CHAPTER THREE

Multiobjectivism in Dark Silicon Age

Amin Rezaei*, Masoud Daneshtalab†, Hai Zhou*
*Department of Electrical Engineering and Computer Science, Northwestern University (NU), Evanston, IL, United States
†Division of Intelligent Future Technologies, Mälardalen University (MDH), Vasteras, Sweden

Contents

Abstract

MCSoCs, with their scalability and parallel computation power, provide an ideal implementation base for modern embedded systems. However, chip designers are facing a design challenge wherein shrinking component sizes though have improved density but started stressing energy budget. This phenomenon, that is called utilization wall, has revolutionized the semiconductor industry by shifting the main purpose of chip design from a performance-driven approach to a complex multiobjective one. The area

Advances in Computers, Volume 110
ISSN 0065-2458
https://doi.org/10.1016/bs.adcom.2018.03.012

of the chip which cannot be powered is known as dark silicon. In this chapter, we address the multiobjectivism in dark silicon age. First, we overview state-of-the-art works in a categorized manner. Second, we introduce a NoC-based MCSoC architecture, named shift sprinting, in order to increase overall reliability as well as gain high performance. Third, we explain an application mapping approach, called round rotary mapping, for HWNoC-based MCSoC in order to first balance the usage of wireless links by avoiding congestion over wireless routers and second spread temperature across the whole chip by utilizing dark silicon. Finally, we conclude the chapter by providing a future outlook of dark silicon research trend.

1. INTRODUCTION AND BACKGROUND

Embedded systems are now ubiquitous, as evidenced by smartphones, automobiles, game players, and smart appliances. Applications with ever-increasing demand of processing speed and capability for handling huge data impose formidable requirements on these systems. Many-Core System-on-Chips (MCSoCs), with their scalability and parallel computation power, provide an ideal implementation base for modern embedded systems.

However, chip designers are facing a design challenge wherein shrinking component sizes though have improved density but started stressing power budget. As a result, only some parts of the entire chip can be run at maximum permissible frequency, while the remaining parts should either be switched off (i.e., dark silicon) or run at lower frequency (i.e., dim silicon). The phenomenon of limiting performance rating, that is called utilization wall [1], has revolutionized the semiconductor industry by shifting the main purpose of chip design from a performance-driven approach to a complex multiobjective one. This multiobjectivism includes but is not limited to designing high-performance, low-power, high-reliability, and low-temperature embedded systems.

The growing interest to do more research in dark silicon field is completely beheld in recent publications. Up until now, researchers have adopted different perspectives toward the utilization wall problem that will be reviewed in this section.

1.1 The Perennial Challenge of Power Management

Power has been always a challenge in modern MCSoCs and the utilization wall problem inherently makes this challenge even more momentous.

In this regard, GreenDoid is presented for mobile applications using Conservation Cores (C-Cores) [2]. C-Cores are specialized coprocessors designed to reduce the energy athwart all the program code. They are combined with energy-efficient General-Purpose Processors (GPPs). Frequently executed pieces of the code are implemented using the C-Cores, while occasionally executed pieces are run on the GPPs. Moreover, a dark silicon architecture for servers is presented [3]. The chip area is occupied with an array of different application-specific cores. By dynamically powering up a small number of these cores, the other cores can remain dark while peak performance and power efficiency are achieved.

As an advanced approach, Computational Sprinting (CS) is proposed [4] using phase-change materials to allow cores to exceed their sustainable thermal budget for subsecond durations, providing a short but substantial computational boost. Nowadays, commercial MCSoCs are available based on Network-on-Chip (NoC) [5] communication infrastructure. It is also predicted that upcoming MCSoCs will progressively continue operating on completely new principles and novel NoC-based architectures. Thus, NoC-Sprinting (NS) is introduced [6], in which the chip selectively sprints to any intermediate stages instead of directly activating all the cores in response to short-burst computations. The mode-switching in CS lacks adaptability and only considers two states of single-core operation or all-core sprinting. However, based on the behaviors of the running applications, some in-between numbers of active cores may be sufficient for reaching the optimal performance speedup with less power consumption. On the other hand, NS lacks reliability and does not clearly consider the cooldown period required for each core after sprinting phase. By assigning sprinting periods to all or some fixed cores of the system, both CS and NS are categorized in periodical sprinting class. To address periodical sprinting problems, the concept of distributional sprinting is introduced [7], utilizing cooldown period to provide high performance for streaming applications.

Most of the papers in dark silicon field have considered heterogeneous architectures. Even the evaluation results in Ref. [8] demonstrate that building heterogeneous MCSoCs with different materials can efficiently utilize the chip-level resource and deliver the optimal balance among performance, power consumption, and cost. However, the authors in Ref. [9] have challenged the conventional wisdom that dark silicon chips must be heterogeneous in nature by building a case for homogeneous MCSoCs. The proposed solutions in this chapter are also based on homogeneous MCSoCs.

Additionally, keeping components cool is one of the most important challenges that becomes more severe by semiconductor technology scaling.

Overheating causes significant reductions in the operating life of a device. Moreover, uncertainties in reliability can lead to performance, cost, and time-to-market penalties [10]. Thus, challenges and opportunities of the emergence of dark silicon in the context of thermal management and reliability are discussed in Ref. [11]. Moreover, in Ref. [12] a new power budget concept, called Thermal Safe Power (TSP), is presented which results in a higher total system performance, while the maximum temperature among all cores remains below the threshold level that triggers dynamic thermal management.

1.2 The Essence of Efficient Application Mapping Schemes

Since the cores asymmetrically degrade in dark silicon age, the application mapping procedure is disrupted. Thus, traditional approaches can no longer satisfy the complex multiobjective goals of the system.

In Ref. [13] a sustainability control system is designed that monitors aging and passes core utilization guidelines to a Central Manager (CM) responsible for application mapping. The mapping procedure follows two main objectives: First, sustaining the benefits of differential reliability and then, maximizing energy efficiency. Moreover, since switching on all the clusters in a cluster-based MCSoC may violate power budget, an application arrival rate runtime scheduler is proposed [14] to minimize mean service time within a power budget. Applications can be migrated between clusters and typically, one cluster is active at any point, while the other clusters remain dark.

In Ref. [15] a thermal-aware application mapping policy is proposed to determine the locations of active and dark cores for each application on the chip, such that the potential temperature increase is reduced as much as possible. In Ref. [16] an efficient heuristic is employed to jointly optimize the dark silicon pattern and application mapping, which is enabled by a light-weight temperature prediction mechanism that provides an estimate of the chip temperature profile resulting from a candidate solution. In Ref. [17] a runtime application mapping approach is presented that dynamically adapts the degree of parallelism to minimize average application service times and energy.

Despite the fact that a NoC-based architecture has many advantages, its multihop nature has negative impact on both latency and power consumption parameters especially when the network size increases. Therefore, alternative technologies such as Hybrid Wireless NoC (HWNoC) have introduced [18]. Also, a temperature- and congestion-aware application mapping algorithm is presented [19] targeted at tackling two critical concerns in emerging

HWNoC-based MCSoCs: Alleviation of the severe congestion on Wireless Routers (i.e., the routers equipped with wireless transceivers, WRs) and prevention of persistent hot spots in the network.

1.3 The Emergence of Novel Perspectives

Another way to tackle the utilization wall problem is to change the traditional thinking by proposing new computing schemes, new computation components, or even new electronic devices. However, in most of the times, changing everything from basis is not an electronic factory cup of tea. Thus, the proposed solutions should be fully or partially compatible with existing technologies.

In Ref. [20] the authors have proposed to use approximate computing in programming level. However, the runtime system should be configured in a way that the approximation does not produce unacceptable outputs. In Ref. [21] the authors have identified new research opportunities in optimization of interconnects in the scenario of accelerators. In Ref. [22] the authors have used a combination of steep slope and CMOS devices in the design of MCSoCs. They have adopted the most promising steep-slope device candidate, interband Tunnel Field-Effect Transistors (TFETs), and evaluate a CMOS-TFET MCSoC.

2. SHIFT SPRINTING: RELIABLE TEMPERATURE-AWARE NoC-BASED MCSoC ARCHITECTURE IN DARK SILICON AGE

In recent years, tile-based architectures with NoC as communication framework are emerging as an attractive alternative to bus-based systems. Nowadays, users are using their smartphones not only for daily communication but also increasingly for media streaming. Thus, the Quality of Service (QoS) of real-time streaming applications will become more and more important for next generations of mobile devices. Furthermore, reliability issues are highly challengeable in the future mobile devices due to the limited cooling options. In short, designing reliable mobile devices to provide QoS-aware real-time streaming for the users is crucially important in dark silicon age. In this section, a fine-grained NoC-based MCSoC architecture, named Shift Sprinting (SS) [7], is introduced in order to reliably utilize dark silicon under the power budget constraint.

2.1 Preliminaries and Motivations

As a preliminary study, a 2×2 NoC-based SoC is simulated under uniform streaming traffic to show the necessity of high-performance and high-reliability demands of real-time streaming applications. By an optimistic assumption each sprinting period is considered to be equal to the cooldown period. Moreover, each core in sprinting status is supposed to gain performance by a factor of four and to lose life span by a factor of two compared to the core in nominal status. Maximum traffic injection rate and the average lifetime of the system running in idle status are called λ_{full} and γ_{idle}, respectively.

2.1.1 High-Performance Demands

It is shown in Refs. [4,6] that for serving short-burst computations, interleaving the status of the cores between nominal and sprinting along with cooldown intervals can be beneficial. Since in sprinting status the core is operating on higher than the Thermal Design Power (TDP) constraint, phase change of the core internal materials can be used to tolerate such situation. It is assumed that the core temperature stays constant for a specified time during the melting phase of the materials.

However, applying short-burst computations is not enough to fully support real-time streaming applications (e.g., watching a live football match) because these applications require continues high-computation demands (i.e., frequent sprinting periods) in order to provide QoS to the users. Also, as shown in Fig. 1A, applying periodical sprinting to a fixed set of cores does not solve the problem neither since it still requires cooldown intervals. On the other hand, as proposed in Fig. 1B, migrating the running application to dark cores utilizes cooldown periods and provides appropriate QoS to the users.

Fig. 2 shows the average network latency for both periodical sprinting and distributional sprinting in a 2×2 NoC-based SoC. By increasing the traffic injection rate, the weakness of periodical sprinting over distributional sprinting is fully observed.

2.1.2 High-Reliability Demands

Systems designed for dark silicon allow periodical sprinting to some specific cores while let others stay at dark. This greatly increases the permanent failure probability of those highly active cores that may lead to system failure. Hence, distributing the sprinting periods through the whole system can reduce the core malfunction possibility. Fig. 3 demonstrates the failure

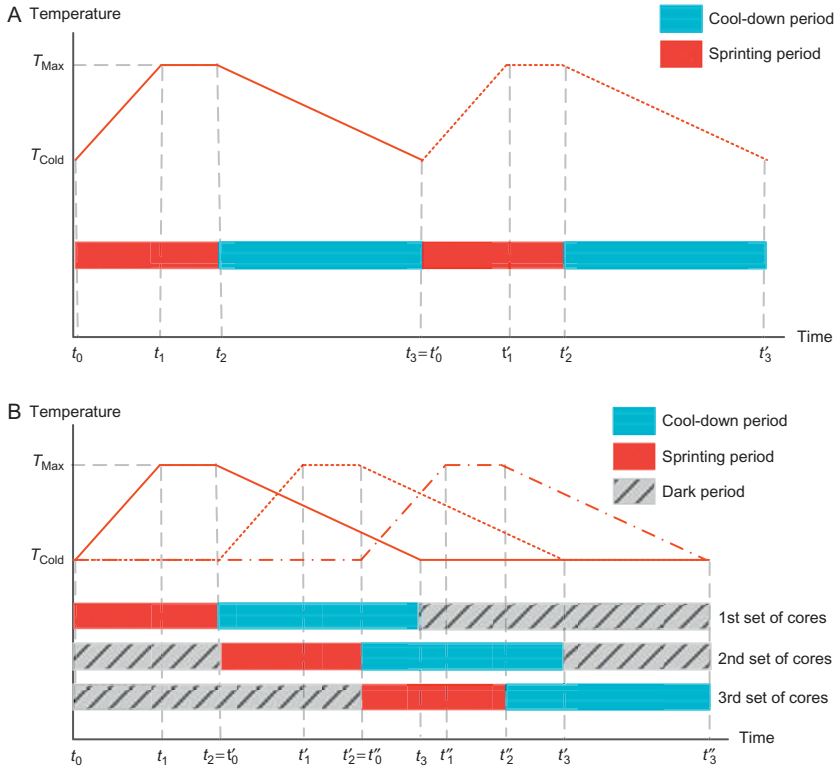

Fig. 1 High-performance demands (A) periodical sprinting and (B) distributional sprinting.

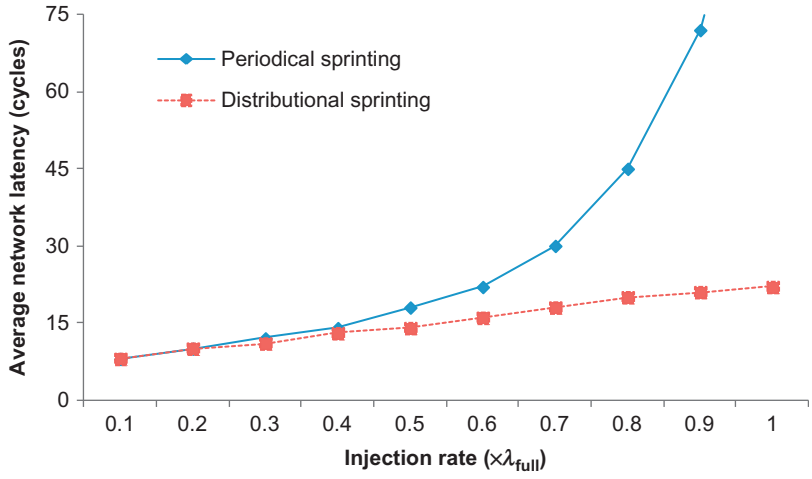

Fig. 2 Average network latency for 2×2 NoC-based SoC with periodical sprinting and distributional sprinting.

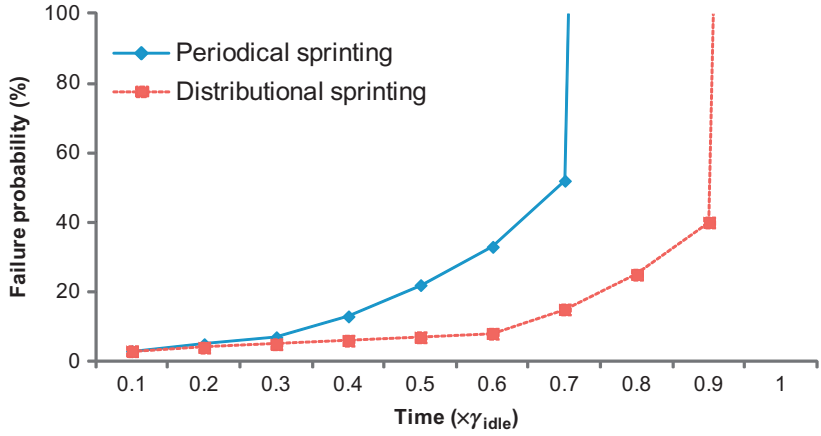

Fig. 3 Failure probability for 2 × 2 NoC-based MCSoC with periodical sprinting and distributional sprinting.

probability of the system over time in both periodical sprinting and distributional sprinting in a 2 × 2 NoC-based SoC under the injection rate of 0.5 λ_{full}. It is supposed that failure of a single core leads to the whole system failure. As it can be seen the failure probability of periodical sprinting is almost as 1.5 × as likely as distributional sprinting.

2.2 SS Architecture

By employing the concept of distributional sprinting, both high-performance and high-reliability demands of NoC-based MCSoCs can be fulfilled in dark silicon age.

2.2.1 Core Behavior Model

The core behavior model of SS is depicted in Fig. 4. Each core has four main states including dark, idle, active, and malfunction.

Dark: In the future generations of MCSoCs, the common state of the core is dark. In this state the core is power-gated.

Idle: After waking up the core, it goes to idle state (i.e., the core is powered on but still no application is assigned to it). Moreover, after the application departure, the core goes to warm status until it cools down and reaches the cold status. Then, it can go back to dark state.

Active: When an application arrives to the core, it goes to active state. In active state the status is interchangeable between nominal and sprinting. In nominal status, the core is operating under the TDP constraint. On the

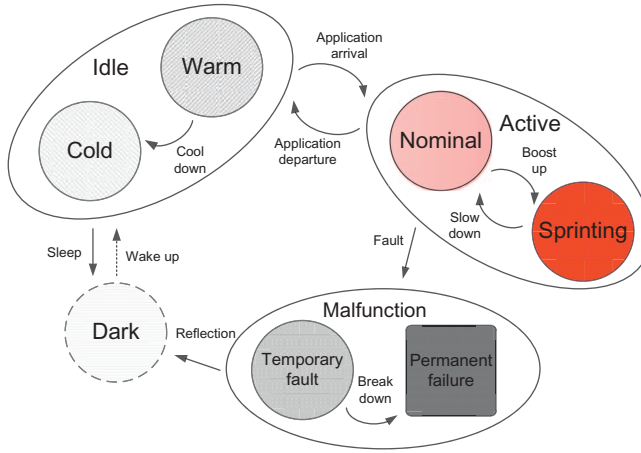

Fig. 4 Core behavior model of shift sprinting.

other hand, in sprinting status, the core is operating on higher than the TDP for a temporary period in order to speed up the process.

Malfunction: In the case of fault happening, the core state is changed to malfunction. If the fault is temporary, after resolving the problem, the core can go back into the normal cycle. Otherwise, it comes to permanent failure.

2.2.2 System Topology

Rather than abandon the benefits of transistor density scaling, some cores are transiently allowed to operate on higher than the TDP. The mobile platform trend shows that the future MCSoCs with the same die area as current mobile chips will have enough dark cores (i.e., on average 52% [1]) to support additional cores during sprinting. The topology of SS is based on the 2D mesh NoC. As an example we have eight different phases (i.e., four sprinting and four nominal) in SS shown in Fig. 5. The shift can happen between different phases whenever necessary. In nominal phases, a single core is operating under the TDP constraint. On the contrary, in sprinting phases thermal capacitance of chosen cores is increased over short timescales in order to boost up the process (i.e., the sprinted cores operate on higher than the TDP). Based on each application characteristics, the number of sprinted cores required to provide maximal performance speedup varies.

In most of the available dynamic application mapping techniques in the literature, one core is already dedicated to the CM. In SS, CM is also used to globally control application migration process. In order to speed up the application migration process, in addition to the CM, there are three

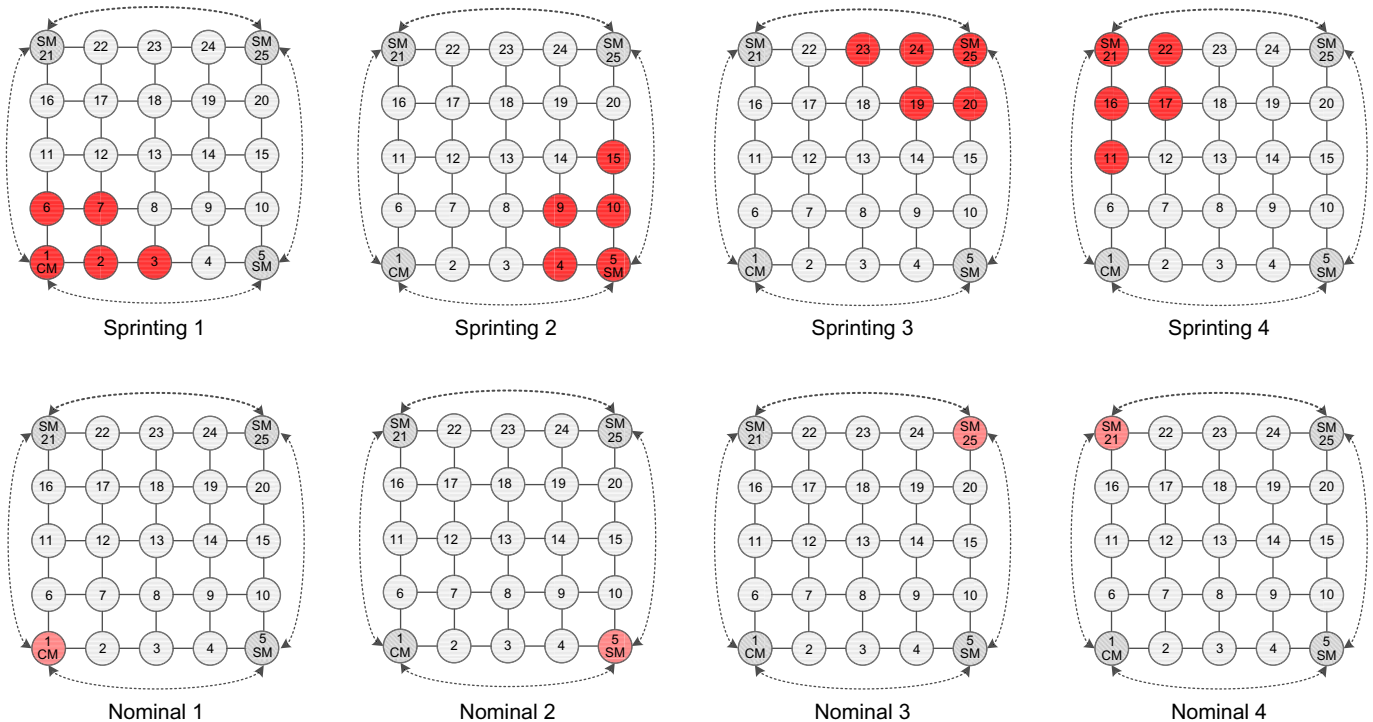

Fig. 5 A 5×5 shift sprinting architecture.

Sub-Managers (SMs) that are responsible for collecting information from other cores. Each core sends and receives information from its nearest manager. For controlling the application migration process efficiently, the managers have created a ring topology network. As it can be seen from Fig. 5 the CM is resided in one corner (i.e., core #1) and three SMs are resided in other corners (i.e., core #5, core #21, and core #25); they formed a ring network all together. This network can be characterized either by wireless or virtual ring network.

Wireless Ring Network (WRN): In WRN, by applying long-range, high-bandwidth, and low-power wireless links [23], a real ring network is formed on top of the 2D mesh NoC. In this case, only the manager routers are required to be equipped with wireless communication capabilities.

Virtual Ring Network (VRN): In VRN, physical channel has a virtual channel that can be dynamically configured for low-latency and low-power connection with a high priority [24]. However, all the NoC routers have to be equipped with extra virtual channels to support these connections.

2.2.3 Application Migration Scheme

If the temperature reaches a certain threshold, the application migration (i.e., shifting between different phases) will start. One upper-bound threshold for each core and a pair of upper-bound and lower-bound thresholds for the whole system are defined. SS state diagram is shown in Fig. 6.

Maximum Core Temperature (MCT): When the temperature of each core reaches the MCT, the shift happens from the current sprinting to the next sprinting phase. MCT is a static threshold relies on the core materials and is defined based on $T_{\max}$ values in Fig. 1. MCT threshold is responsible for the inner loop of Fig. 6 (i.e., shifting between different sprinting phases).

Maximum and Average Overall Temperature (MOT and AOT): When the overall temperature of the system reaches the MOT, the shift happens from the current sprinting to the next nominal phase. In reverse, when the overall temperature of the system goes back to the AOT, the shift happens from the current nominal to the sprinting phase. Both MOT and AOT are dynamic thresholds that depend highly on application behaviors. Note that obtaining the optimal values for these dynamic thresholds is beyond the scope of this chapter. MOT and AOT are managed in the outer loop of Fig. 6 (i.e., shifting between sprinting and nominal phases).

In dark silicon age, the required free cores are guaranteed to be available for application migration. With this scheme, instead of waiting for the cores to cooldown after each sprinting phase, the intervals between cooldown periods of the cores are utilized by migrating applications to dark cores

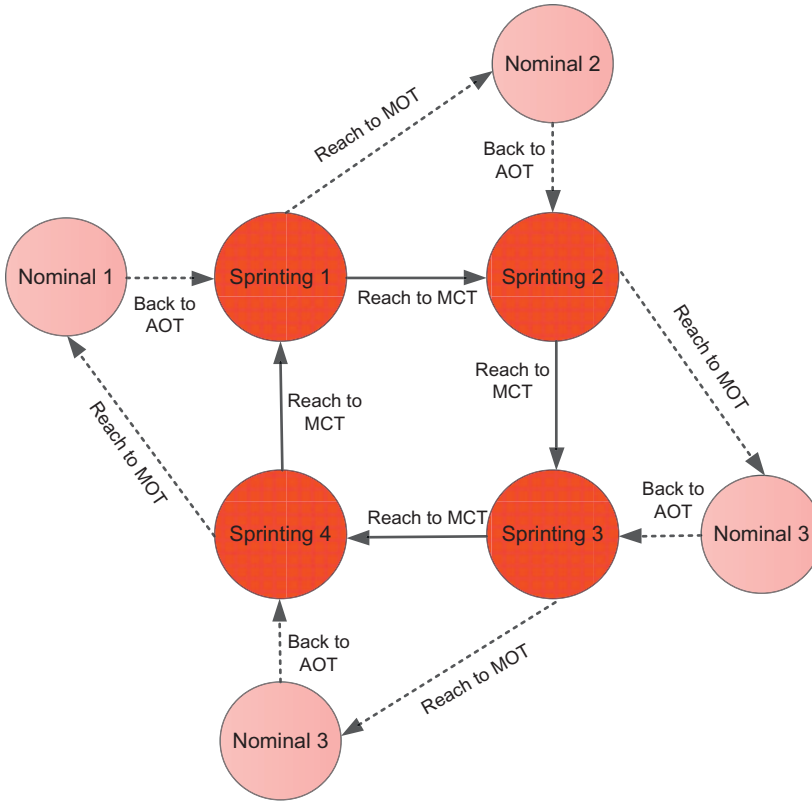

Fig. 6 Shift sprinting state diagram.

and maximizing the sprinting timelines. From evaluation perspective, SS state diagram has an inherent reliability consideration. By distributing the sprinting periods across the whole system, SS spreads the gradual aging process to all the cores. On the other hand, the more the system stays in the inner loop, the better the overall performance is.

2.2.4 Controlling Mechanism

A cost function is needed for the CM in order to determine the best destination core for application migration. When the threshold in one sprinting phase reaches the MCT, the running application will migrate to the next sprinting phase. The destination cores in the next phase are chosen based on the least current temperature order. The system stays in this loop (i.e., inner loop of Fig. 6) until it reaches the MOT threshold. For MOT threshold, all the running applications will migrate to the next phase manager until the overall temperature of the system goes back to AOT. In this case, the running

```
MCT: Maximum core temperature
MOT: Maximum overall temperature
AOT: Average overall temperature
C: Set of all the cores
j: Phase of the system (i.e., phases 1 to 4)
D_j: Set of the dark cores in j_th phase
m_j: Manager core in j_th phase
t_i: Temperature of the core c_i ∈ C
T: Overall temperature of the system

Set the value of MCT
Initiate phase j
while true do
    Set the value of MOT and AOT
    if j is in sprinting phase then
        if T < MOT then
            for ∀c_i ∈ j do
                if t_i > MCT then
                    Choose c_d ∈ D_j+1 with min[t_d]
                    Migrate the running application from c_i to c_d
            Start sprinting phase in j+1
        else
            for ∀c_i ∈ j do
                Migrate the running application from c_j to m_j+1
            Start nominal phase in j+1
    else
        if T<AOT then
            while ∃ running application in m_j do
                Choose multiple c_ds ∈ D_j with min[t_d]s
                Migrate the running application from m_j to the chosen c_ds
            Start sprinting phase in j
    Set the next phase as j
```

Fig. 7 Shift sprinting controlling algorithm.

applications in the manager spread through the dark cores of that phase based on the optimal number of sprinted cores required for each application. Then, the sprinting phase is started again. Fig. 7 shows SS controlling algorithm. It is assumed that only one application can be executed in a sprinted core at each time and the application itself knows the optimal number of required sprinted cores to provide maximal performance speedup.

2.3 Methodology

Experiments are performed on a cycle-accurate many-core platform implemented in SystemC. A pruned version of an open source simulator for mesh-based NoCs, called Noxim [25], is utilized as its communication architecture. For power and temperature simulations, power and thermal models taken from Refs. [26,27] are integrated as libraries into the simulator. By an optimistic assumption each sprinting period is considered to be equal

to the cooldown period. Moreover, each core in sprinting status is supposed to gain performance by a factor of 4 and to lose life span by a factor of 2 compared to the core in nominal status. Maximum traffic injection rate and the average lifetime of the system running in idle status are called λ_{full} and γ_{idle}, respectively. Some multithreaded applications from the PARSEC [28] benchmark suite are used in the experiments. Three different network sizes of 16 (no dark silicon), 36 (55% dark silicon), and 64 (75% dark silicon) cores are considered in the simulations. Comparisons are also made between SS and two state-of-the-arts architectures: CS [4] and NS [6].

2.3.1 Power Model

Power consumption of each core is modeled as two major parameters, i.e., dynamic power due to transistor switching and static power due to leakage. Therefore, total power for each core is given by:

$$P_{\text{total}} = P_{\text{dynamic}} + P_{\text{leakage}} \tag{1}$$

Generally, dynamic power is given by:

$$P_{\text{dynamic}} = \alpha\, C_{\text{e}}\, V_{\text{dd}}^2 f \tag{2}$$

where α is the activity factor, C_{e} is the effective capacitance, V_{dd} is the supply voltage, and f is the running frequency of the core. Assuming both the activity factor and the effective capacitance as constants, dynamic power changes quadratically to supply voltage and linearly to frequency. According to Ref. [29], the static power of a system is given by:

$$P_{\text{leakage}} = V_{\text{dd}}\, N_{\text{tr}}\, k_{\text{d}}\, I_{\text{s}} \tag{3}$$

where N_{tr} is the number of transistors, k_{d} is a device-specific constant, and I_{s} is the normalized static current for each transistor that is proportional to the leakage current of a single transistor.

2.3.2 Thermal Model

For the thermal model, a well-known RC thermal network, defined in Ref. [27], is adopted in the simulator, which considers the duality between thermal and electrical circuits. According to this model, the steady-state temperatures of the cores can be computed as follows:

$$T^{\text{C}} = BP^{\text{C}} + H_1 + H_2 \tag{4}$$

where T^{C} is the steady-state temperatures of all the cores on the chip. Matrix B contains the amount of the heat contribution of all the cores. Thus, heat

transfer among the cores is considered. Column vector H_1 contains the heat contribution of the other thermal nodes to the cores, while H_2 contains the heat contribution of the ambient temperature to the cores. Assuming only one application can be executed in a sprinted core, with respect to the mapping of applications, a binary matrix $L=[L_{i,j}]$ is defined. If application a_j is mapped to core c_i, $[L_{i,j}]=1$; otherwise, $[L_{i,j}]=0$.

By involving the power vector of the applications and the mapping matrix in Eq. (4), we get the following equation:

$$T^{\mathrm{C}} = BLP^{\mathrm{A}} + H \tag{5}$$

where P^A is a column vector containing the power consumptions of all the applications and H is the sum of H_1 and H_2. Furthermore, Eq. (6) expresses the direct relation between application power consumption and the steady-state temperature of any core c_i:

$$T^{c_i} = \sum_{j=1}^{k} b_i l_j p_j + H \tag{6}$$

where b_i is the row i from matrix B which corresponds to core c_i, l_j is the column j of matrix L which corresponds to application a_j, and p_j is the power consumption of application a_j. Hence, the thermal model calculates the steady-state temperature for any core of the system given the mapping matrix of the applications.

2.4 Experimental Results

2.4.1 Performance Evaluation

Fig. 8 shows the normalized execution time of different workloads in SS. In comparison with 16-core NoC (no dark silicon), the average performance improvement of 64-core NoC (75% dark silicon) is 36% and that of 36-core

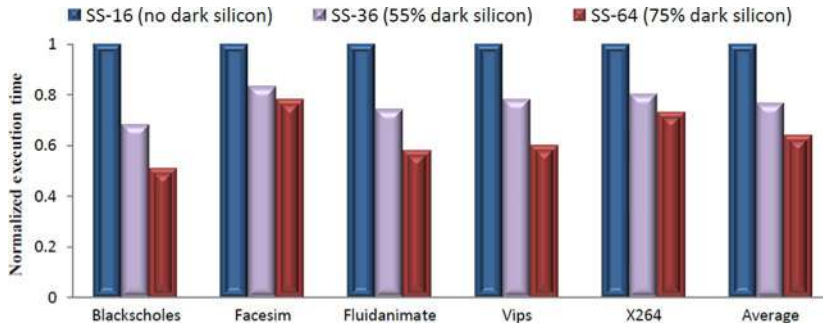

Fig. 8 Execution time comparison between different sizes of SS.

NoC (55% dark silicon) is 24%. As a result, even with increasing of dark silicon area in upcoming MCSoCs, the performance of SS will still improve. This happens because SS utilizes cooldown periods by activating dark cores.

Fig. 9 demonstrates the normalized executing time of different workloads with different architectures for 64-core NoC (75% dark silicon). The results show that SS considerably reduces the execution time compared to other approaches. It achieves 55% and 25% average performance improvement compared to CS and NS, respectively. The performance gain is because of the higher overall sprinting periods provided by utilizing cooldown periods. As it can be seen from Fig. 9, all the architectures performed quite the same under "Blackscholes" workload that is a nonstreaming financial analysis application. This is because "Blackscholes" achieves the optimal performance speedup in CS and hence leave no space for power gating in NS and neither power gating nor application migration in SS.

In addition, Fig. 10 shows the normalized network latency of different workloads with different architectures for 64-core NoC (75% dark silicon)

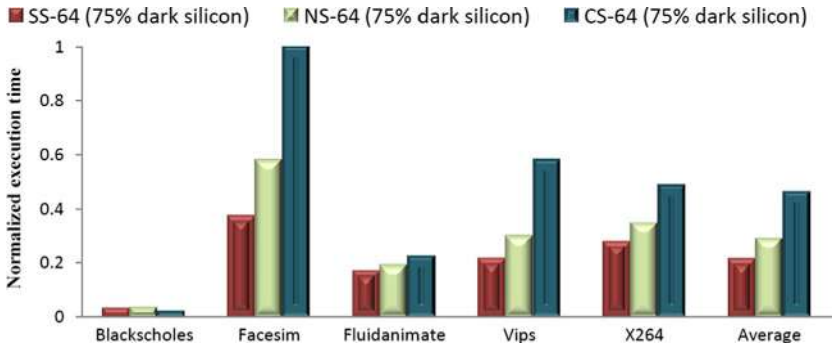

Fig. 9 Execution time comparison between different architectures.

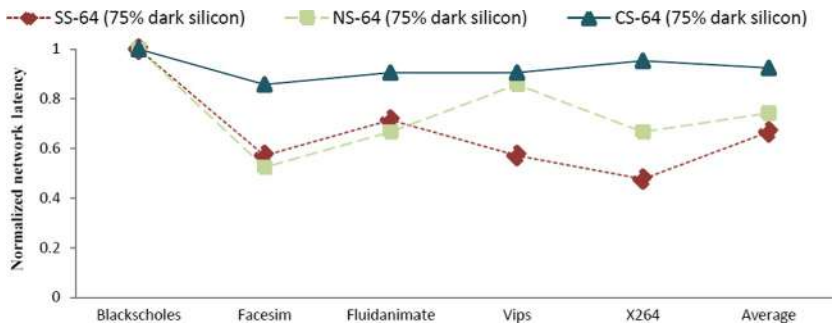

Fig. 10 Network latency comparison between different architectures.

under the injection rate of 0.75 λ_{full}. It can be seen that SS reduces the communication latency for all the applications (28% in average) in comparison with CS and for most of the applications (11% in average) in comparison with NS. SS performs quite well in the media-processing applications (e.g., Vips and X264). On the other hand, performance degradation of SS in the animation workloads (e.g., facesim and fluidanimate) compared with NS is due to application migration overhead. Therefore, still more attempts are required to minimize migration overhead by finding optimal thresholds and making low-overhead migration mechanisms. Applying self-aware mechanisms based on application behavior prediction [30] may decrease the migration overhead.

2.4.2 Power Consumption Measurement

Fig. 11 displays the normalized network power consumption of different workloads with different architectures for 64-core NoC (75% dark silicon) under the injection rate of 0.75 λ_{full}. On average, SS saves 58% power compared to CS while consuming almost the same power as NS. This is because of the fact that dark cores (i.e., 75% of the cores) are power-gated in both SS and NS.

As another evaluation parameter, Fig. 12 depicts the normalized Energy Delay Product (EDP) of different workloads with different architectures for 64-core NoC (75% dark silicon). It can be seen that even with the overhead of application migration approach, the average EDP of SS in the media-processing applications (e.g., Vips and X264) is less than the other architectures that makes it a promising architecture for future mobile devices.

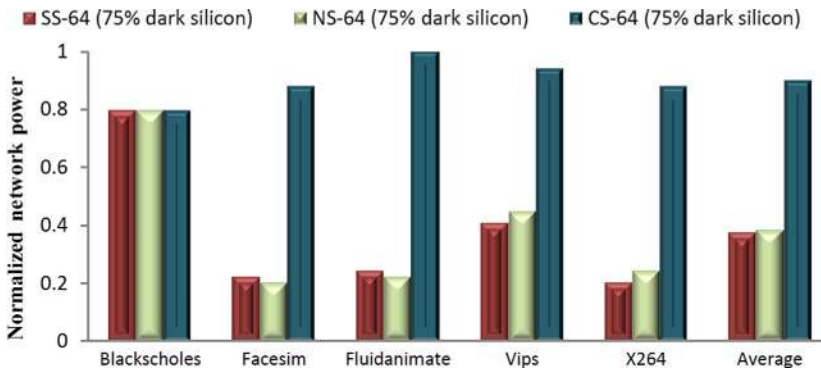

Fig. 11 Network power comparison between different architectures.

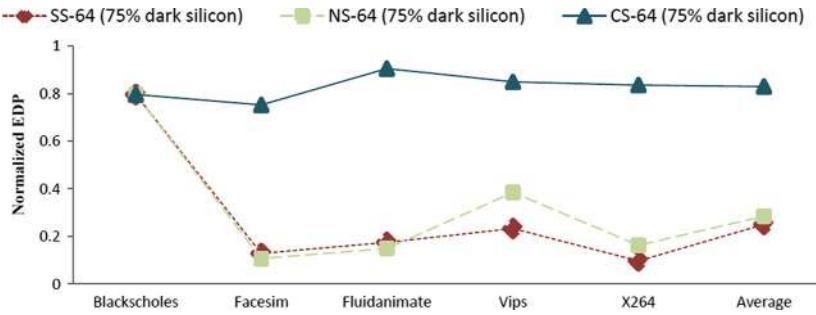

Fig. 12 EDP comparison between different architectures.

2.4.3 Thermal Analysis

Fig. 13 demonstrates the thermal analysis of SS under X264 workload for 36-core NoC (55% dark silicon) after one, two, and three consecutive sprinting. Since SS uses simultaneous techniques of core sprinting, application migration, and power gating to distribute the heat across the chip, it can efficiently avoid hot spots in the system. The peak temperature is 322.8 K and the average temperature of the system is 298.9, 304.3, and 312.5 K after one, two, and three consecutive sprinting, respectively.

Furthermore, Fig. 14 displays the thermal analysis of different architectures under X264 workload for 36-core NoC (55% dark silicon) after four consecutive sprinting. As shown in Fig. 14A, CS results in a hot spot in the center of the chip. Moreover, since thermal-aware floor planning of NS tries to physically separate logical connected cores, heat is distributed to the corners of the chip as depicted in Fig. 14B. Such floor-planning proposal has three disadvantages: First, it requires additional overheads at design stage; second, it is highly application specific and is not suitable for dynamic workloads; third, it leads to performance degradation due to long-distance communications between physically separated cores.

On the other hand, as shown in Fig. 14C, SS outperforms the other two architectures to efficiently distribute the heat across the chip. First, it does not require any temperature-aware floor planning; second, it does not rely on specific running applications to avoid hot spots; third, there is no need to change the physical positions of the cores.

2.4.4 Reliability Assessment

The central hot spot in CS and the angular hot spots in NS greatly increase the failure probability (both temporary and permanent) of those highly active cores due to frequent phase change of the core internal materials.

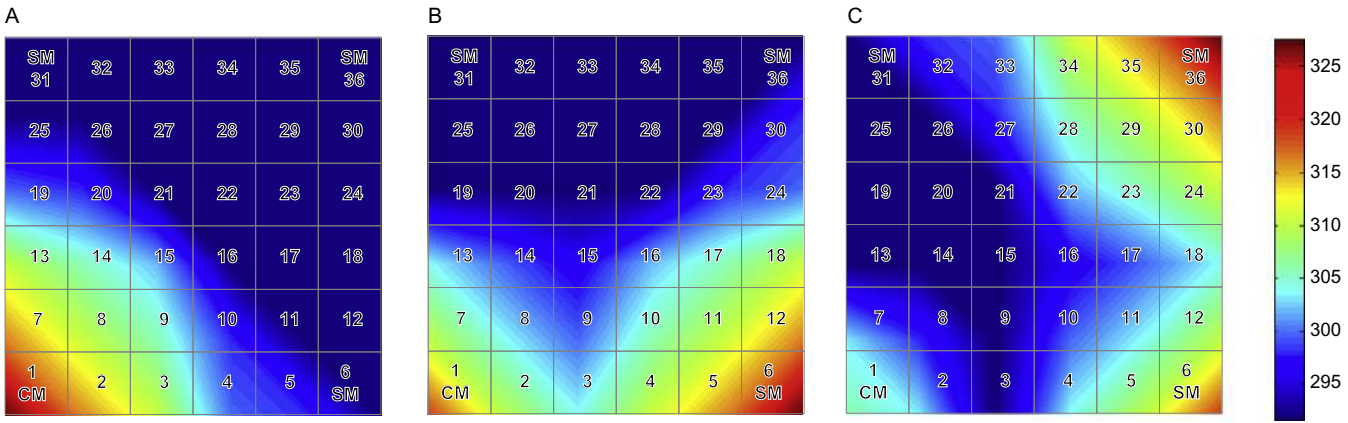

Fig. 13 Thermal distribution in SS-36 (35% dark silicon) under X264 workload after (A) one, (B) two, and (C) three consecutive sprinting.

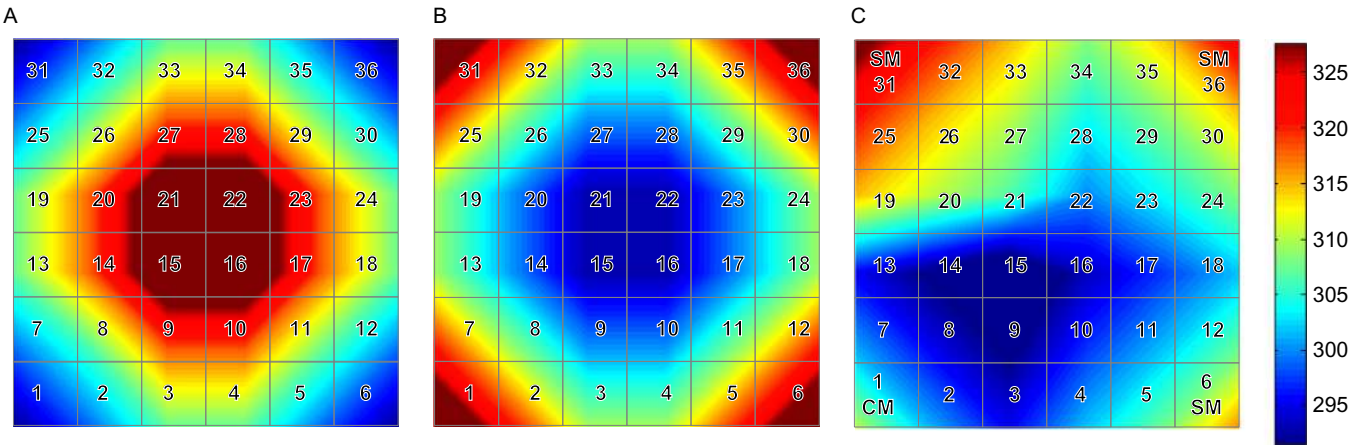

Fig. 14 Thermal distribution comparison between different architectures under X264 workload (A) CC-36, (B) NC-36, and (C) SS-36 (55% dark silicon).

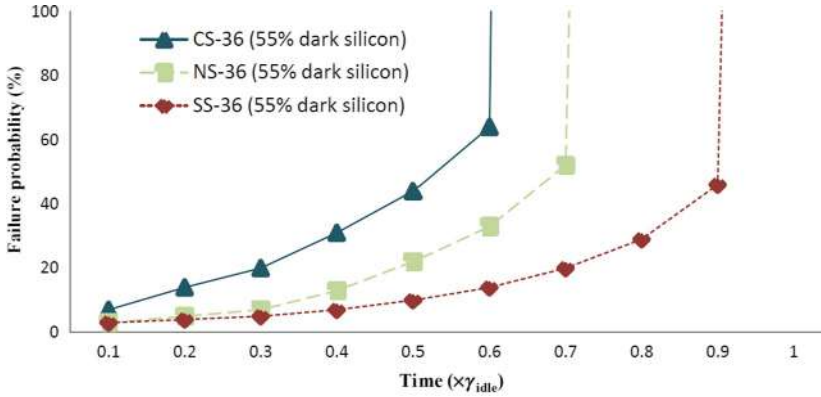

Fig. 15 Reliability comparison between different architectures.

Fig. 15 demonstrates the failure probability of different architectures over time under X264 workload for 36-core NoC (55% dark silicon) under the injection rate of 0.5 λ_{full}. It is assumed that failure of a single core leads to the whole system failure. It can be seen that fair core unitization in SS not only efficiently distributes the heat across the chip but also decreases the possibility of failure of each core which ultimately increases the reliability of the system. In other words, in SS the cores are aging almost evenly. On the contrary, there are some central aged cores in CS as well as some cornered aged ones in NS through time, while the others are still young. The aged cores are increasingly subjected to failure than the young ones. This fact makes the requirement of fault-tolerant mechanisms inevitable in CS and NS.

2.5 Summary

Of all the challenges the mobile device industry faces, keeping components cool is the most important, since overheating causes significant reductions in the operating life of a device and leads to device failure. Moreover, QoS of real-time streaming applications will become more and more important for future generations of mobile devices. On the other hand, due to the dark silicon problem, the threshold voltage cannot be scaled without exponentially increasing leakage, and as a result, the operating voltage should be kept roughly constant.

Therefore, in this section, a fine-grained NoC-based MCSoC architecture, called SS, along with core behavior model, system topology, application migration scheme, and controlling mechanism was introduced in order to handle high-performance QoS-aware mobile demands by reliably utilizing

	Computational sprinting	NoC sprinting	Shift sprinting
Performance	High performance in short-burst computation demands	High performance in short-burst computation demands	High performance in media streaming demands
Power consumption	High-power consumption	Low-power consumption	Low-power consumption
Energy delay product	High EDP	Low EDP	Low EDP
Temperature	Central hot spot	Angular hot spots	No hot spot
Reliability	Low reliability	Average reliability	High reliability
Challenges	On-chip regulators	On-chip regulators, application dependency, floor-planning overhead	On-chip regulators, application migration overhead

Fig. 16 Comparison summary.

dark silicon. Simulation results reported meaningful gain in performance, temperature, and reliability of the system compared to state-of-the-art works. Fig. 16 represents a comparison summary between SS, CS, and NS. As it can be seen from Fig. 16, SS is a promising architecture for future MCSoC mobile devices capable of providing QoS media streaming demands.

3. ROUND ROTARY MAPPING: TEMPERATURE- AND CONGESTION-AWARE APPLICATION MAPPING APPROACH FOR WIRELESS NoC IN DARK SILICON AGE

HWNoC provides high-bandwidth, low-latency, and flexible topology configurations, making this emerging technology a scalable communication fabric for future MCSoCs. However, high energy costs of the WRs in comparison with the conventional routers not only limits the integration of WRs on a single chip but also introduces a direct confrontation with the utilization wall. On the other hand, by employing limited number of WRs, they are more vulnerable to congestion since far apart traffics intend to utilize wireless express links which result in high wireless channel competitions. Moreover, as it is mentioned in previous section, keeping future MCSoCs—with hundreds of embedded cores—cool has a high priority. Thus, in this section, we introduce a temperature- and congestion-aware application mapping algorithm, named Round Rotary Mapping (RRM) [19], targeted at tackling two critical concerns in HWNoC-based MCSoCs in dark silicon age: Alleviation of the severe congestion on WRs and prevention of persistent hot spots in the network.

3.1 Preliminaries and Motivations

A 4 × 4 HWNoC-based SoC (Fig. 17A) with two WRs is simulated using random task mapping (Fig. 17B) and congestion-aware dynamic task mapping (Fig. 17C) presented in Ref. [31] to show the necessity of both congestion avoidance and hot spot prevention. Several applications each with two to five tasks are randomly generated. Each application is considered to have 75% intracommunication among its tasks and 25% intercommunication with the tasks of other applications. Applications are scheduled based on the First-Come-First-Serve (FCFS) policy and the maximum traffic injection rate is λ_{full}. An allocation request for the scheduled application is sent to the CM of the system. CM keeps track of the free cores to map the new tasks. Moreover, each core sends its status to the CM periodically.

3.1.1 Congestion

Fig. 18 shows the average network latency comparison between random and dynamic task mapping schemes in 4 × 4 HWNoC-based SoC. In random task mapping, the traffic is not evenly distributed in the network, resulting in high congestion surrounding WRs which degrades the network performance significantly. On the other hand, in dynamic task mapping, the traffic is more balanced globally, resulting in great average latency reduction.

3.1.2 Hot Spot

The thermal analysis of 4 × 4 HWNoC-based SoC with random and dynamic task mapping in 0.5 λ_{full} is depicted in Fig. 19. In random task mapping, the hot spot regions are observed around the WRs since most of the traffics are moving toward them. With dynamic task mapping, although the congestion around WRs is decreased, the hot spot problem is getting worse around the CM because the applications are contiguously mapped as close as possible to the CM.

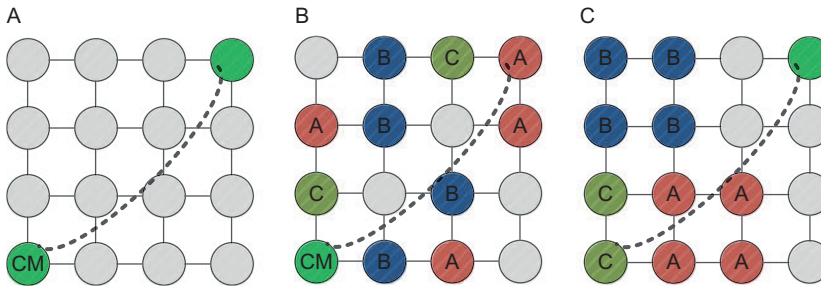

Fig. 17 (A) 4 × 4 HWNoC-based SoC with two WRs using (B) random task mapping and (C) dynamic task mapping.

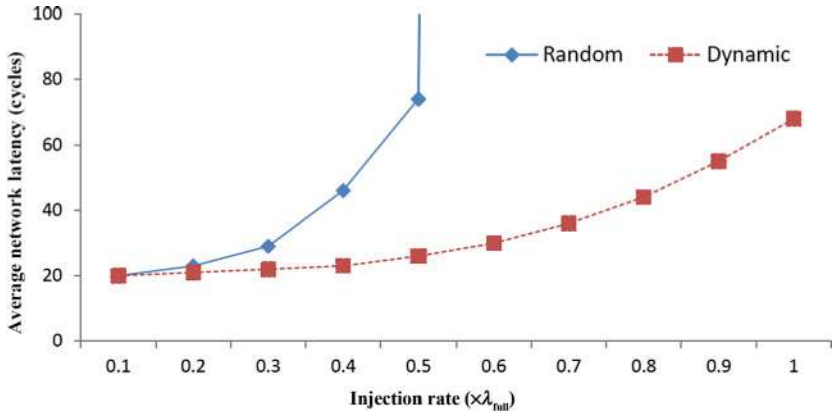

Fig. 18 Average network latency in 4 × 4 HWNoC-based SoC with random and dynamic task mapping.

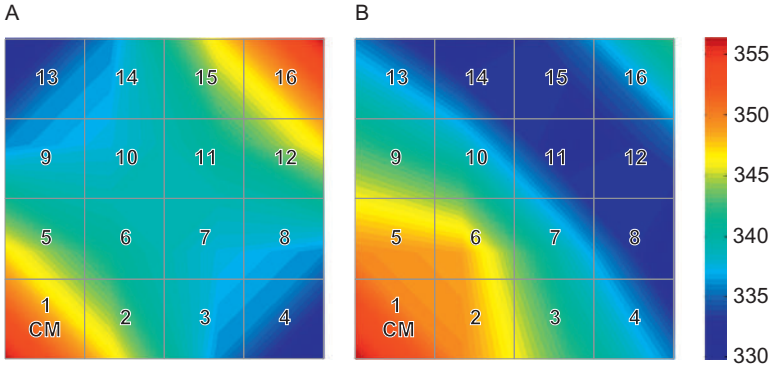

Fig. 19 Thermal distribution for 4 × 4 HWNoC-based SoC (A) random task mapping and (B) dynamic task mapping.

3.2 RRM Algorithm

Based on the above discussions, a reliable mapping is essential for HWNoC-based MCSoC that not only improves network performance by reducing severe congestion around WRs, but rather achieves energy efficiency by preventing hot spots in the network. In other words, we may combine the performance gain obtained by dynamic task mapping shown in Fig. 18 with a temperature-aware method to avoid hot spots depicted in Fig. 19.

3.2.1 System Configuration

Without loss of generality, a 2D mesh HWNoC is virtually divided into several regions where the number of regions equals to the number of available WRs.

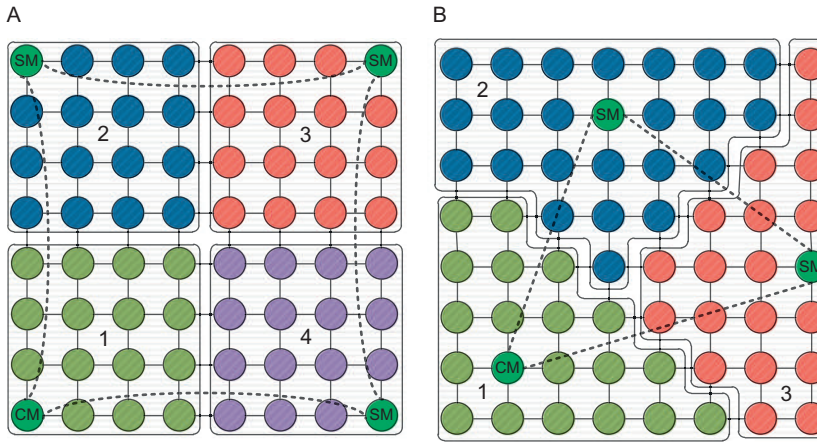

Fig. 20 8×8 HWNoC-based MCSoC (A) four regions and (B) three regions.

The WRs are interconnected to form a wireless highway if they fall within each other transmission range. Thus, in each dedicated region only one WR exists and the WR is associated to serve as the access point to the highway. For network efficiency, HWNoC is partitioned in a way that any core within a region has the minimum hop count toward the WR of that region than the WRs of the other regions. For borderline cases that a core may have the same hop count from two or more WRs, the core will be randomly assigned to one of the candidate regions. Fig. 20 shows two 64-core HWNoC-based MCSoCs with four and three regions.

Moreover, regardless of the number of regions, four Cartesian coordinate systems are defined as Down-Left (DL), Top-Left (TL), Top-Right (TR), and Down-Right (DR) shown in Fig. 21. Origin of each coordinate system is one of the four corners of the network. At each moment, there is one active region (*active_R*) along with one active coordinate system (*active_C*). Furthermore, the WRs are equipped with the control logic to manage the application mapping within their regions. One of the WRs is assigned as CM and the other WRs are named Regional Managers (RMs). Since each manager is responsible to assign the tasks on its own region, the hierarchical managing scheme helps balance the workload distribution between different managers.

3.2.2 Application Representation

In many embedded system applications like robotics, biomedical systems, and multimedia control systems, not only the tasks of each application are able to

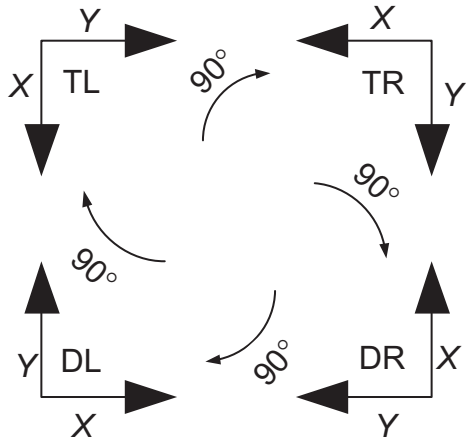

Fig. 21 Cartesian coordinate systems.

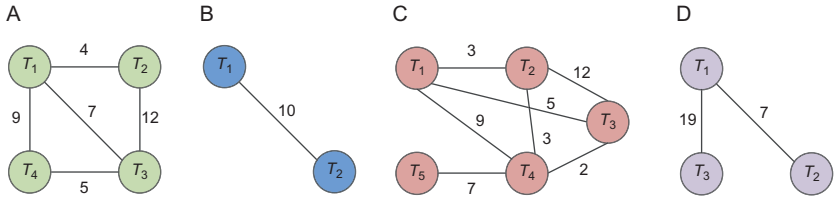

Fig. 22 Task graphs of different applications.

communicate with each other (i.e., intracommunication), but also multiple applications may also communicate with each other (i.e., intercommunication).

Intracommunication: An undirected graph, naming Task Graph (TG), represents each application and intracommunication between its tasks in the system. Each vertex denotes one task of the application, while each edge stands for communication between each two tasks as given in Eq. (7). The TGs of four applications each with two to five tasks are shown in Fig. 22. The amount of data transferred between any two tasks is indicated on the edge.

$$\forall t_i \in T, \forall e_{i,j} \in E, app = \mathrm{TG}(T, E) \tag{7}$$

Intercommunication: Since the applications are considered to enter the system at runtime, no static graph can be defined for intercommunication between different applications. An incoming application may request to communicate with an already mapped application. Moreover, an existing application in the system may ask to communicate with a newly mapped one or the one

which is not yet mapped onto the system. Thus, the intercommunication graph between different applications is highly dynamic.

3.2.3 Mapping Algorithm

RRM tries to map incoming applications region by region in a round-robin manner to balance the thermal distribution globally while periodically rotating the Cartesian coordinate system to balance the thermal distribution within each region locally. On the other hand, the tasks of each application are mapped with regard to the minimum Hop-Count Contiguity (HCC) in order to reduce congestion caused by long-distance communications of the same tasks of each application.

Fig. 23 represents the RRM algorithm. In *Initialization*() function, the number of regions (i.e., r), set of regions (i.e., $R = \{R_1, R_2, \ldots, R_r\}$), and set of coordinate systems (i.e., $C = \{\text{DL}, \text{TL}, \text{TR}, \text{DR}\}$) are initialized. Moreover, the active region and active coordinate system are also initialized to the first element of each set (i.e., *active_R* $= R_1$ and *active_C* $=$ DL). Then, applications are chosen based on the FCFS policy since no background information is considered about incoming applications. In case of having background

```
r: Number of regions (i.e., number of WRs)
C: Set of coordinate systems = {DL, TL, TR, DR}
R: Set of regions = {R1, R2,..., Rr}
A: Set of applications
active_C: Active coordinate system
active_R: Active region

Initialization() while true do
    if A is empty then
        Sleep()
    app = Choose an application from A
    free_Y = Choose the set of free cores with the smallest Y
    free_XY = Choose the free core with the smallest X from free_Y
    ft = FirstTask(app, free_XY)
    Map(ft, free_XY)
    ConMap(app, free_XY)
    if active_R == Rr then
        active_C = Choose the next coordinate system from C
    active_R = Choose the next region from R
```

Fig. 23 RRM algorithm.

information about the incoming applications, an appropriate application selection policy can be applied which is beyond the scope of this chapter.

In each region, RRM first tries to find a set of free cores with the smallest "*Y*"s. Then among them, the core with the smallest "*X*" is chosen in order to map the first task of the application. The first task of each application is returned from the *FirstTask*(*app*,*free_XY*) function. This function returns the task of the selected application (i.e., *app*) with the equal or smaller number of edges than the available free cores around the chosen core (i.e., *free_XY*). If there is more than one task with the aforementioned criteria, then the first task would be the one with the most intensive communication among the candidates. If there is no task with the equal or smaller edges as the available free cores around *free_XY*, the task with the least intensive communication among all the tasks is chosen. In the case of existing two or more candidates with the same characteristics, one of them is randomly chosen.

Fig. 24 shows the first task selection for different values of available free cores around *free_XY* for the four applications of Fig. 22. For example, in application A (i.e., Fig. 22A) if the number of available free cores around *free_XY* is "1," there is no task with the equal or smaller number of edges as "1"; thus, the task with the least intensive communication among all the tasks of application A (i.e., T_4) is selected. However, if the available free cores are "2," two candidates (i.e., T_2 and T_4) have equal or smaller number of edges than "2"; among them, T_2 is selected because it has more intensive communications than T_4 (i.e., 16 vs 14). If the number of available cores is equal or greater than "3," the task with the most intensive communications (i.e., T_3) will be chosen. Note that in a mesh-based NoC the maximum available cores around each core are eight. Also in each region, only the free cores within that region are considered. The first task selection helps choose the most suitable central task of the selected application (i.e., *app*) to be mapped into the chosen free core of the system (i.e., *free_XY*) according to minimum HCC mapping.

After mapping the first task to *free_XY*, RRM tries to map the other tasks of the application based on minimum HCC around the first task within that

Number of free cores	0	1	2	3	4	5	6	7	8
Application A: Fig. 22A	T_4	T_4	T_2	T_3	T_3	T_3	T_3	T_3	T_3
Application B: Fig. 22B	T_1/T_2	T_1/T_2	T_1/T_2	T_1/T_2	T_1/T_2	T_1/T_2	T_1/T_2	T_1/T_2	T_1/T_2
Application C: Fig. 22C	T_5	T_5	T_5	T_3	T_4	T_4	T_4	T_4	T_4
Application D: Fig. 22D	T_2	T_3	T_1	T_1	T_1	T_1	T_1	T_1	T_1

Fig. 24 First task selection example.

region. Minimum HCC is defined as minimum overall hop count between all the cores in which the application is mapped into. In the case that the application does not fit into the current region (i.e., *active_R*), the current region will be merged with the next region temporarily. After mapping of each application, the active region is shifted to the next region to balance the thermal distribution globally. Moreover, after a complete round (i.e., all the regions become active once in one coordinate system) the origin of the coordinate system is rotated to the next origin to balance the thermal distribution within each region as well. Note that when the RRM algorithm reaches the last element of R (or C), it starts from the beginning again, i.e., *active_R* is set to R_1 (or *active_C* is set to DL). In the case that there is no available application to be mapped into the system, RRM goes to the *Sleep*() mode until a new application arrives and signals to wake up.

Overall, RRM tries to map the task of each application as contiguous as possible based on minimum HCC to avoid long-distance communications that mostly influence the WRs. Also, it tries to spread the temperature across the chip by periodically changing the regions (i.e., global heat distribution) and coordinate systems (i.e., local heat distribution).

3.2.4 Step-by-Step Examples

In order to have a better understanding of RRM algorithm, two step-by-step examples are discussed as follows. In each example the four applications of Fig. 22 as well as four more applications are considered to be mapped into the system. Fig. 25 represents a visual example of the RRM algorithm in a region-based 64-core HWNoC with four symmetric regions. In Fig. 25A application A (i.e., Fig. 22A) with four tasks arrives. Since the first region is active and the active coordinate is DL (i.e., down-left), RRM chooses the core $(0,0)$ as *free_XY* because it is the core with the smallest Y and then the smallest X in the first region based on DL coordinate system. Since the number of available free cores around the core $(0,0)$ is "3," T_3 is chosen as the first task of application A based on Fig. 24. Then T_3 is assigned to core $(0,0)$. Afterward, the other three tasks of application A are mapped as contiguous as possible to T_3. Now the active region is shifted to the second one.

In Fig. 25B application B (i.e., Fig. 22B) with two tasks arrives. The second region is active and the active coordinate is still DL. Thus, RRM chooses the core $(0,4)$ as *free_XY* because it is the core with the smallest Y and then the smallest X in the second region with respect to DL coordinate system. Then, either T_1 or T_2 is selected as the first task of application B according to Fig. 24. Since application B has only one more task, it can be mapped at core $(1,4)$ or core $(0,5)$ with respect to contiguity. In these cases

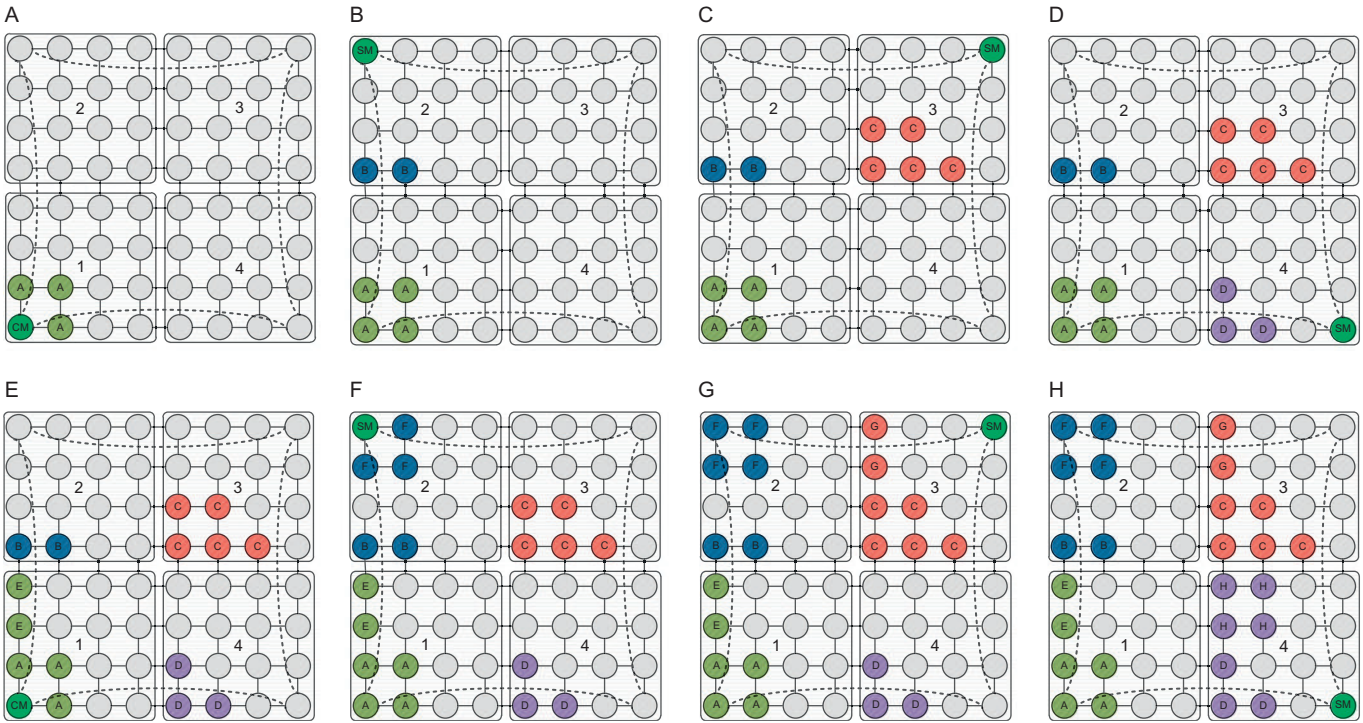

Fig. 25 RRM example in 8×8 HWNoC-based MCSoC with four symmetric regions.

one of the cores is randomly chosen as the target. Then the active region is changed to the third one.

In Figs. 24D and 25C applications C (i.e., Fig. 22C) and D (i.e., Fig. 22D) are mapped. Then, because the active region is reached to the last region in the first round, the coordinate system is rotated to TL (i.e., top-left) and the active region is again shifted to the first one. Now, in Fig. 25E for application E with four tasks the core $(4, 0)$ of new coordinate system (i.e., TL) is chosen as *Free_XY*. The procedure continues for the applications F, G, and H.

Moreover, Fig. 26 shows a visual example of RRM algorithm in a region-based 64-core HWNoC-based MCSoC with three asymmetric regions. The RRM works exactly the same for both symmetric and asymmetric regions. However, more attention is required to following up the step-by-step procedure in asymmetric regions. At first, in Fig. 26A application A (i.e., Fig. 22A) with four tasks arrives. Since the first region is active and the active coordinate is DL, RRM chooses the core $(0, 0)$ as *free_XY*. Since the number of available free cores around the core $(0, 0)$ is "3," T_3 is chosen as the first task of application A based on Fig. 24. Then T_3 is assigned to core $(0, 0)$. After mapping the other tasks of application A as contiguous as possible to T_3, the active region is shifted to the second one.

In Fig. 26B application B (i.e., Fig. 22B) with two tasks arrives. Since the second region is active and the active coordinate is still DL, RRM chooses the core $(3, 3)$ because it is the core with the smallest Y and then the smallest X in the second region based on the DL coordinate system. Then, either T_1 or T_2 is selected as the first task of application B according to Fig. 24. Since application B has only one more task, it can be only mapped at the core $(3, 4)$ with respect to the minimum HCC. Then the active region is shifted to the third one.

In Fig. 26C application C (i.e., Fig. 22C) with five tasks arrives and core $(6, 0)$ is selected as *free_XY*. Since the number of available free cores around the core $(6, 0)$ is "4," T_4 is chosen as the first task of application C based on Fig. 24.

After mapping application C, since all the regions are met in the first round, the coordinate system is rotated to TL. Applications D, E, and F are mapped based on the new coordinate system to the first, second, and third regions, respectively. Next, because all the regions are met again, the coordinate system is rotated to TR (i.e., top-right). Now, as shown in Fig. 26G for application G with two tasks, the core $(5, 4)$ is chosen as *free_XY* because it has the smallest Y and then the smallest X among all the free cores in the first region with respect to the TR coordinate system. The active region is then shifted to the second one and the procedure is repeated for application H.

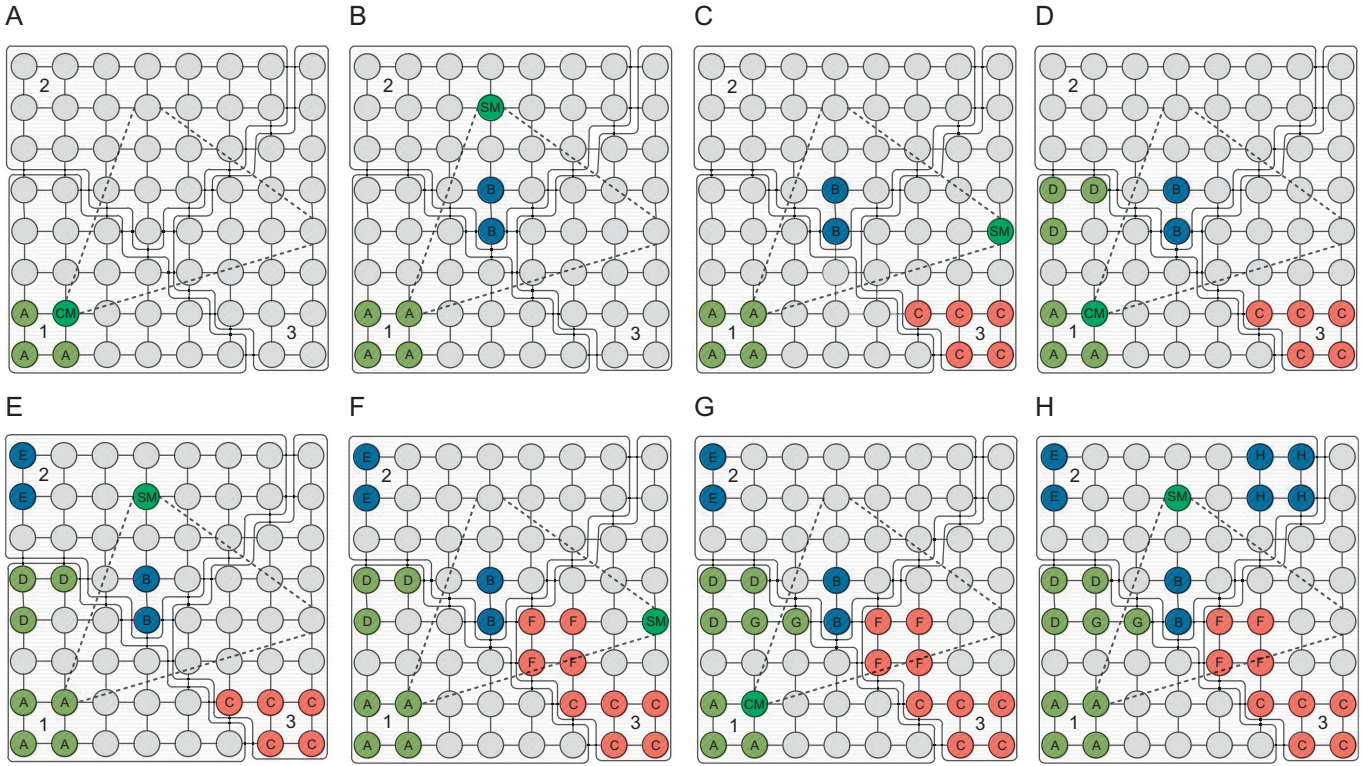

Fig. 26 RRM example in 8×8 HWNoC-based MCSoC with three asymmetric regions.

3.3 Methodology

The basic experimental setups are the same as Section 2.3. In addition, several sets of applications each with 2–5 tasks are generated using TGG [32] where the amount of data transferred from the source task to the destination task are randomly distributed between 2 and 36 flits of data. Each application is considered 75% intracommunication and 25% intercommunication between the other applications. The intercommunication between different applications is conducted by the first task of each application mapped to the system.

Applications are scheduled based on the FCFS policy and the maximum possible scheduling rate is λ_{full}. An allocation request for the scheduled application is sent to the CM of the system. CM then based on the active region sends the information to the responsible RM through the hierarchical managing network. The hierarchical *XY* routing algorithm taken from Ref. [23] is implemented in which intraregion communications are handled through wired path and interregion communications are supported via both wired and wireless paths. Two 64-core HWNoC (52% dark silicon) with three and four WRs (Fig. 20) are considered in the simulations. Comparisons are also made between RRM and random task mapping as baseline in addition to the congestion-aware dynamic task mapping algorithm (DMA) presented in Ref. [31].

3.4 Experimental Results

3.4.1 Hop Counts and Energy Saving

As shown in Refs. [33,34], decreasing Manhattan distance (MD) between tasks of application edges is an effective way to minimize the communication energy consumption of the applications. The percentage of packets that are delivered over different path lengths (i.e., MD) is illustrated in Fig. 27. The experiments have been run for different algorithms in the injection rate of $0.5\,\lambda_{\text{full}}$. As can be seen, more than 60% (and more than 50%) of the packets are delivered by one-hop distance using RRM algorithm in 64-core HWNoC with four regions (and three regions).

Accordingly, Fig. 28 represents the average MD for different algorithms in the injection rate of $0.5\,\lambda_{\text{full}}$ based on the percentage of the intracommunication and intercommunication. By decreasing the percentage of intercommunication between applications, more energy can be saved by RRM, since it maps all the tasks of each application based on minimum HCC. As can be seen RRM outperforms DMA in less than 5% intercommunication between different applications.

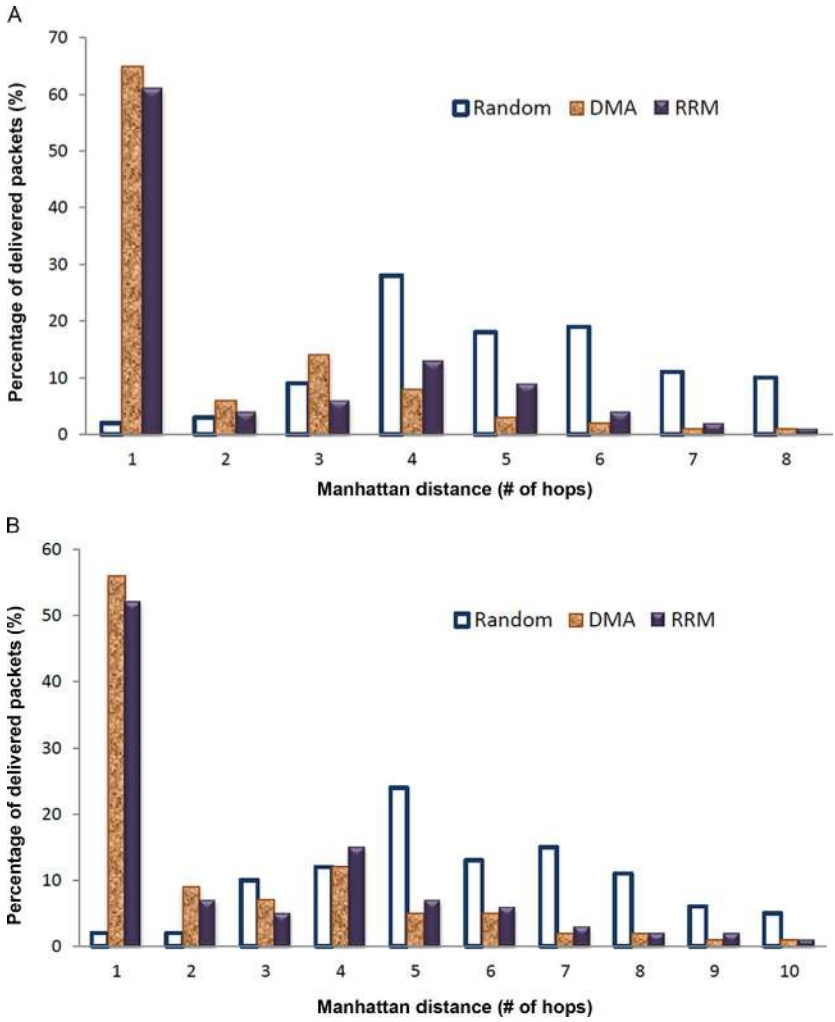

Fig. 27 Percentage of delivered packets in different path lengths in 64-core HWNoC (A) four regions and (B) three regions.

Intracommunication (%)/intercommunication (%)		70/30	75/25	80/20	85/15	90/10	95/5	100/0
64-core HWNoC with four regions	Random	5.01	5.08	5.21	5.36	5.14	5.2	5.07
	DMA	2.34	1.93	1.61	1.31	1.13	0.93	0.78
	RRM	2.78	2.3	1.84	1.55	1.27	0.92	0.66
64-core HWNoC with three regions	Random	5.61	5.79	5.6	5.87	6.03	5.92	5.72
	DMA	2.98	2.47	2.07	1.68	1.4	1.16	0.96
	RRM	3.21	2.77	2.2	1.82	1.49	1.09	0.78

Fig. 28 Average MD comparison.

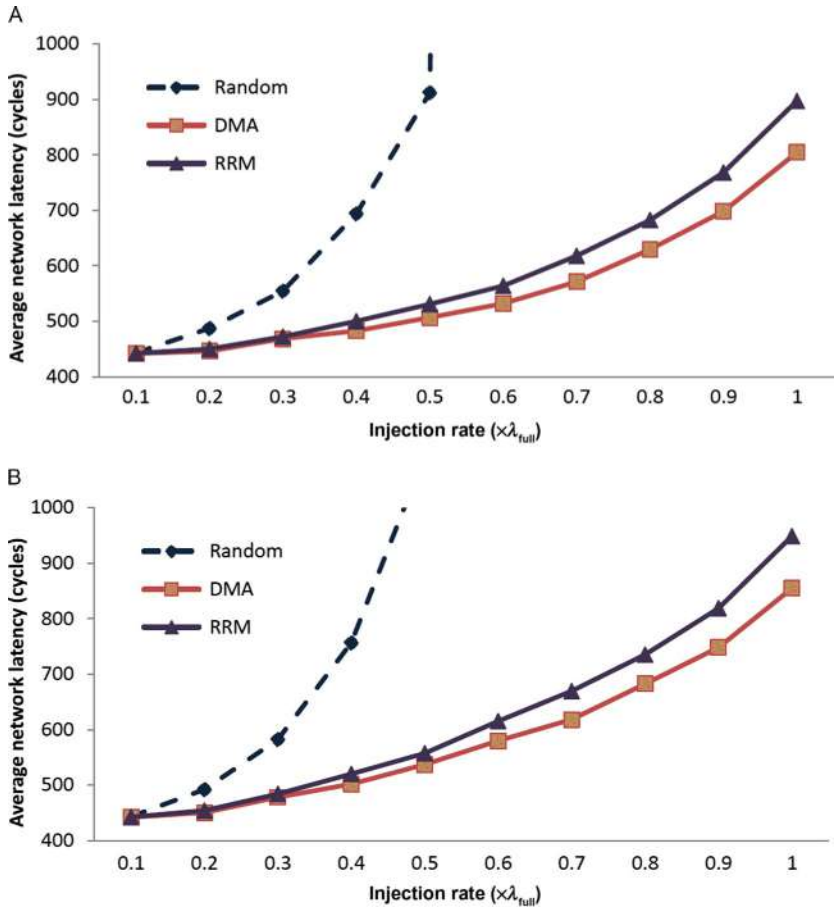

Fig. 29 Average network latency in 64-core HWNoC (A) four regions and (B) three regions.

3.4.2 Network Latency and Congestion Avoidance

Fig. 29 shows the average network latency for different algorithms. It is supposed that there is no gap between application arrivals. As can be seen, RRM has a reasonable average network latency next to DMA. Contiguous mapping of each application tasks in both DMA and RRM results in lower network latency than random mapping. Also, in DMA and RRM, by increasing the injection rate, the network becomes uniformly congested because the usage of WRs is more balanced.

It can be concluded from Figs. 27 and 29 that employing hierarchical XY routing that uses wired paths for short-distance communications combined

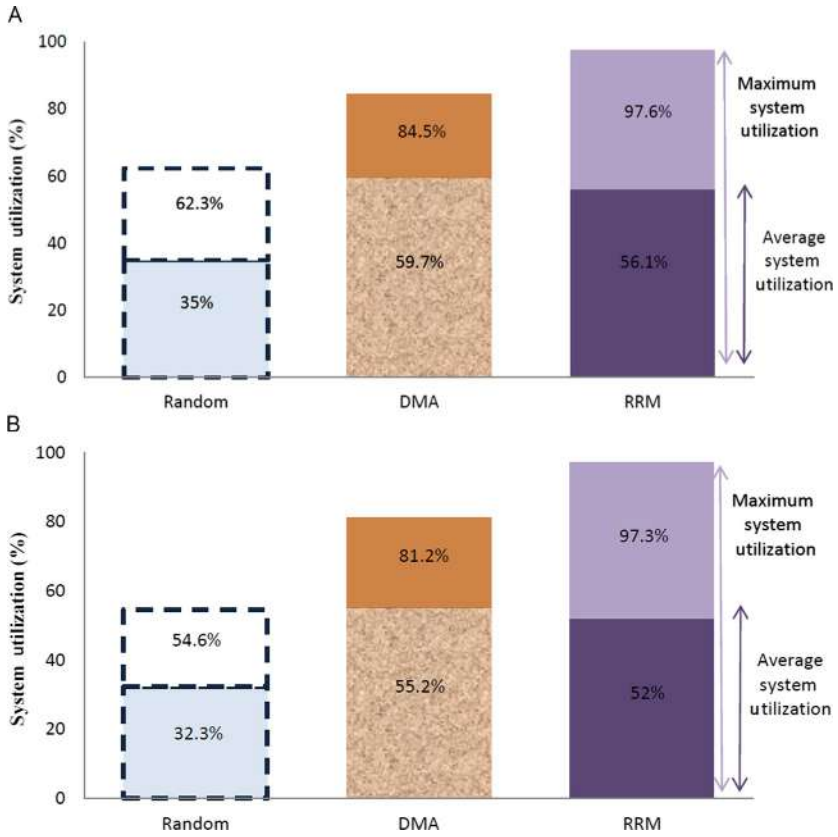

Fig. 30 System utilization in 64-core HWNoC (A) four regions and (B) three regions.

with RRM can dynamically reduce the congestion over WRs while minimized the communication energy consumption.

3.4.3 System Utilization

System utilization is another parameter that has been analyzed among the different algorithms. As shown in Fig. 30, RRM has lower average system utilization than DMA mapping but has better maximum system utilization that is defined as the highest percentage of the utilization during the simulation time. Note that the system utilization is based on the number of tasks can be mapped on nondark cores (i.e., 48% of the cores in 64-core HWNoC)

which communicate with each other without dropping due to the high congestion.

This happens because DMA tries to map not only all the tasks of each application but also all the applications contiguous (i.e., close to CM) that results in better average system utilization. On the other hand, RRM unlike DMA does not suffer from area fragmentation and can reach higher maximum utilization (i.e., almost 98%).

3.4.4 Thermal Analysis

Figs. 31 and 32 demonstrate the thermal distribution of different algorithms in the maximum system utilization based on Fig. 30. Furthermore, Fig. 33 shows the average and the peak temperature comparisons. Permanent hot spots in the system greatly increase the failure probability. Since in dark silicon age more than half of the chip is dark, RRM utilizes the dark cores in order to efficiently avoid hot spots in the system. On the other hand, DMA suffers from severe hot spot around CM and random mapping has multiple hot spots around WRs. Moreover, unlike RRM, the peak temperature is gotten worse in DMA by decreasing the number of WRs.

3.5 Summary

In this section, a temperature- and congestion-aware task mapping algorithm named RRM was introduced in order to solve some of the key concerns in future HWNoC-based MCSoCs. Simulation results showed significant improvement in both congestion and temperature control of the system. More than 60% (and more than 50%) of the packets are delivered by one-hop distance using RRM algorithm in 64-core HWNoC with four regions (and three regions); this saves communication energy consumption significantly. RRM also has a reasonable average network latency. Contiguous mapping of each application tasks results in lower network latency and finally total execution time gain. Overall, RRM provides more than 50% system utilization and about 98% maximum system utilization. Moreover, the heat is distributed evenly across the whole chip using RRM algorithm. The peak and average temperature are less than 352 and 330 K, respectively, in 64-core HWNoC. Fig. 34 shows the comparison summary of different mapping algorithms.

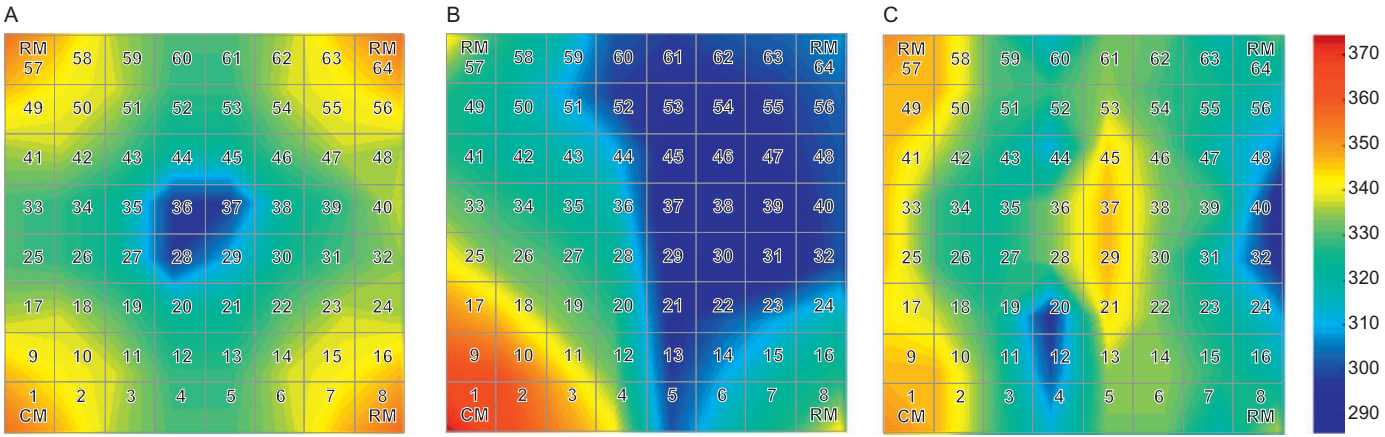

Fig. 31 Thermal distribution comparison in 64-core HWNoC with four regions (A) random, (B) DMA, and (C) RRM.

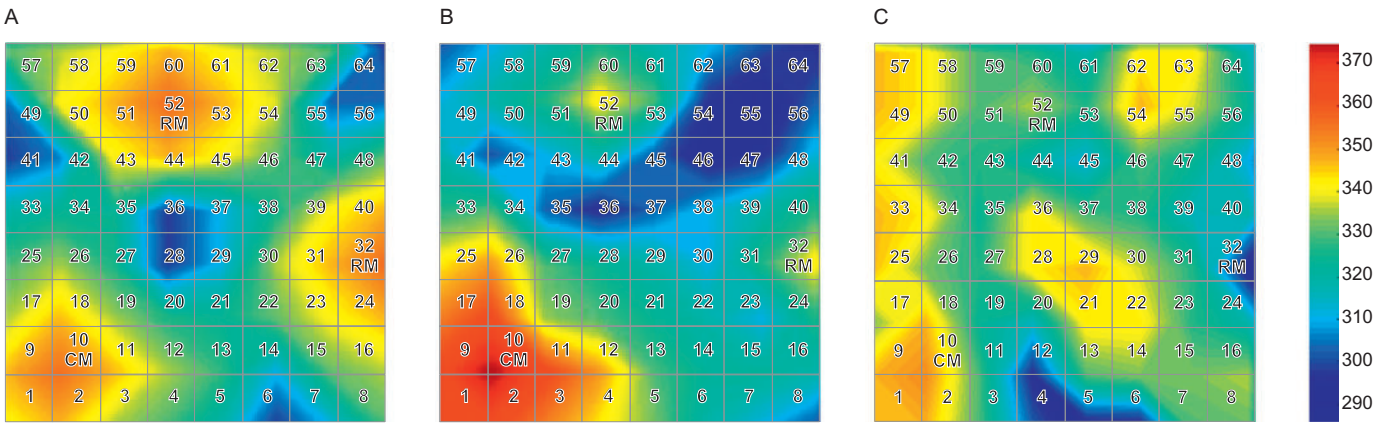

Fig. 32 Thermal distribution comparison in 64-core HWNoC with three regions (A) random, (B) DMA, and (C) RRM.

		Random	DMA	RRM
64-core HWNoC with four regions	Average temperature (K)	333.6	318.1	329.6
	Peak temperature (K)	356.4	372.8	351.7
64-core HWNoC with three regions	Average temperature (K)	330.1	323.5	329.9
	Peak temperature (K)	357.1	375.2	351.4

Fig. 33 Average and peak temperature comparison.

	Random mapping	Dynamic mapping algorithm	Round rotary mapping
Energy consumption	High communication energy consumption	Low communication energy consumption	Low communication energy consumption
Execution time	High network latency	Low network latency	Low network latency
System utilization	Medium system utilization	High average system utilization	High average and maximum system utilization
Temperature	Hot spots around WRs	Intensive hot spot around CM	No hot spot

Fig. 34 Comparison summary.

4. CONCLUSION AND THE FUTURE OUTLOOK

Due to the utilization wall problem, the threshold voltage cannot be scaled without exponentially increasing leakage, and as a result, the operating voltage should be kept roughly constant. This is an exponentially worsening problem that accumulates with each process generation [35]. Recent studies [1] have predicted that, on average, 52% of a chip's area will stay dark for the 8-nm technology node.

Based on the multiobjective nature of MCSoCs in dark silicon age, the many-core system should be capable of observing and automatically adapting its behavior and recourses in order to accomplish its assigned job efficiently. On-chip self-awareness is a key enabling technology for efficient use of heterogeneous architectures and applications with guaranteed runtime system objectives [36].

Moreover, we are expecting billions of devices connected to the Internet of Things in the near future. Manual management to reach multiobjective goals of these devices will soon become impossible. Thus, we need to bring

learning-based methods and control theory into the game as well. As a result, the technology trend in embedded system design is not only shifting from single-objective homogenous designs to multiobjective heterogeneous ones but also inevitably changing from manually tune-up approaches to self-aware platforms.

REFERENCES

[1] J. Henkel, H. Khdr, S. Pagani, M. Shafique, New trends in dark silicon, in: ACM/EDAC/IEEE Design Automation Conference (DAC), 2015. Article 119.

[2] N. Goulding-Hotta, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, P.-C. Huang, M. Arora, S. Nath, V. Bhatt, J. Babb, S. Swanson, M. Taylor, The GreenDroid mobile application processor: an architecture for silicon's dark future, IEEE Micro 31 (2) (2011) 86–95.

[3] N. Hardavellas, M. Ferdman, B. Falsafi, A. Ailamaki, Toward dark silicon in servers, IEEE Micro 31 (4) (2011) 6–15.

[4] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K.P. Pipe, T.F. Wenisch, M.M.K. Martin, Computational sprinting, in: IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2012, pp. 1–12.

[5] L. Benini, G. De Micheli, Networks on chips: a new SoC paradigm, IEEE Comput. 35 (1) (2002) 70–78.

[6] J. Zhan, Y. Xie, G. Sun, NoC-sprinting: interconnect for fine-grained sprinting in the dark silicon era, in: ACM/EDAC/IEEE Design Automation Conference (DAC), 2014. Article 160.

[7] A. Rezaei, D. Zhao, M. Daneshtalab, H. Wu, Shift sprinting: fine-grained temperature-aware NoC-based MCSoC architecture in dark silicon age, in: ACM/EDAC/IEEE Design Automation Conference (DAC), 2016. Article 155.

[8] Y. Zhang, L. Peng, F. Xin, H. Yue, Lighting the dark silicon by exploiting heterogeneity on future processors, in: ACM/EDAC/IEEE Design Automation Conference (DAC), 2013. Article 82.

[9] B. Raghunathan, Y. Turakhia, S. Garg, D. Marculescu, Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013, pp. 39–44.

[10] ITRS, International Technology Roadmap for Semiconductors, 2013. edition.

[11] M. Shafique, S. Garg, J. Henkel, D. Marculescu, The EDA challenges in the dark silicon era: temperature, reliability, and variability perspectives, in: ACM/EDAC/IEEE Design Automation Conference (DAC), 2014. Article 185.

[12] S. Pagani, H. Khdr, W. Munawar, J.-J. Chen, M. Shafique, M. Li, J. Henkel, TSP: thermal safe power: efficient power budgeting for many-core systems in dark silicon, in: International Conference on Hardware/Software Codesign and System Synthesis (CODES), 2014. Article 10.

[13] J.M. Allred, S. Roy, K. Chakraborty, Long term sustainability of differentially reliable systems in the dark silicon era, in: International Conference on Computer Design (ICCD), 2013, pp. 70–77.

[14] B. Raghunathan, S. Garg, Job arrival rate aware scheduling for asymmetric multi-core servers in the dark silicon era, in: International Conference on Hardware/Software Codesign and System Synthesis (CODES), 2014. Article 14.

[15] H. Khdr, S. Pagani, M. Shafique, J. Henkel, Thermal constrained resource management for mixed ILP-TLP workloads in dark silicon chips, in: ACM/EDAC/IEEE Design Automation Conference (DAC), 2015. Article 179.

[16] M. Shafique, D. Gnad, S. Garg, J. Henkel, Variability-aware dark silicon management in on-chip many-core systems, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, pp. 387–392.
[17] N. Kapadia, S. Pasricha, VARSHA: variation and reliability-aware application scheduling with adaptive parallelism in the dark-silicon era, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, pp. 1060–1065.
[18] A. Rezaei, F. Safaei, M. Daneshtalab, H. Tenhunen, HiWA: a hierarchical wireless network-on-chip architecture, in: IEEE International Conference on High Performance Computing & Simulation (HPCS), 2014, pp. 499–505.
[19] A. Rezaei, D. Zhao, M. Daneshtalab, H. Zhou, Multi-objective task mapping approach for wireless NoC in dark silicon age, in: IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP), 2017, pp. 589–592.
[20] H. Esmaeilzadeh, A. Sampson, M.R. ingenburg, L. Ceze, D. Grossman, D. Burger, Addressing dark silicon challenges with disciplined approximate computing, in: Dark Silicon Workshop (DaSi) With International Symposium on Computer Architecture (ISCA), 2012, pp. 1–2.
[21] J. Cong, B. Xiao, Optimization of interconnects between accelerators and shared memories in dark silicon, in: International Conference on Computer-Aided Design (ICCAD), 2013, pp. 630–637.
[22] K. Swaminathan, E. Kultursay, V. Saripalli, V. Narayanan, M.T. Kandemir, S. Datta, Steep-slope devices: from dark to dim silicon, IEEE Micro 33 (5) (2013) 50–59.
[23] A. Rezaei, M. Daneshtalab, F. Safaei, D. Zhao, Hierarchical approach for hybrid wireless network-on-chip in many-core era, Elsevier Int. J. Comput. Electr. Eng. 51 (C) (2016) 225–234.
[24] B. Goodarzi, H. Sarbazi-Azad, Task migration in mesh NoCs over virtual point-to-point connections, in: IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2011, pp. 463–469.
[25] V. Catania, A. Mineo, S. Monteleone, M. Palesi, D. Patti, Noxim: an open, extensible and cycle-accurate network on chip simulator, in: IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), 2015, pp. 162–163.
[26] L. Wang, K. Skadron, Dark vs. Dim Silicon and Near-Threshold Computing Extended Results, Technical Report (UVA-CS-2013-01), Department of Computer Science, University of Virginia, 2013.
[27] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, M. Stan, HotSpot: a compact thermal modeling methodology for early-stage VLSI design, IEEE Trans. Very Large Scale Integr. VLSI Syst. 14 (5) (2006) 501–513.
[28] C. Bienia, S. Kumar, J.P. Singh, K. Li, The PARSEC benchmark suite: characterization and architectural implications, in: International Conference on Parallel Architectures and Compilation Techniques (PACT), 2008, pp. 72–81.
[29] J.A. Butts, G.S. Sohi, A static power model for architects, in: IEEE/ACM International Symposium on Microarchitecture (MICRO), 2000, pp. 191–201.
[30] A. Rezaei, M. Daneshtalab, D. Zhao, M. Modarressi, SAMi: self-aware migration approach for congestion reduction in NoC-based MCSoC, in: IEEE International System-on-Chip Conference (SOCC), 2016.
[31] A. Rezaei, M. Daneshtalab, D. Zhao, F. Safaei, X. Wang, M. Ebrahimi, Dynamic application mapping algorithm for wireless network-on-chip, in: IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP), 2015, pp. 421–424.
[32] Source Forge, Task Graph Generator (TGG), Available: http://sourceforge.net/projects/taskgraphgen/.
[33] C.L. Chou, U.Y. Ogras, R. Marculescu, Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 27 (10) (2008) 1866–1879.

[34] A. Rezaei, M. Daneshtalab, M. Palesi, D. Zhao, Efficient congestion-aware scheme for wireless on-chip networks, in: IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP), 2016, pp. 742–749.
[35] M.B. Taylor, Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse, in: ACM/EDAC/IEEE Design Automation Conference (DAC), 2012, pp. 1131–1136.
[36] N. Dutt, A. Jantsch, S. Sarma, Toward smart embedded systems: a self-aware system-on-chip (SoC) perspective, ACM Trans. Embed. Comput. Syst. 15 (2) (2016). Article 22.

ABOUT THE AUTHORS

Amin Rezaei is currently a Ph.D. candidate in Electrical Engineering and Computer Science at Northwestern University, the USA. He received his B.Sc. degree in Computer Engineering from University of Isfahan, Iran, in 2011 and two M.Sc. degrees one in Computer Engineering from Shahid Beheshti University, Iran, in 2014 and the other in Computer Science from University of Louisiana at Lafayette, the USA, in 2016. His main research interests include Parallel Computing, Dark Silicon, and Logic Encryption.

Masoud Daneshtalab is currently is currently a tenured associate professor at Mälardalen University (MDH) and visiting researcher at Royal Institute of Technology (KTH), Sweden. Before that he was lecturer and project manager at University of Turku in Finland. He has been appointed as Associate Editors of Elsevier Journals of Computers & Electrical Engineering and Microprocessors and Microsystems; and in the Editorial Board of The Scientific World Journal, IJDST, IJARAS, IJERTCS, and IJDATICS. He is also in the Euromicro's board of directors since 2016 while representing Sweden in the management committee of the EU COST Actions IC1202 (TACLe). His research interests include interconnection networks, many-core and reconfigurable systems, and neuromorphic architecture. He has published 1 book, 4 book chapters, and over 200 refereed international journals and conference papers. He is currently TPC member

of different IEEE and ACM conferences, including NOCS, DATE, ASPDAC, ICCAD, ESTIMedia, VLSI Design, ICA3PP, SOCC, VDAT, DSD, PDP, ICESS, Norchip, MCSoC, CADS, EUC, DTIS, NESEA, CASEMANS, NoCArc, MES, HPIN, PACBB, MobileHealth, and JEC-ECC.

Hai Zhou is an associate professor in Electrical Engineering and Computer Science at Northwestern University. He got his Ph.D. degree in Computer Sciences from the University of Texas at Austin, and his B.S. and M.S. degrees in Computer Science and Technology from Tsinghua University in Beijing, China. He was a recipient of a CAREER Award from the National Science Foundation. His research interests include VLSI computer-aided design, algorithm design, and formal methods. He has published more than 150 papers in flagship journals and conferences in these areas.