $T \rightarrow P$
Definition Ω: the tasks that are not mapped;

1: $\Omega \leftarrow T$ ▷ Tasks that are not mapped in $AG(T, E)$.
2: $r \leftarrow 1$ ▷ Radius of square around the first node
3: Take the first task t_f from Ω in breadth-first order of AG
4: Map t_f on n_f
5: $S_r \leftarrow$ The set of unoccupied cores that are within current square, which is with radius r around n_f
6: **while** $\Omega \neq \varnothing$ **do**
7: Take the next task t_n in Ω in breadth-first order
8: **if** $S_r = \varnothing$ **then**
9: $r \leftarrow r + 1$
10: S_r is updated
11: **end if**
12: **if** communication-biased mapping **then**:
13: $n_m \leftarrow$ the core within S_r and with the minimum number of weighted hops with all predecessors of task t_n
14: **else if** computation-biased mapping **then**:
15: $n_m \leftarrow$ the core with $\Delta A_m > 0$ and within S_r and with the minimum number of weighted hops with all predecessors of task t_n
16: **end if**
17: Map t_n on n_m
18: **end while**

In the mapping process, the *borrowing strategy* is adopted to relax the lifetime constraint in a short-term scale when there is a high communication requirement from the application. Specifically, the mapping algorithm includes two subroutines as shown in Fig. 4, communication-biased mapping and computation-biased mapping. If the incoming application is a communication-intensive application, then the communication-biased mapping is called to map the application to a near-square area with the lifetime constraint locally relaxed. This is because the communication performance requirement in the local time window is higher, and the performance is the first priority to optimize.

1) Communication-biased Mapping: If the application is a communication-intensive application, then it has a higher requirement on the communication performance. Hence, the communication distance is more important. The widely adopted method for neighborhood allocation is CONA [21]. With the objective of selecting a near-square region to mitigate the internal and external congestion [5], CONA selects the node that fits the smallest square when mapping the next task.

However, the communication distance with predecessors of the current task is not considered, which possibly leads to a high communication cost.

In this paper, we propose an improved method for neighborhood node allocation named lifetime-aware neighborhood allocation (LNA), which takes both the task communication overhead and aging information into consideration. The algorithm for the method is presented in Algorithm 1. Given the first node of an application, the next task to map is taken from AG_i in breadth-first order (line 3 and line 7). The first task is mapped on the first node (line 4). S_r is expanded if there is no available core in the current square (lines 8-11). If the application is communication intensive, then the next core is selected as the core with minimum weighted hops with all predecessors of t_n (lines 12-13). Finally, the task t_n is mapped on n_m (line 17). Compared to CONA, the major improvement is that the proposed method maps the next task to the core that is with the minimum weighted communication distance with predecessors of the task. The weighted hops with all predecessors of task t_n can be represented as $\sum w_{i,n} MD(map(t_i), map(t_n)), \forall e_{i,n} \in E$, where $MD(map(t_i), map(t_n))$ denotes the Manhattan distance between $map(t_i)$ and $map(t_n)$ in the many-core system.

Assuming the maximum number of predecessors in the task graph is D, the complexity of the algorithm is $O(|T||P|D)$, where $|P|$ is the number of cores and $|T|$ is the number of tasks. For comparison, the complexity of CONA is $O(|T||P|)$, which is slightly lower than that of LNA because LNA considers the weighted communication distance with predecessors. The computation overhead is further analyzed in Section V-F.

2) Computation-biased Mapping: Compared to a communication-intensive application, the performance of a computation-intensive application is less affected by the communication distance. This creates opportunities to map the application to a more dispersed area to avoid high-aged cores. The over-stressed cores can also recover from high-aged status. We propose a method that combines the high-aged cores ($\Delta A < 0$) with other allocated cores to form a compact shape. The mapping algorithm for computation-intensive application is also presented in Algorithm 1 (lines 14-16). The high-aged cores are avoided when mapping the application, but the communication distance is still considered. Compared to mapping for communication-intensive applications, the communication distance increases because more cores are avoided and the mapping area is more dispersed.

D. Lifetime Constraint Satisfaction

In addition to short-term relaxation of the lifetime constraint, the runtime mapping that adopts the *borrowing strategy* should also satisfy the lifetime constraint of the many-core system in a long-term scale. The implementation of the *borrowing strategy* is an application mapping-based two-level controller that controls the lifetime reliability and performance by making mapping decisions and choosing to queue or map the incoming application. If the lifetime reliability constraint is not satisfied, then the incoming application is placed in the waiting queue until there is enough lifetime budget. Therefore, similar to [13], the two-level controller includes a long-term controller and a short-term controller. The long-term controller maintains the lifetime budget, while the short term controller attempts to optimize the performance. In contrast to the method in [13] which controls the lifetime reliability of cores via per-core DVFS, the proposed two-level controller is based on application mapping that selects a set of cores each time for an incoming application.

Algorithm 2 Lifetime Constraint Satisfaction

Input AG_i: the arrival application;
$\boldsymbol{AM}$: ΔA of all cores;
n_f: First node.
Output Choose Queue() or Map().
1: $r \leftarrow (\sqrt{(AppSize)} - 1)/2$ ▷ Radius around the first node;
2: $S_r \leftarrow$ The square region with radius r around n_f
3: $\Delta A(S_r) = \sum_i \Delta A_i, i \in S_r$
4: $C \leftarrow$ number of free cores
5: $|T| \leftarrow$ the number of tasks in AG_i
6: **if** $C < |T|$ **then**
7: **Queue**(AG_i)
8: **else if** $\Delta A(S_r) < 0$ **then**
9: **Queue**(AG_i)
10: **else**
11: **Map**(AG_i)
12: **end if**

1) Long-term Controller: The long-term controller groups the applications in batches. Each batch contains at most N applications. When a new application arrives, the application is placed in the current batch if the number of applications in the current batch has not reached N. Otherwise, it is placed in the next batch. In the mapping process, only when the minimum lifetime budget of the system becomes positive after mapping all applications in the previous batch, can the application of next batch be mapped.

2) Short-term Controller: Within each batch, the short-term controller decides when and how to map the next application. An algorithm is presented in 2. In the algorithm, if there are not enough free cores for the incoming application, then the application is placed in the waiting queue until a running application leaves (line 6). The total amount of lifetime budget of the cores around the first node is also calculated and denoted as $\Delta A(S_r)$(lines 1-3). If $\Delta A(S_r)$ is positive, then it indicates that the communication-intensive applications can be mapped around the first node even though some cores are over-stressed. In other words, the lifetime constraint is relaxed in short time scale. Therefore, when $\Delta A(S_r) > 0$ and there are enough free cores, the application can be mapped (line 11). If the incoming application is computation intensive, then it is mapped around the first without occupying the over-stressed core.

In this paper, the thermal and power constraints, such as thermal design power (TDP) are not considered for two reasons. (1) TDP-constrained optimization problems in many-core systems have been extensively studied in prior research papers [29]–[31]. TDP-constrained design can easily be incor-

porated in the proposed two-level controller. (2) The lifetime reliability is closely associated with power consumption since high power leads to high temperature and increasing temperature can exponentially aggravate aging effect.

E. Threshold for Application Classification

As presented in the previous section, the runtime mapping scheme includes different strategies for communication-intensive applications and computation-intensive applications. Determining the optimal threshold to classify the applications still remains unsolved. A suboptimal threshold may lead to the following two scenarios:

(1) If the threshold is too low, then most applications would be classified as communication-intensive applications which are mapped in compact regions. This situation possibly leads to long waiting time, such as the example shown in Fig. 2;

(2) If the threshold is too high, then most applications are classified as computation-intensive applications, which are mapped in dispersed regions. A possible scenario is that some applications with a high CCR are mapped to dispersed regions, leading to a long communication time and high communication overhead. The high communication time also results from low throughput.

Ultimately, an inappropriate threshold can possibly lead to low throughput of the system. It is essential to determine an appropriate threshold to effectively improve the throughput of the system.

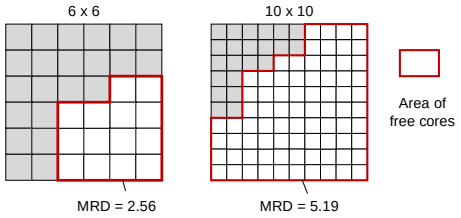
In this paper, the applications are classified as communication-intensive applications and computation-intensive applications according to their CCR. CCR is defined as $comm/comp$, which indicates the average communication time divided by the average computation time in one application. To avoid the aforementioned 2 scenarios, the threshold should make a balance between the communication time and computation time. Since the threshold should be defined before the mapping process starts, the mapping area and the communication time are unknown. Therefore, it is assumed that all available cores are possibly mapped. The average distance of all available cores is calculated and represented as MRD_{free} in Equation 11

$$MRD_{free} = \frac{\sum_{n_i,n_j \in D} MD(n_i, n_j)}{\binom{|D|}{2}} \tag{11}$$

where $|D|$ denotes the number of all remaining available cores. $MD(n_i, n_j)$ denotes the Manhattan distance between cores n_i and n_j. The threshold is defined in Equation 12.

$$Threshold = \frac{1}{MRD_{free}} \tag{12}$$

The $Threshold$ value indicates that if the CCR of an incoming application is equal to the critical value, it is expected that the communication time and computation time can be balanced if the application is randomly mapped to all remaining nodes of the many-core system. It is known that the actual communication time is determined by both the communication volume and communication distance of an application. If the CCR of an incoming application is higher than the $Threshold$ value, then the application is classified as communication intensive and can only be allowed to map to a compact region with the communication distance minimized. Otherwise, the application is classified as computation intensive and allowed to map to a dispersed region.



(a) 6×6 many-core system (b) 10×10 many-core system

Fig. 5. Average distance of remaining nodes. MRD is the average distance. Grayed blocks represent unavailable cores. The MRD value of the remaining cores increases with the size of the many-core system.

The rationale behind the $Threshold$ value is that the average distance of the tasks is approximately equal to MRD_{free} when the tasks are randomly mapped. In Equation 12, MRD_{free} depends on the locations of the remaining available cores. Fig. 5 presents a comparison in 6×6 and 8×8 topologies. The comparison shows that with the same number of occupied cores, MRD_{free} is much higher in a larger topology. In other words, the computation-intensive application is more likely to be mapped to a highly dispersed region in a large topology. Hence, the threshold should be lower in a larger topology such that more applications are classified as communication-intensive applications and mapped to compact regions to avoid a high communication distance.

F. Thermal Considerations

It is known that the lifetime reliability is closely related to temperature hotspots. It is stated in [32], [33] that if the tasks are mapped onto a compact region, then the communication cost can be minimized but a thermal hotspot possibly occurs. However, if the tasks are mapped in a dispersed region, then the peak temperature is lower but with a higher communication distance. Therefore, the proposed computation-biased mapping not only mitigates the aging issues by freeing high-aged cores, but also achieves a more balanced temperature distribution that indirectly mitigates the lifetime reliability issues. Since it requires complicated thermal models to calculate the predicted temperature for all possible mapping decisions, the consideration of thermal hotspots would greatly increase the complexity of the runtime mapping algorithm. Therefore, our proposed application mapping algorithm does not consider the thermal distribution for simplicity. However, this will not significantly affect the effectiveness of the proposed approach, because our approach still considers the impact of temperature on the lifetime reliability and the thermal hotspots would negatively aggravate the aging effects, while the aging issues are directly mitigated by the proposed application mapping algorithm. The runtime application mapping algorithms that consider both thermal hotspots and lifetime reliability are left as a possible future work.

TABLE I
CONFIGURATIONS OF SYNTHETIC APPLICATIONS AND PLATFORM.

Synthetic graph parameters	
Number of tasks	[10, 20], [20, 30]
Task execution time	[100, 400] (cycles)
Power of task	[50, 100] (mW)
CCR	[0, 0.5], [0, 2]
Network parameters	
Size & Topology	$8 \times 8, 12 \times 12$ 2D mesh
Routing algorithm	XY routing
Latency	2 cycles per hop

TABLE II
CONFIGURATIONS FOR EVALUATION.

8×8 many-core	A1	A2	A3	A4
Application size	[10, 20]		[30, 40]	
CCR	[0, 2]	[0, 0.5]	[0, 2]	[0, 0.5]

V. EXPERIMENTAL RESULTS

A. Experimental Setup

In this paper, the experiments are performed in an in-house many-core simulator. McPat [34] is integrated to model the power consumption. Hotspot [35] is used to model the temperature, assuming that thermal sensors are integrated to monitor the temperature of cores. CALIPER [36] is used to calculate the aging rate. The specifications of cores are adopted from Niagara2 processors in McPat [34]. Both synthetic and realistic task graphs are adopted to evaluate the proposed mapping schemes in the experiments. The synthetic task graphs are generated by DAGGEN [37] with the configurations shown in Table I. Other configurations to generate the synthetic graphs are adopted from the default parameters in DAGGEN. Moreover, we also evaluate the mapping scheme using realistic task graphs that are extracted from video processing applications [38]. The proposed lifetime-constrained mapping schemes, which incorporate CONA and LNA neighborhood allocation methods, are named *LBRM-CONA* and *LBRM-LNA*, respectively. The mapping scheme is also compared with the state-of-the-art lifetime-constrained scheme named *DSRM* [18]. The lifetime-agnostic runtime mapping named *MAPPRO* [6] is also compared.

B. Evaluations on Synthetic Applications

The proposed mapping algorithm is evaluated in terms of throughput and communication cost under various lifetime constraints. There are a total of 10^6 applications generated in the simulation, and the arrival rate of applications is 0.001 applications per cycle. Since the proposed two-level controller groups the applications into batches, each batch contains 10^3 applications in the evaluation.

1) Throughput Comparisons: We adopt aging effect as the lifetime reliability constraint. Since aging effect reflects the accumulated aging status with time (shown in Equation 3), we set the lifetime constraint from $1M$ to $0.4M$, where M is a reference aging effect value without any lifetime constraints. In other words, the initial lifetime budget is set from $1M$ to $0.4M$, among which $0.4M$ is the tightest constraint imposed on the many-core system.

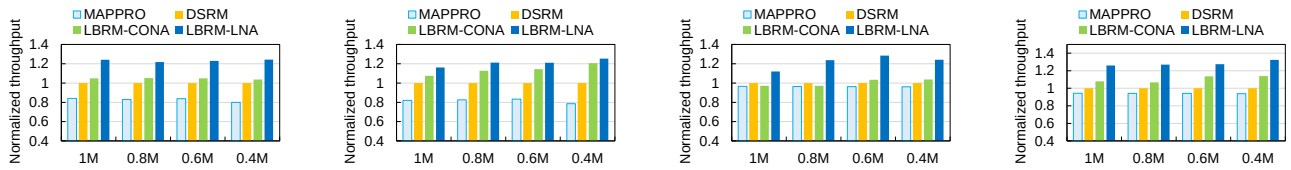
Fig. 6 and Fig. 7 show comparisons of normalized throughput in 8×8 and 12×12 many-core systems respectively. A1-A4 and B1-B4 correspond to configurations with various application sizes and CCR values, as shown in Table II. The throughput is normalized to *DSRM* for comparison. In Fig. 6, when the initial lifetime budget is $1M$, the average throughput improvement of *LBRM* over *DSRM* is 19.5%. The improvement comes from the proposed new neighborhood allocation scheme *LNA*. When the initial lifetime budget is $0.4M$, the average throughput improvement of *LBRM-LNA* over *DSRM* reaches 26.5% because of both the *borrowing strategy* and *LNA*. By using the *borrowing strategy*, *LBRM* maps computation-intensive applications in dispersed regions without occupying high-aged cores. The throughput is improved due to less waiting time of computation-intensive applications. *LBRM-CONA*, which combines the *borrowing strategy* with *CONA*, shows at most 4.5% and 0.5% throughput improvements for A1 and A3, respectively, and at most 13.7% and 10.5% throughput improvements for A2 and A4, respectively. The reason for the higher improvements for A2 and A4 is that more applications are classified as computation-intensive applications when the CCR range is $[0, 0.5]$. Fig. 7 presents similar results in a 12×12 topology size. When the lifetime constraint is $0.4M$, the average throughput improvement of *LBRM-LNA* over *DSRM* reaches 20.6%, which is less than that of the 8×8 topology size. This result is because the relative size of the applications is smaller when in a larger topology, thus creating less opportunities to avoid high-aged cores. Similar to the 8×8 topology, *LBRM-CONA* has no significant throughput improvement when the CCR value is $[0, 2]$ (A1 and A3), but it has at most 9.6% and 12.5% throughput improvements when the CCR value is $[0, 0.5]$ (A2 and A4).

2) Comparisons of Communication Cost: Average weighted Manhattan distance (AWMD) is a widely used metric to evaluate the communication cost. As defined in [21], AWMD is the sum product of communication distance and all edges' weights of the mapped application, averaged by the total communication weights. To evaluation the communication cost of all mapped applications, AWMD as defined in this paper considers all mapped applications rather than a single application. AWMD is defined as follows,

$$\text{AWMD} = \frac{\sum_n \sum_{e_{i,j} \in E_n} w_{i,j} \times MD(e_{i,j})}{\sum_n \sum_{e_{i,j} \in E_n} w_{i,j}} \tag{13}$$

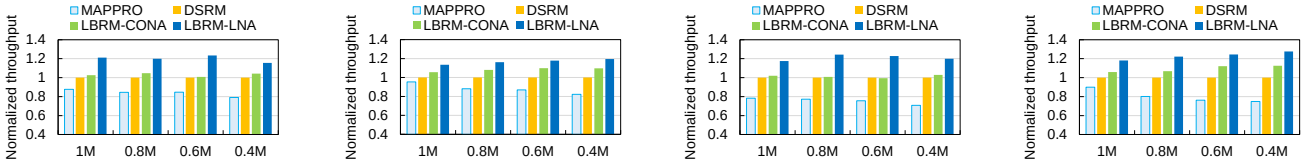
where n is the total number of mapped applications, E_n is the set of edges of the n-th application, $w_{i,j}$ is the weight of the edge $e_{i,j}$ and $MD(e_{i,j})$ is the Manhattan distance of tasks t_i and t_j. A lower AWMD value indicates lower communication cost in terms of communication time and communication energy.

Table III shows partial comparisons of the AWMD values with respect to Fig. 6 and Fig. 7. Due to space limitations, only the AWMD values of $1M$ and $0.6M$ are shown. It shows that on average *LBRM-CONA* leads to a slight increase in AWMD compared to *DSRM*. This is because for *LBRM-CONA*, the computation-intensive applications are allowed to map to dispersed regions, which increases the communication



(a) A1 (b) A2 (c) A3 (d) A4

Fig. 6. Many-core platform size: 8×8. (a)-(d) are the results under various application sizes and CCR values. The results are normalized throughput under various lifetime constraints. $1M$, $0.8M$, $0.6M$, and $0.4M$ denote various lifetime constraints. M is a reference aging effect value of the system for *LBRM-LNA* when there is no lifetime constraint. The configurations of A1-A4 are presented in Table II.



(a) A1 (b) A2 (c) A3 (d) A4

Fig. 7. Many-core platform size: 12×12. (a)-(d) are the results under various application sizes and CCR values. The results are normalized throughput under various lifetime constraints.

distance. However, because only low-communication applications are mapped to dispersed regions, the AWMD is less affected. It can also be concluded that *LBRM-CONA* can effectively improve the throughput without a significant impact on the communication cost. *LBRM-LNA* has a considerably smaller AWMD value because an improved neighborhood allocation scheme is adopted, which significantly reduces the communication distance.

C. Throughput Improvement vs. CCR range

In this section, we characterize the throughput improvement over *DSRM* given different CCR ranges. The CCR range is between 0 and the maximum CCR. The maximum CCR is chosen from 0.05 to 5. Fig. 8 presents the results for different topology sizes and application sizes.

When the CCR is low, it can be observed that *LBRM-CONA* can effectively improve the throughput for all configurations. This is because when the CCR is low, nearly all applications are classified as computation-intensive applications which can only be mapped to a compact area for *DSRM*. However, *LBRM-CONA* relaxes the requirement of compact area and reduces the waiting time of applications. Hence, the throughput is improved. The improvement is even greater than 50% when the topology size is 8×8 and the application size is $[30, 40]$. The reason is that the application sizes of most applications even exceed half of the topology size, making it more difficult to find a compact area for *DSRM*. The waiting time can be greatly reduced if they are mapped to dispersed regions. When the CCR is high, the improvement of *LBRM-CONA* is much less obvious because most applications are with high CCR values and classified as communication-intensive applications. Therefore, *LBRM-CONA* maps more applications to a compact area as *DSRM*, showing less advantages over *DSRM*.

LBRM-CONA improves the throughput mainly due to the *borrowing strategy*. In addition to the *borrowing strategy*, the improvement of *LBRM-LNA* also comes from the improved neighborhood allocation scheme *LNA*. When the CCR is high, it shows that *LNA* provides an additional 20%-40% throughput improvement. However, when the CCR is low, *LBRM-LNA* is close to *LBRM-CONA*. This is because *LNA* mainly improves the task allocation by considering the communication with all predecessors. If there is less communication, *LNA* becomes closer to *LBRM-CONA*. Fig. 8(b) also shows a quite different curve from the others because the throughput improvement of *LBRM* reaches more than 50% at low CCR ranges and relatively large application sizes.

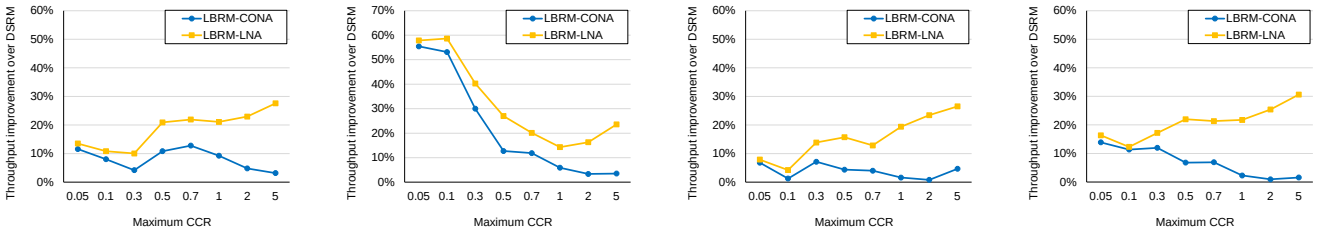
D. Advantages of Proposed Classification Method

To show the advantages of the proposed classification method, we compare it with a naive method that determines the threshold according to the average CCR of all applications. For example, the threshold for the CCR range $[0, 0.5]$ is 0.25. In contrast, the proposed method determines the threshold dynamically, which depends on the current locations of all available cores and topology size.

As shown in Fig. 9, the proposed *LBRM-LNA* mapping schemes that incorporate the proposed classification method and the naive method are named *MRD* and *AVG*, respectively. Additionally, *CP* and *CM* indicate that all applications are classified into computation-intensive applications (*CP*) and communication-intensive applications (*CM*), respectively. In Fig. 9, A1-A4 correspond to the configurations in Table II, and the lifetime constraint is $0.6M$. It shows that when the CCR is $[0, 0.5]$, *MRD* has more than 10% throughput improvement than *AVG*. However, when the CCR is $[0, 2]$, the improvement is not significant. It can be concluded that the advantages of *MRD* depend on the CCR range. The threshold determined by the proposed method is more appropriate when the CCR is $[0, 0.5]$. Overall, *MRD* outperforms *AVG* in most cases. Moreover, it also shows that when there is no classification, the throughputs of *CP* and *CM* are much lower than *MRD*, except

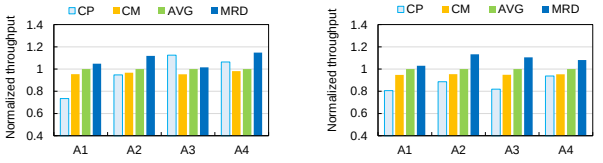
TABLE III
AVERAGE WEIGHTED MANHATTAN DISTANCE (AWMD).

Many-core size	8×8								12×12							
Configurations	A1		A2		A3		A4		A1		A2		A3		A4	
Constraints	1M	0.6M	1M	0.6M	1M	0.6M	1M	0.6M	1M	0.6M	1M	0.6M	1M	0.6M	1M	0.6M
MAPPRO	2.98	2.89	2.97	2.88	4.35	4.35	4.35	4.34	3.05	2.90	3.01	2.92	4.48	4.43	4.49	4.42
DSRM	2.95	2.95	2.93	2.92	4.47	4.45	4.46	4.44	3.07	3.00	3.03	3.04	4.47	4.47	4.48	4.45
LBRM-CONA	2.91	2.91	3.08	3.00	4.42	4.12	4.42	4.10	2.92	2.91	3.06	2.98	4.44	4.34	4.42	4.28
LBRM-LNA	1.96	1.94	2.24	2.19	2.74	2.74	2.78	2.79	1.99	1.93	2.16	2.08	2.86	2.86	2.95	2.87



(a) 8×8, app. size: [10, 20] (b) 8×8, app. size: [30, 40] (c) 12×12, app. size: [10, 20] (d) 12×12, app. size: [30, 40]

Fig. 8. Throughput improvement over *DSRM* vs. CCR range for various topology sizes and application sizes. The CCR range is between 0 and maximum CCR. The CCR values of the applications are uniformly distributed within the given range.



(a) Many-core size: 8×8 (b) Many-core size: 12×12

Fig. 9. Comparisons of the classification methods.

for the case $A3$ in Fig. 9(a), in which the relative size of an application is larger while *CP* allows the applications to map in dispersed regions without a long waiting time. In other words, the throughput is lower when all applications are classified as either computation-intensive applications or communication-intensive applications. This also proves the effectiveness of the classification method.

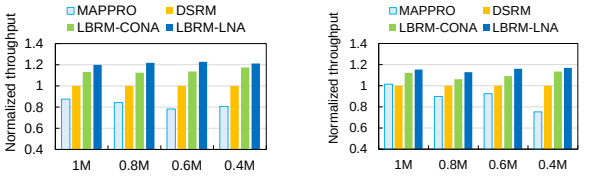
E. Evaluations on Realistic Task Graphs

In this paper, the proposed mapping scheme is also evaluated using realistic task graphs, including video object plane decoder (VOPD), picture-in-picture application (PIP), and multi window display application (MWD), triple video object plane decoder (TVOPD), MPEG4, MP3 encoder, H.263 encoder and H.263 decoder [38], [39]. Since only the communication profiles of these applications are provided, we assume that the computation time of the tasks is identical and that the CCR value of VOPD is 1. The CCR values of the other applications are normalized accordingly, as shown in Table IV. In the simulation, there are a total of 10^6 applications in which the 8 task graphs are uniformly distributed.

1) Throughput Comparisons: Fig. 10 shows the comparisons of throughput for realistic task graphs. This figure shows that in the 8×8 topology size, *LBRM-CONA* and *LBRM-LNA* have at most 17.4% and 22.7% throughput improvements respectively. When the topology size is 12×12, *LBRM-CONA* and *LBRM-LNA* have at most 13.4% and 16.9% throughput improvements, respectively. *LBRM-LNA* has higher throughput because the *borrowing strategy* is also adopted in addition to *LNA* neighborhood allocation. The improvement is less in the 12×12 topology size because the relative size of an application is smaller compared to the platform, creating less chances for avoiding high-aged cores. The results also show that *LBRM* is also effective in terms of throughput improvement for realistic task graphs.

TABLE IV
INFORMATION OF REALISTIC TASK GRAPHS.

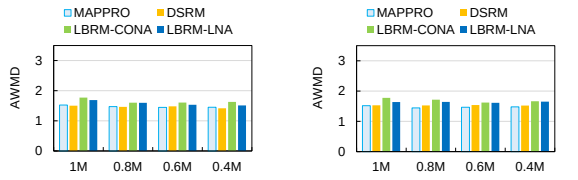
Name	Size	CCR	Name	Size	CCR
MP3 Encoder	13	0.008	H.263 Decoder	14	0.008
H.263 Encoder	12	0.093	VOPD	16	1
MWD	12	0.256	MPEG4	12	0.256
PIP	8	0.608	TVOPD	38	0.724



(a) Many-core size: 8×8 (b) Many-core size: 12×12

Fig. 10. Throughput comparison for realistic benchmarks.

2) AWMD Comparisons: Fig. 11 shows the comparisons of communication distance for realistic task graphs. The results show that although *LBRM-CONA* and *LBRM-LNA* allow dispersed mapping of some applications, the AWMD does not increase significantly. The main reason for this result is that most of the dispersed applications have low communication. We can conclude that *LBRM* can effectively increase the throughput with only a slight penalty on the communication cost. In addition, *LBRM-LNA* has a lower AWMD than *LBRM-*



(a) Many-core size: 8×8 (b) Many-core size: 12×12

Fig. 11. AWMD Comparisons for realistic benchmarks.

TABLE V
COMMUNICATION OVERHEAD VS. MANY-CORE SIZE.

Many-core size	8×8	16×16	24×24	32×32
Communication (bits)	17,472	157,440	569,664	1,364,992

CONA because of the *LNA* neighborhood allocation method.

F. Overhead Analysis of the Runtime Mapping

In this paper, the runtime mapping is performed by a centralized resource manager, which is responsible for monitoring the status of the cores and making mapping decisions. According to [20], the overhead of the runtime mapping includes the message transmissions in the control network and the online mapping algorithms.

1) Communication Overhead: For a mesh-based on-chip interconnect, the communication energy is nearly proportional to the total communication volume [40]. Assuming that the 32 bits are used to represent the lifetime budget, Table V presents the communication volume vs. the many-core size according to the method in [20]. When the many-core size is 8×8, the energy overhead for transmitting the control messages is negligible compared to realistic applications (usually in the megabit range) [20]. However, when the size increases to 32×32, the high communication volume poses a higher communication bandwidth requirement around the centralized resource manager. Therefore, it can be concluded that a distributed resource management method is more preferable in a larger many-core system, which is beyond the scope of this paper and left as a future work.

2) Computational Overhead of the Reliability Factor and Neighborhood Allocation: The mapping includes neighborhood allocation algorithm, the calculation of Threshold parameter, and the calculations of reliability factor. We evaluate the computational overhead of these three calculations on an ARM processor (1 GHz) in gem5 simulator. It is assumed that the application size is 30, and the topology size is 8×8 or 12×12. (1) The results show that the proposed neighborhood allocation algorithm *LNA* takes $34\mu s$ and $36\mu s$ for 8×8 and 12×12 topology sizes, respectively. For comparison, *CONA* takes $30\mu s$ and $31\mu s$ for 8×8 and 12×12 topology sizes, respectively. This is reasonable because *LNA* has higher complexity than *CONA*. (2) The calculations of reliability factor takes $272\mu s$ and $483\mu s$ for 8×8 and 12×12 topology sizes, respectively. However, they are calculated proactively, e.g., the calculations are performed after last application mapping, and the results are ready when the next application mapping arrives. Thus it would not affect the running time of the algorithm if the time interval of arrival time is long enough.

3) Computational Overhead of Threshold Parameter: Eq. 11 needs to calculate $\sum_{n_i,n_j \in D} MD(n_i, n_j)$, which is the all-to-all Manhattan distance of all the remaining cores. Therefore, the calculation of the Threshold parameter involves a high complexity and possibly incurs too long running time at runtime. We propose an algorithm to simplify this calculation. The main idea is to calculate the horizontal distance and the vertical distance from other nodes separately, and some calculations are judiciously reused. For each node n_i, the total horizontal distances to all the nodes on the left side, on the right side, on the upside and on the down side are denoted as $LD(n_i)$, $RD(n_i)$, $UD(n_i)$ and $DD(n_i)$, respectively. Therefore, the total distance from n_i to all the other nodes is calculated as $\sum_{n_j \in D} MD(n_i, n_j) = LD(n_i) + UD(n_i) + RD(n_i) + DD(n_i)$. We use $L(k)$ to denote the total horizontal distance from column k to all the nodes on the left of column k, and (x_i, y_i) denotes the coordinate of n_i in the mesh. It can be derived that $LD(n_i) = L(x_i)$. Because the horizontal distance to each node on the left side increases by 1 when moving 1 hop to the right column, $L(k)$ can be represented in a recursive manner as follows,

$$L(k+1) = L(k) + M(k+1), \forall k \in [0, N-2] \qquad (14)$$

where $M(k)$ is the total number of nodes on the left side of column k. Since $L(0) = 0$, $L(1)$ to $L(N-1)$ can be calculated recursively. Then $LD(n_i)$ of all the nodes can be easily obtained from $L(k)$. Similarly, $RD(n_i)$, $UD(n_i)$ and $DD(n_i)$ can be calculated. Finally, the total distance of each node to all the other nodes is calculated. Using this method, the Threshold parameter can be calculated with a much shorter runtime.

The experimental results show that the method reduces the computational overhead of the Threshold parameter from $138\mu s$ to $10\mu s$ for 8×8 topology size, from $972\mu s$ to $31\mu s$ for 12×12 topology size, and from $2207\mu s$ to $63\mu s$ for 16×16 topology size. Besides, the calculation of the Threshold parameter is also performed proactively similar as the calculation of reliability factor. Therefore, the Threshold parameter has low computational complexity and has little impact on the running time of the mapping algorithm.

VI. ACKNOWLEDGMENTS

This work is supported in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China (Grant No. 2018ZX01028201), and in part by the National Natural Science Foundation of China (Grant No. 61672317, No.61834002).

VII. CONCLUSIONS

In this paper, we propose a lifetime-constrained runtime mapping scheme, which can dynamically map the incoming applications on the many-core system under a lifetime constraint. Compared to existing lifetime reliability-constrained mapping schemes, the proposed mapping algorithm adopts the *borrowing strategy* to manage the many-core resources in

multiple scales. In other words, the lifetime is managed in a long-term scale but is locally relaxed to satisfy performance requirements in a short-term scale. An application mapping-based two-level controller is presented to control the lifetime reliability and performance by runtime application mapping. The applications are classified into communication-intensive and computation-intensive applications, and they are mapped with different strategies. We also propose a method to classify the applications as communication-intensive and computation-intensive applications depending on the current locations of available cores such that the throughput can be effectively improved. The simulation results show that compared to the state-of-the-art lifetime-constrained mapping scheme, the proposed runtime mapping can effectively improve the throughput of many-core systems given a lifetime reliability constraint.

REFERENCES

[1] C. Ramey, "Tile-gx100 manycore processor: Acceleration interfaces and architecture," in *Proceedings of IEEE Hot Chips 23 Symposium (HCS)*, 2011, pp. 1–21.

[2] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu, "Knights landing: Second-generation intel xeon phi product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, Mar. 2016.

[3] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proceedings of 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–10.

[4] C. L. Chou and R. Marculescu, "Run-time task allocation considering user behavior in embedded multiprocessor networks-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, no. 1, pp. 78–91, Jan. 2010.

[5] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *Proceedings of 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–6.

[6] M.-H. Haghbayan, A. Kanduri, A.-M. Rahmani, P. Liljeberg, A. Jantsch, and H. Tenhunen, "Mappro: Proactive runtime mapping for dynamic workloads by quantifying ripple effect of applications on networks-on-chip," in *Proceedings of the 9th International Symposium on Networks-on-Chip (NoCS)*, 2015, pp. 26:1–26:8.

[7] J. Ng, X. Wang, A. K. Singh, and T. Mak, "Defragmentation for efficient runtime resource management in noc-based many-core systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 11, pp. 3359–3372, Nov. 2016.

[8] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel, "Defragmentation of tasks in many-core architecture," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 1, pp. 2:1–2:21, Mar. 2017.

[9] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA)*, 2004, pp. 276–285.

[10] W. Song, S. Mukhopadhyay, and S. Yalamanchili, "Architectural reliability: Lifetime reliability characterization and management of many-core processors," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 103–106, 2015.

[11] L. Huang, F. Yuan, and Q. Xu, "Lifetime reliability-aware task allocation and scheduling for mpsoc platforms," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2009, pp. 51–56.

[12] Z. Yang, C. Serafy, T. Lu, and A. Srivastava, "Phase-driven learning-based dynamic reliability management for multi-core processors," in *Proceedings of the 54th Annual Design Automation Conference (DAC)*, 2017, pp. 46:1–46:6.

[13] P. Mercati, A. Bartolini, F. Paterna, T. S. Rosing, and L. Benini, "Workload and user experience aware dynamic reliability management in multicore processors," in *Proceedings of 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–6.

[14] C. Zhuo, D. Sylvester, and D. Blaauw, "Process variation and temperature-aware reliability management," in *Proceedings of Design, Automation and Test in Europe Conference Exhibition (DATE)*, 2010, pp. 580–585.

[15] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability modeling and management in dynamic microprocessor-based systems," in *2006 43rd ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 1057–1060.

[16] T. Kim, X. Huang, H. B. Chen, V. Sukharev, and S. X. D. Tan, "Learning-based dynamic reliability management for dark silicon processor considering em effects," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 463–468.

[17] L. Huang and Q. Xu, "Energy-efficient task allocation and scheduling for multi-mode mpsocs under lifetime reliability constraint," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2010, pp. 1584–1589.

[18] M. H. Haghbayan, A. Miele, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "A lifetime-aware runtime mapping approach for many-core systems in the dark silicon era," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 854–857.

[19] L. Wang, X. Wang, H.-f. Leung, and T. Mak, "Throughput optimization for lifetime budgeting in many-core systems," in *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, 2017, pp. 451–454.

[20] C. L. Chou, U. Y. Ogras, and R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 10, pp. 1866–1879, Oct. 2008.

[21] M. Fattah, M. Ramirez, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Cona: Dynamic application mapping for congestion reduction in many-core systems," in *Proceedings of 30th International Conference on Computer Design (ICCD)*, 2012, pp. 364–370.

[22] M. Fattah, P. Liljeberg, J. Plosila, and H. Tenhunen, "Adjustable contiguity of run-time task allocation in networked many-core systems," in *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014, pp. 349–354.

[23] A. S. Hartman and D. E. Thomas, "Lifetime improvement through runtime wear-based task mapping," in *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2012, pp. 13–22.

[24] J. Sun, R. Lysecky, K. Shankar, A. Kodi, A. Louri, and J. Roveda, "Workload assignment considering nbti degradation in multicore systems," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 10, no. 1, pp. 4:1–4:22, Jan. 2014.

[25] W. Liu, J. Yi, M. Li, P. Chen, and L. Yang, "Energy-efficient application mapping andscheduling for lifetime guaranteed mpsocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2018, early access.

[26] D. Gnad, M. Shafique, F. Kriebel, S. Rehman, D. Sun, and J. Henkel, "Hayat: Harnessing dark silicon and variability for aging deceleration and balancing," in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 180:1–180:6.

[27] S. Kobbe, L. Bauer, D. Lohmann, W. Schrder-Preikschat, and J. Henkel, "Distrm: Distributed resource management for on-chip many-core systems," in *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2011, pp. 119–128.

[28] P. Mercati, A. Bartolini, F. Paterna, L. Benini, and T. S. Rosing, "An online reliability emulation framework," in *Proceedings of the 12th IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, 2014, pp. 334–339.

[29] S. Pagani, H. Khdr, W. Munawar, J.-J. Chen, M. Shafique, M. Li, and J. Henkel, "Tsp: Thermal safe power: Efficient power budgeting for many-core systems in dark silicon," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2014, pp. 10:1–10:10.

[30] M. H. Haghbayan, A. Rahmani, A. Y. Weldezion, P. Liljeberg, J. Plosila, A. Jantsch, and H. Tenhunen, "Dark silicon aware power management for manycore systems under dynamic workloads," in *Proceedings of International Conference on Computer Design (ICCD)*, 2014, pp. 509–512.

[31] H. Khdr, S. Pagani, M. Shafique, and J. Henkel, "Thermal constrained resource management for mixed ilp-tlp workloads in dark silicon chips," in *Proceedings of the 52nd Annual Design Automation Conference (DAC)*, 2015, pp. 179:1–179:6.

[32] X. Wang, A. K. Singh, B. Li, Y. Yang, H. Li, and T. Mak, "Bubble budgeting: Throughput optimization for dynamic workloads by exploit-

ing dark cores in many core systems," *IEEE Transactions on Computers (TC)*, vol. 67, no. 2, pp. 178–192, Aug. 2018.
[33] L. Yang, W. Liu, W. Jiang, M. Li, P. Chen, and E. H. M. Sha, "Fotonoc: A folded torus-like network-on-chip based many-core systems-on-chip in the dark silicon era," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 28, no. 7, pp. 1905–1918, July 2017.
[34] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 469–480.
[35] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, May 2006.
[36] C. Bolchini, M. Carminati, M. Gribaudo, and A. Miele, "A lightweight and open-source framework for the lifetime estimation of multicore systems," in *Proceedings of International Conference on Computer Design (ICCD)*, 2014, pp. 166–172.
[37] F. Suter, "Daggen: A synthethic task graph generator," https://github.com/frs69wq/daggen, 2013.
[38] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. D. Micheli, "Noc synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 16, no. 2, pp. 113–129, Feb. 2005.
[39] S. Murali, C. Seiculescu, L. Benini, and G. D. Micheli, "Synthesis of networks on chips for 3d systems on chips," in *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2009, pp. 242–247.
[40] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular noc architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 24, no. 4, pp. 551–562, April 2005.

Liang Wang received the BEng and MSc degree in electronics engineering from Harbin Institute of Technology, Harbin, China, in 2011 and 2013 respectively, and the Ph.D degree in Computer Science and Engineering from The Chinese University of Hong Kong, Hong Kong, China, in 2017.

He is currently a postdoctoral research fellow at Institute of Microelectronics, Tsinghua University, China. His research interests include power-efficient and reliability-aware design for network-on-chip and many-core system.

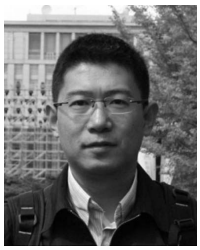
He was a recipient of the VLSI-SoC 2014 Best Paper Awards.

Ping Lv received the M.S. degree of communication and information system from National Digital Switching System R&D Center (NDSC) in 2007. She is currently an Associate Professor with NDSC.

Her current research interests include reconfigurable computing, mobile computing, low power design technology and application-specific integrated circuit.

Leibo Liu received the BS degree in electronic engineering from the Tsinghua University, Beijing, China, in 1999 and the PhD degree in the Institute of Microelectronics, Tsinghua University, in 2004. He now serves as a Tenured Professor in the Institute of Microelectronics, Tsinghua University.

His research interests include reconfigurable computing, mobile computing and VLSI DSP.



Jie Han received the BSc degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the PhD degree from Delft University of Technology, Delft, The Netherlands, in 2004. He is currently an assistant professor in the Department of Electrical and Computer Engineering, the University of Alberta, Edmonton, AB, Canada.

His research interests include approximate computing, stochastic computation, reliability and fault tolerance, nanoelectronic circuits and systems, novel computational models for nanoscale and biological applications.

Dr. Han and coauthors received the Best Paper Award at the NanoArch 2015 and Best Paper Nominations at the GLSVLSI 2015 and NanoArch 2016.

Ho-fung Leung received the B.Sc. and M.Phil. degrees in computer science from The Chinese University of Hong Kong, Hong Kong, and the Ph.D. degree from the University of London, London, U.K., with a Diploma of Imperial College in computing from Imperial College London, London, U.K. He is currently a professor with the Department of Computer Science and Engineering, the Chinese University of Hong Kong.

His research interests includes ontologies, intelligent agents, self-organizing systems, complex systems, social computing, service-oriented architectures, and cloud computing. He is a Fellow of BCS, HKIE, a senior member of ACM and IEEE, a full member of HKCS.

He was the chairperson of ACM Hong Kong Chapter, and is a Chartered Engineer registered by the Engineering Council.

Xiaohang Wang received the BEng and PhD degrees in communication and electronic engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. He is currently an Associate Professor with the South China University of Technology, Guangzhou, China.

His current research interests include manycore architecture, power efficient architectures, optimal control, and network-on-chip-based systems.

Dr. Wang was a recipient of the PDP 2015 and VLSI-SoC 2014 Best Paper Awards.

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

His research interests include SoC design, reconfigurable computing, and mobile computing.

Shaojun Wei was born in Beijing, China, in 1958. He received the Ph.D. degree from the Fault Polytechnique de Mons, Belgium, in 1991. He is currently a Professor with the Institute of Microelectronics, Tsinghua University. He is also a Senior Member of the Chinese Institute of Electronics.

His main research interests include VLSI SoC design, EDA methodology, and ASIC design.

Terrence Mak received the BEng and MPhil degrees in systems engineering from the Chinese University of Hong Kong, in 2003 and 2005, respectively, and the PhD degree from Imperial College London in 2009. He is currently an associate professor at Electronics and Computer Science, University of Southampton. Supported by the Royal Society, he was a Visiting Scientist at Massachusetts Institute of Technology during 2010, and also, affiliated with the Chinese Academy of Sciences as a Visiting Professor since 2013.

He has received three prestigious best paper awards at the Design, Automation & Test in Europe Conference and Exhibition in 2011, the IEEE/Association for Computing Machinery VLSI-SoC 2014, and the IEEE PDP 2015. He also received the Croucher Foundation Scholar from Sun Lab. In 2015, his published journal based on 3-D adaptation and deadlock-free routing received the prestigious 2015 IET Computers & Digital Techniques Premium Award.