Power Budgeting of Homogeneous 3-D Processors in Dark Silicon

Hai Wang, *Member, IEEE,* Yanhong Luo, Jincheng Guo, Yu Chen, and Jiming Li

***Abstract*—The performance of 3-D processors is constrained by the power budget, defined as the maximum power consumption to guarantee the system's thermal safety. In this work, we present a dynamic power budgeting method for homogeneous 3-D processors in dark silicon. The new method provides a power budget that leads to higher system potential performance over the existing methods with two new techniques. First, a sub-optimal active core distribution in 3-D space is computed on the fly which increases the total power budget. Second, we introduce an intelligent power assignment strategy across die layers with optimal power ratios computed offline, which further increases the potential system performance. Experiments on two homogeneous 3-D processors with different architectures show that the new method leads to a higher system performance, compared with the existing techniques.**

***Index Terms*—Power budget, homogeneous system, multi-core, 3-D IC, dark silicon.**

I. Introduction

3-D integration has been invented to increase the integration density of integrated circuits (ICs) by stacking multiple silicon layers in the vertical direction [1]. Compared with the traditional 2-D ICs, 3-D ICs have many advantages including higher memory bandwidth, higher computing density, and shorter interconnection delay [2].

3-D processors were invented to utilize the advantages of 3-D ICs [3] and some prototypes have been developed such as the 3D-MAPS CPU [4]. Architecturally, 3-D processors can be classified into two categories: the heterogeneous ones and the homogeneous ones. A heterogeneous 3-D processor is composed of layers with different architectures, such as the core (logic) layers and memory layers. Whereas a homogeneous 3-D processor has an identical floorplan in all layers or mirrored floorplans in adjacent layers, with mixed core and memory parts in each layer. In this work, we call the homogeneous 3-D processors with the former architecture as the *identical processors* and the ones with the latter architecture as the *mirrored processors*, with examples given in Fig. 1. Specially, for a mirrored processor, we treat its two adjacent mirrored layers as one layer logically as shown in Fig. 1b, to keep a consistent expression with the identical 3-D processors.

Although having many advantages, 3-D processors experience severe thermal induced dark silicon problem because their 3-D stacking structure makes heat dissipation more difficult than the 2-D processors [5]. With limited heat dissipation ability, not all parts of the 3-D processor can be powered on at the same time in order to satisfy the thermal constraint. The inactive part of the processor is called dark silicon [6], [7].

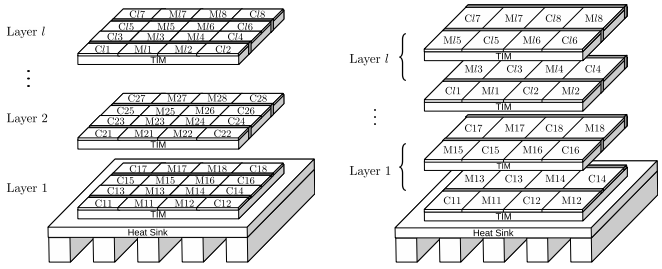
To improve the system performance under thermal constraint, many thermal/power management methods were first proposed for traditional 2-D processors, which adjust the temperature of the chip at runtime through techniques like dynamic voltage/frequency scaling (DVFS) and task migration [8]–[13]. Recently, machine learning based dynamic thermal management and performance optimization have been researched to improve the management quality. For example, echo state networks have been used as the nonlinear thermal model to improve temperature prediction accuracy [14]. Huang *et al.* used reinforcement learning to improve the control optimality considering environmental effects [15]. Multi-agent reinforcement learning has been introduced to optimize the performance of many-core systems under thermal constraint [16]. Decentralized control with centralized training scheme has been proposed for multi-core performance optimization [17]. For the heterogeneous 3-D processors, thermal management schemes [18]–[21] were proposed by adjusting the temperature distribution within the 3-D space. Runtime performance optimization technique for dark silicon heterogeneous 3-D processors [22] was proposed through co-optimization of the active core/memory's locations and settings. For the homogeneous 3-D processors, a task scheduling method was introduced in [23] which considers the vertical thermal coupling effects. Later, Wang *et al.* proposed a thermal and reliability management method [24], which improves the performance of the 3-D processors through fast lifetime estimation and lifetime banking. There are also methods that work for both the heterogeneous and homogeneous architectures, such as the task scheduling method introduced in [25] and the thermal management scheme proposed in [26].

Besides the thermal/power management techniques, determining the power budget, which is the maximum power allowed to be consumed under thermal constraint, is also important to enhance the system performance and reliability. However, to the best of our knowledge, there is no existing power budgeting method specially designed for the homogeneous 3-D processors in dark silicon, where the existing multi/many-core dark silicon power budgeting methods (including TSP [27], GDP [7], and DBP [28]) are only designed for the traditional 2-D processors. The challenge of designing a power budgeting method for homogeneous 3-D processors is two-fold. First, it is difficult to develop a runtime algorithm

This research is supported by National Natural Science Foundation of China under grant No. 61974018.

H. Wang, Y. Luo, and J. Guo are with School of Integrated Circuit Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 611731 China.

Y. Chen and J. Li are with HiSilicon Technologies Co., Ltd, Shenzhen, 518129 China.



(a) Identical processor, which has the same floorplan in all layers.

(b) Mirrored processor, which has mirrored floorplans in adjacent physical layers.

Fig. 1: The structures of two types of homogeneous 3-D processors. Cij and Mij denote for the jth core and the jth cache block in the ith layer, respectively. The grey strips in the layers are through silicon via (TSV) regions.

to find the optimal active core distribution and obtain the corresponding power budget in 3-D space with the goal of maximizing the potential system performance.[1] Second, we discover that the power budget ratios among the active cores in different layers have a significant impact on the total power budget of the homogeneous 3-D processors. As a result, proper power budget ratios among layers need to be determined to maximize the potential system performance. This is ignored in all previous methods, which simply assume the power budget to be the same in all layers. These challenges will be discussed in detail later in Section III-B.

By overcoming the challenges above, in this work, we propose a power budgeting method for homogeneous 3-D processors in dark silicon, with the following two new techniques:

- First, we propose a technique which finds a sub-optimal active core distribution in 3-D space at runtime by solving a performance optimization problem approximately in a greedy manner.
- Second, we have analyzed the problem of power assignment across the homogeneous 3-D layers for the first time, and found the layer wise power assignment has a significant impact on the system performance. Based on this analysis, we propose to compute the optimal power ratios among layers by solving a series of nonlinear optimization problems offline. Then, the optimal power ratios are integrated into the runtime optimization framework to obtain the power budget for each active core.

With the new techniques above, the new method has the following advantages over the existing methods:

- The new method provides a higher power budget than the existing methods because a sub-optimal active core distribution is determined at runtime.
- The performance of the 3-D system is higher by following the power budget provided by the new method, because the optimal power ratios among layers are followed in the power budgeting process.

[1]Although some task scheduling methods were proposed to work with the homogeneous 3-D processors without dark silicon [23], [25], [26], [29], they do not guarantee optimality for the dark silicon 3-D processors.

- In the experiments, we verify that the new method leads to a higher system performance on two homogeneous 3-D processors with multithreaded tasks, compared with the existing techniques.

II. Background

In this section, the power and thermal models used in this work are briefly presented.

A. Power model

The power of integrated circuits is composed of dynamic power and static (leakage) power.

The dynamic power, caused by the switching activity of the circuits, is modeled as

$$p_{d_i} = \alpha \cdot C_e \cdot V_i^2 \cdot f_i, \tag{1}$$

where α is the active factor of the core, C_e is the effective switching capacitance, V_i, p_{d_i}, and f_i are the voltage, dynamic power, and frequency of the ith core, respectively. The frequency and voltage of the ith core are further connected as

$$f_i = k_f \cdot \frac{(V_i - V_{th})^2}{V_{th}}, \tag{2}$$

where V_{th} is the threshold voltage and k_f is a fitting factor [30].

Leakage power, denoted as p_s, mainly depends on the temperature and is written as

$$p_s = V I_{leak}(T_p), \tag{3}$$

where T_p is the temperature in scalar form, I_{leak} is the leakage current, which is a monotonic increasing nonlinear function of temperature T_p. For the details of leakage power modeling, please refer to [31], [32].

B. Thermal model

The thermal model of the homogeneous 3-D system is built by exploiting the well-known duality between the thermal system and the electrical circuit system.

Assume we have a 3-D system with n_c cores, n_m cache blocks, n_l layers, and n_e cores in each layer ($n_c = n_e \cdot n_l$). Then, we can build a thermal model for this 3-D system with n thermal nodes, including the nodes for cores, cache blocks, and package parts. There are also $n_c + n_m$ power inputs in the thermal model, representing the powers of the n_c cores and n_m cache blocks. This thermal model can be written as

$$\begin{aligned} GT(t) + C\frac{\mathrm{d}T(t)}{\mathrm{d}t} &= \underbrace{\begin{bmatrix} B_c & B_m \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} P_c(t) \\ P_m(t) \end{bmatrix}}_{P(t)}, \\ \underbrace{\begin{bmatrix} Y_1(t) \\ \vdots \\ Y_{n_l}(t) \end{bmatrix}}_{Y(t)} &= \underbrace{\begin{bmatrix} L_1 \\ \vdots \\ L_{n_l} \end{bmatrix}}_{L} T(t), \end{aligned} \tag{4}$$

where $T(t) \in \mathbb{R}^n$ is the temperature vector, representing the temperatures of all n thermal nodes; $G \in \mathbb{R}^{n \times n}$ and

$C \in \mathbb{R}^{n\times n}$ contain thermal resistance and thermal capacitance information, respectively; $B \in \mathbb{R}^{n\times(n_c+n_m)}$ stores the topology information of how the powers are injected into the thermal nodes; $T_c(t) \in \mathbb{R}^{n_c}$ stores the temperature rises of the n_c cores; $L \in \mathbb{R}^{n_c\times n}$ is the output selection matrix which selects the temperatures of the n_c cores from $T(t)$ to form $Y(t)$. $P(t) \in \mathbb{R}^{n_c+n_m}$ contains the power consumptions of the n_c cores ($P_c(t)$) and n_m cache blocks ($P_m(t)$) at time t. $Y(t) \in \mathbb{R}^{n_c}$ is the output, containing the core temperatures.

We can further divide B_c and P_c into block matrices as

$$B_c = \begin{bmatrix} B_{c_1} & \cdots & B_{c_{n_l}} \end{bmatrix}, \quad P_c = \begin{bmatrix} P_{c_1} \\ \vdots \\ P_{c_{n_l}} \end{bmatrix}, \tag{5}$$

where B_{c_i} and P_{c_i} are matrices for the cores in the ith layer.

III. Power Budgeting of Dark Silicon Homogeneous 3-D Processors

A. Problem formulation

The goal of power budgeting is to determine the power consumption under thermal constraint which leads to the highest potential system performance. We can formulate power budgeting as the following optimization problem:

$$\begin{aligned} &\text{maximize } \|F\|_1 \\ &\text{subject to } \begin{cases} \text{card}(F) = n_a, \\ Y_{n_l}(P(F)) \preceq Y_{th}, \end{cases} \end{aligned} \tag{6}$$

where $Y_{th} \in \mathbb{R}^{n_e}$ is the temperature threshold vector. The optimization goal is to maximize the system total potential performance $\|F\|_1$. The constraint $\text{card}(F) = n_a$ means the number of active cores is n_a, where card is defined as the number of nonzero components in a vector. $Y_{n_l}(P(F)) \preceq Y_{th}$ is the thermal constraint, which means the maximum temperature[2] of the 3-D system should not exceed the threshold. $Y_{n_l}(P(F))$ contains both the thermal model ($Y_{n_l}(P)$) and the power model ($P(F)$) presented in Section II.

The solution to the optimization problem is the optimal P_c, which is the power budget of the cores that leads to the best potential system performance $\|F\|_1$.

B. Power budgeting challenges of homogeneous 3-D processors in dark silicon

There are two major challenges for the power budgeting of homogeneous 3-D processors in dark silicon.

First, solving the combinational optimization problem in (6) at runtime is difficult because there are many possible active core distributions (each active core distribution is a combination) for a given active core number n_a.[3] If we directly solve the problem by testing all the combinations and choosing the best solution (which leads to the largest $\|F\|_1$) among all the combinations, the computational delay will be unacceptable even for a system with a moderate number of cores.

[2]Please note that the lth layer has the highest temperature (Y_{n_l}) because it is at the end of the heat dissipation path in the 3-D system.

[3]For example, there are $\binom{16}{3} = 560$ active core distributions/combinations for a 16-core system with 3 active cores.

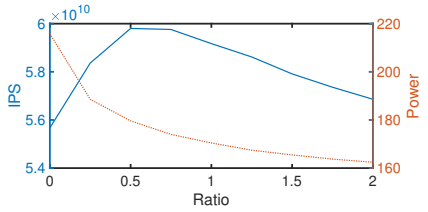


Fig. 2: The power (in dotted line with unit W) and performance (IPS in solid line) of the two-layer identical processor with different power budget ratios between the two layers. All cores are active and are running the *swaptions* bechmark.

Second, even for a fixed active core distribution, it is difficult to find the power budget that leads to the optimal potential total system performance $\|F\|_1$. Please note that to reach the same thermal constraint, power can be assigned with different ratios among layers. Since different power ratios among layers lead to different total system performances with the same thermal constraint, we need to find the optimal ratios that lead to the best total system performance.

For example, assume we have an identical two-layer 3-D processor (the 3-D processor in Fig. 1a with two layers) with all cores active. There are many choices to assign power between the two layers which satisfy the thermal constraint in (6): we can put all power in only one of the two layers, assign power equally, or divide power according to a ratio.

In Fig. 2, we plot the results of an experiment on this identical two-layer 3-D processor with all cores active by changing the power ratio between the two layers $r = P_{c_2}[i]/P_{c_1}[i]$ ($i = 1, 2, \ldots, 8$) from 0 to 2. We can see that the best total system performance is achieved at around the ratio $r = 0.5$, i.e., when the power budgets of the second layer cores are half of the first layer cores. It is interesting to see that the maximum total power of the 3-D system is reached at the power ratio $r = 0$, i.e., when all power is assigned to the first layer cores, simply because the first layer has a better heat dissipation condition. However, this ratio ($r = 0$) leads to poor system performance because power grows cubically with the frequency (performance), as shown in the power model (1) and (2). It is also undesirable to divide the power budget equally ($r = 1$) as in most existing methods, because it will overly suppress the total power budget as shown in Fig. 2.

In addition, we remark that each layer has its own optimal power ratio against the first layer (when there are more than two layers) and the optimal power ratios are also different with different active core numbers (different n_a).

C. Overview of the new power budgeting method

To overcome the challenges above, we propose a new power budgeting method which mainly contains two parts: the online active core locating and power budgeting part, and the offline power ratio computation part.

In the online part, to overcome the first challenge, a greedy based algorithm will locate the active cores one by one with the goal of maximizing system performance, which avoids the

costly exhaustive search. The power budgets of the located active cores will be determined by solving a thermal constraint problem, with the power budget ratios computed in the offline part.

In the offline part, to overcome the second challenge, the optimal power ratios among the active cores in different layers are computed by solving a series of nonlinear optimization problems. The ratios are then stored in a look-up table (LUT) for online usage.

In the following, we will first present the online part in Sections III-D and III-E, by assuming the power ratios computed offline are available. Then, the offline part, which serves the online part, will be presented in Section III-F.

D. Locating active cores to maximize system performance

We first present the active core locating flow by assuming that we already have a power ratio matrix R computed offline and stored in a LUT. Here R is a matrix whose element $R[n_a][l]$ is defined as the optimal power ratio of an active core in the lth layer against its overlapping first layer core, when there are totally n_a active cores in the 3-D processor. Mathematically, an element in R, take $R[n_a][l]$ for example, means the following power ratio constraint

$$\frac{P_{c_l}[i]}{P_{c_1}[i]} = R[n_a][l] \tag{7}$$

should be satisfied for all $i = 1, 2, \ldots, n_e$, $l \geq 2$, and $P_{c_l}[i] \neq 0$, when the active core number is n_a. The power ratios are computed using the procedure presented in Section III-F, such that the power budget determined by following these ratios will lead to the best total system performance.

With the power ratios available, we are now ready to solve the optimization problem in (6) by looking for the optimal active core distribution.

As discussed in Section III-B, traditionally we need to try all possible active core distributions to find the best one. To avoid the exhaustive search, we look for a sub-optimal solution by extending the strategy used in the greedy dynamic power (GDP) [7] to the 3-D space as shown in the following. The basic idea is to look for the local best solution one by one in a greedy manner.

Similar to [7], we first transform the original optimization problem (6) equivalently into an easier solving temperature optimization problem

$$\begin{aligned} &\text{minimize } \|Y_{th} - Y_{n_l}\|_2 \\ &\text{subject to } \begin{cases} \text{card}(P_c) = n_a \\ \frac{P_{c_l}[i]}{P_{c_1}[i]} = R[n_a][l], \forall i, l \geq 2, P_{c_l}[i] \neq 0 \\ Y_{n_l}(P) \preceq Y_{th}, \end{cases} \end{aligned} \tag{8}$$

where the power ratio constraint $\frac{P_{c_l}[i]}{P_{c_1}[i]} = R[n_a][l]$ is added to guarantee the equivalence between the new temperature optimization and the performance optimization in (6).

Since there is

$$Y_{n_l} = L_{n_l} G^{-1} BP = L_{n_l} G^{-1} (B_m P_m + B_c P_c) \tag{9}$$

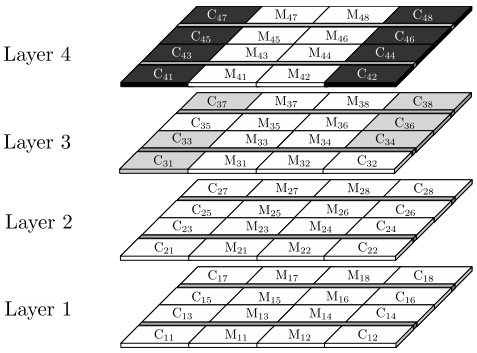


Fig. 3: Illustration of locating the 19-th active core in a 32-core 3-D processor with 4 layers ($n_e = 8$). The white cores are active, the black cores are off, and the grey cores are the candidates for the 19-th active core.

according to the thermal model (4) in steady state, we can further simplify (8) into

$$\begin{aligned} &\text{minimize } \|Y_c - A_c P_c\|_2 \\ &\text{subject to } \begin{cases} \text{card}(P_c) = n_a \\ \frac{P_{c_l}[i]}{P_{c_1}[i]} = R[n_a][l], \forall i, l \geq 2, P_{c_l}[i] \neq 0 \\ A_c P_c \preceq Y_c, \end{cases} \end{aligned} \tag{10}$$

where $A_c = L_{n_l} G^{-1} B_c \in \mathbb{R}^{n_e \times n_c}$ and $Y_c = Y_{th} - L_{n_l} G^{-1} B_m P_m \in \mathbb{R}^{n_e}$.

To solve the optimization problem above approximately, we are going to locate the active cores one by one in a greedy manner. However, to locate n_a active cores, we only need to perform ($n_a \bmod n_e$) iterations instead of n_a iterations as explained here. Please note that generally, any core in a lower layer is "better" than any core in a higher layer because a lower layer, being closer to the heat sink, has a better heat dissipation condition. In other words, we only activate a core in the i-th layer when all the cores in the lower layers (layer $1, 2, \ldots, i-1$) are already fully activated. As a result, we only need to pick the active cores in the $(\lceil \frac{n_a}{n_e} \rceil)$-th layer because all the cores in the lower layers should be active and all the cores in the upper layers should be off. In this case, only ($n_a \bmod n_e$) active cores need to be located. Take the 32-core 3-D processor with 4 layers ($n_e = 8$) in Fig. 3 as an example. If the active core number is 20 ($n_a = 20$), we only need to locate the active cores in the third layer ($\lceil \frac{n_a}{n_e} \rceil = \lceil \frac{20}{8} \rceil = 3$), because all the cores in the first and second layers should be active. Only 4 iterations ($n_a \bmod n_e = 20 \bmod 8 = 4$) are needed to find the optimal locations for the 4 active cores in the third layer.

To describe this iterative algorithm, we present the search for the ith active core ($n_e \lfloor \frac{n_a}{n_e} \rfloor < i \leq n_a$). In this iteration, to find the best ith core to activate when $i-1$ active cores have already been picked, we need to test the unpicked cores (called candidates) and choose the candidate that leads to the largest system performance (with all i picked cores active, including the candidate). Note that the ith active core will be searched only in the $(\lceil \frac{i}{n_e} \rceil)$-th layer, because all lower layer cores have already been activated and all higher layer cores are worse.

The example in Fig. 3 demonstrates the iteration to locate the 19-th active core with 18 active cores already located (C_{11} to C_{28}, C_{32} and C_{35}). In this example, the 19-th active core will be searched only in the third layer, with the 6 candidates shown in the grey color.

To test the remaining cores (candidates) within the $(\lceil \frac{i}{n_e} \rceil)$-th layer, we first subtract the thermal effects of the $i-1$ existing active cores from the temperature threshold and form the residual

$$Y_r = Y_c - \bar{A}_c \bar{P}_c, \tag{11}$$

where $\bar{A}_c \in \mathbb{R}^{n_e \times (i-1)}$ is a sub-matrix of A_c in the previous iteration, which is composed of the $i-1$ columns in A_c corresponding to the $i-1$ existing active cores. $\bar{P}_c \in \mathbb{R}^{i-1}$ is the power budgets of the existing $i-1$ active cores computed in the previous iteration (the $(i-1)$-th iteration).

The physical meaning of Y_r is the thermal headroom (with temperature threshold as the ceiling) left after activating the $i-1$ existing cores. As a result, the "best" candidate to activate is the one which complements the thermal headroom best. For each candidate (take the core with index j as an example) in the $(\lceil \frac{i}{n_e} \rceil)$-th layer, we simply compute the inner product of its corresponding column in A_c (denote as a_j) and Y_r as $\langle a_j, Y_r \rangle$, and pick the candidate with the largest inner product as the ith active core, and store its index in a vector idx as $idx[i]$. In case that the thermal headrooms above all candidates are zero, we can add a small perturbation (in our experiment, we find 0.1 works well) to Y_r to proceed.

When the ith active core is picked, the power budgets $\bar{P}_c$ will be updated for the i active cores using the procedure shown in Section III-E, which completes the ith iteration.

The greedy iteration will terminate when n_a active cores have been located, i.e., after the n_a-th iteration.

Note that the greedy algorithm does not guarantee to find an optimal solution. Fortunately, what we require is not the optimal solution, but a practically good solution that can be found in runtime speed. According to our analysis, the solution given by the greedy algorithm is acceptable: sometimes it will even find the optimal solution, usually when the first greedy active core is in the optimal mapping. Even when it finds a sub-optimal solution, the maximum power budget difference against the optimal is only around 1%.

E. Power budget computation for the active cores

In the greedy iteration presented in Section III-D, we need to update the power budgets $\bar{P}_c \in \mathbb{R}^i$ for the i active cores.

We first update $\bar{A}_c$ by adding the column of A_c corresponding to the newly picked ith active core as

$$\bar{A}_c = \begin{bmatrix} \bar{A}_c & a_{idx[i]} \end{bmatrix}. \tag{12}$$

Then, the power budgets of active cores are connected to the thermal constraint by the following equation

$$M_v Y_c = M_v \bar{A}_c \bar{P}_c, \tag{13}$$

where $M_v \in \mathbb{R}^{n_v \times n_e}$ is a mapping matrix which chooses the n_v top layer cores with at least one overlapping active core (including itself), out of all n_e top layer cores. Since the active cores are located in the lower layers first, there is $n_v = \min(i, n_e)$. Specially, when $i \geq n_e$, all first layer cores are active, making M_v an $n_e \times n_e$ identity matrix.

By introducing the ratio matrix $\bar{R} \in \mathbb{R}^{i \times n_v}$ for the ith iteration, which consists of elements in R to connect the power budgets of all active cores $\bar{P}_c \in \mathbb{R}^i$ with the first layer active core power budgets $\bar{P}_{c_1} \in \mathbb{R}^{n_v}$ as

$$\bar{P}_c = \bar{R} \bar{P}_{c_1}, \tag{14}$$

we readily have

$$M_v Y_c = M_v \bar{A}_c \bar{R} \bar{P}_{c_1}. \tag{15}$$

Since $M_v \bar{A}_c \bar{R} \in \mathbb{R}^{n_v \times n_v}$ is a square matrix, $\bar{P}_{c_1}$ can be solved uniquely from (15). Finally, the power budgets of all active cores $\bar{P}_c$ are simply recovered using (14).

The runtime algorithm of the new method is summarized in Algorithm 1. Please note that it is only necessary to perform the full algorithm when the active core number changes (i.e., when threads are scheduled to some new active cores or when the threads on some active cores terminate). If there is no change in the active core number, we may skip the active core locating process and just update the power budget by executing only lines 15 and 18 of Algorithm 1.

Algorithm 1 The runtime power budgeting algorithm for homogeneous 3-D processors in dark silicon

Input: A_c, Y_c, R, n_a, n_e
Output: power budget vector P_c

1: Form $\bar{A}_c$, M_v, $\bar{R}$ with $n_e \lfloor \frac{n_a}{n_e} \rfloor$ active cores in lower layers
2: $\bar{P}_c = \bar{R}(M_v \bar{A}_c \bar{R})^{-1} M_v Y_c$
3: $Y_r = Y_c - \bar{A}_c \bar{P}_c$
4: **for** $i \leftarrow n_e \lfloor \frac{n_a}{n_e} \rfloor + 1, n_a$ **do**
▷ *Find the i-th active core within the $(\lceil \frac{i}{n_e} \rceil)$-th layer*
5: $k = (\lceil \frac{i}{n_e} \rceil - 1) \cdot n_e + 1$ ▷ *Index of the 1st core in layer*
6: $idx[i] = k$ ▷ *Index of the i-th active core*
7: **for** $j \leftarrow k+1, k+n_e-1$ **do**
8: **if** $\langle a_j, Y_r \rangle > \langle a_{idx[i]}, Y_r \rangle$ **then**
9: $idx[i] = j$
10: **end if**
11: **end for**
▷ *Update power budgets for i active cores*
12: $\bar{A}_c = \begin{bmatrix} \bar{A}_c & a_{idx[i]} \end{bmatrix}$
13: Update M_v according to $idx[i]$
14: Update $\bar{R}$ according to $idx[i]$, $R[i][l]$, $l = 2, \ldots, \lceil \frac{i}{n_e} \rceil$
15: $\bar{P}_c = \bar{R}(M_v \bar{A}_c \bar{R})^{-1} M_v Y_c$
16: $Y_r = Y_c - \bar{A}_c \bar{P}_c$
17: **end for**
18: Formulate P_c using $\bar{P}_c$

F. Computing optimal power ratios among layers

To perform the greedy based performance optimization of the homogeneous 3-D system, we need to first determine the optimal power ratios among layers, which maximize the performance of the homogenous 3-D system, and store them in a lookup table (LUT).

For each active core number, the number of possible active core distributions can be extremely large, so it is infeasible

to compute and maintain a large LUT indexed by each distribution. To solve this problem, we only compute and store the average optimal power ratio in each greedy iteration. In this way, the LUT size and computation can be reduced to a small size without losing generality: now the LUT is not indexed by each distribution, but only by each active core number n_a.

For the active core number n_a that is smaller than or equal to the core number in a layer ($n_a \leq n_e$), we do not need any power ratios because all active cores are located in the first layer to gain power/performance.

For each active core number n_a that is larger than the core number of a layer ($n_a > n_e$), we need to compute $\lceil \frac{n_a}{n_e} \rceil - 1$ power ratios $R[n_a][l]$, $l = 2, 3, \ldots, \lceil \frac{n_a}{n_e} \rceil$, where $R[n_a][l]$ was defined previously in (7).

To obtain the power ratios which match the runtime algorithm without losing generality, we utilize a similar greedy iterative strategy in the power ratio computation. Specifically, to compute the power ratios for n_a active cores ($R[n_a, l]$), we fix the $n_a - 1$ active core locations found in the previous iterations, and obtain the distributions of n_a active cores by changing the location of the n_a-th active core in its search layer.[4] For each distribution, an optimal power ratio will be computed as described in the following paragraphs. The final power ratio $R[n_a, l]$ is the average of the optimal power ratios for the distributions of n_a active cores.

To get the optimal power ratio for each active core distribution, we solve the following nonlinear optimization problem, which is just the original performance optimization problem (6) with the specified active core distribution:

$$\begin{aligned} &\text{maximize } \|F_a\|_1 \\ &\text{subject to } \begin{cases} F = M_a F_a, \\ Y_l(P(F)) \preceq Y_{th}, \end{cases} \end{aligned} \tag{16}$$

where $M_a \in \mathbb{R}^{n_c \times n_a}$ is a mapping matrix which specifies the distribution of the n_a active cores.

The optimization problem above can be solved offline, which gives us the optimal frequencies F_a and power budgets P_c of the active cores. Then, the optimal power ratio of this distribution (assume it is the kth distribution) is computed and stored in a temporary matrix R_{tmp} as

$$R_{tmp}[l][k] = \frac{1}{\text{card}(P_{c_l})} \sum_{i=1}^{n_e} P_{c_l}[i] / P_{c_1}[i], \tag{17}$$

where $\text{card}(P_{c_l})$ is the number of active cores in the lth layer. The total system frequency of the kth distribution is also recorded in a vector F_{tot} as $F_{tot}[k] = \|F_a\|_1$.

Next, to be consistent with the runtime procedure, the core that leads to the best performance (largest $F_{tot}[k]$) is activated as the n_a-th active core, whose location will be fixed for the next iteration.

Finally, we obtain the power ratios for each active core number n_a by taking the average of all ratios tested. Specifically, the power ratio for the lth layer with n_a active cores is computed as

$$R[n_a][l] = \frac{1}{n_e - j + 1} \sum_{k=1}^{n_e - j + 1} R_{tmp}[l][k], \tag{18}$$

which will be stored in the LUT for runtime use.

The offline optimal power ratio computation algorithm is summarized in Algorithm 2.

Algorithm 2 Computing optimal power ratios among layers

Input: Thermal & power models, Y_{th}, n_e, n_l
Output: Ratio LUT R

$S_a = [1, 2, \ldots, n_e]$ ▷ *Indices of active cores*
for $i \leftarrow 2, n_l$ **do** ▷ *Inter layer iteration*
  $S_r = [1, 2, \ldots, n_e]$ ▷ *Local inactive indices in layer*
  for $j \leftarrow 1, n_e$ **do** ▷ *Intra layer iteration*
    $n_a = n_e(i-1) + j$ ▷ *Current active core number*
    ▷ *Compute power ratio of activating one more core in layer*
    for $k \leftarrow 1, n_e - j + 1$ **do**
      $S_t = [S_a, n_e(i-1) + S_r[k]]$
      Formulate M_a with active core indices in S_t
      Solve nonlinear optimization problem (16)
      Compute $R_{tmp}[l][k]$ for $l = 2, 3, \ldots, \lceil \frac{n_a}{n_e} \rceil$
      Obtain $F_{tot}[k]$ ▷ *Performance of the k-th test*
    end for
    ▷ *Activate the core that leads to the best performance*
    $idx = \max(F_{tot})$
    $S_a = [S_a, n_e(i-1) + S_r[idx]]$
    Delete $S_r[idx]$ from S_r
    ▷ *Compute the power ratio as the average of the ratios tested*
    for $l \leftarrow 2, \lceil \frac{n_a}{n_e} \rceil$ **do**
      $R[n_a][l] = \frac{1}{n_e - j + 1} \sum_{k=1}^{n_e - j + 1} R_{tmp}[l][k]$
      Store $R[n_a][l]$ in the ratio LUT R
    end for
  end for
end for

IV. Experimental Results

To show the advantage of the new method, we would like to compare it with existing power budgeting schemes for homogeneous 3-D processors. Through extensive studies, we find most of the existing power budgeting and dynamic thermal management related techniques for 3-D processors only work for the heterogeneous 3-D architecture (such as [18]–[22]). The important methods that are compatible with the homogeneous 3-D architecture include [23], [25], [26], [29]. Among these candidates, we choose TPAVA (full name of the algorithm is TPAVA+VGVS) [23] and 3D-DNaPE [29], to be compared with the new method. TPAVA is a classical method which was reported to be superior to the previous methods (with system throughput improvement of 6.50% and 32.06% compared with [25] and [26], respectively). 3D-DNaPE represents a recent approach which improves from the classical ones by sensing the neighbor cores' temperatures.

[4]Remind that an active core is searched only in one layer, because the active cores are located in order from the lower layer to the higher layer.

In order to analyze the new method thoroughly, we perform the experiment using the PARSEC [33] and SPLASH2 [34] benchmarks, where the 3-D processor runs a PARSEC/SPLASH2 multithreaded task, with the active core number equal to the thread number of the task. In this way, we can analyze the system performance with the multithreaded tasks, which are widely used in multi-core processors today.

In this part, we will first present the general experimental settings that are shared by all tests in Section IV-A. From Section IV-B to Section IV-D, we compare the new method with the existing methods [23], [29] using the multithreaded PARSEC/SPLASH2 benchmarks. Then, in Section IV-E, the impact of different TIM thicknesses is analyzed. Finally, the computing overhead results and behavior on many-core system are presented in Section IV-F.

A. General experimental settings

We built two homogeneous 3-D processor models using HotSpot [35] with 3-D extension [36] to test the new method. The first one is a two-layer identical 3-D processor (as in Fig. 1a), with 8 cores in each layer, making a total of 16 cores. We call this processor the *identical processor* in short. The second processor is a mirrored 3-D processor (as in Fig. 1b), with four physical die layers (two layers logically) and 4 cores in each physical layer, also making a total of 16 cores. We call this processor the *mirrored processor* in short.

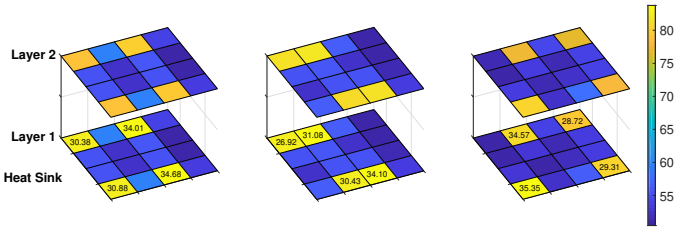
The areas of the identical processor and the mirrored processor are $10\,\mathrm{mm} \times 10\,\mathrm{mm}$ and $7.07\,\mathrm{mm} \times 7.07\,\mathrm{mm}$, respectively. The thickness of each die layer is $0.15\,\mathrm{mm}$, and the thickness of each TIM is $0.02\,\mathrm{mm}$. The rest parameters are default in HotSpot for 3D MCPs.

All simulations are performed using HotSniper [37] by running the PARSEC/SPLASH2 benchmarks with X86 architecture. Some parameters are modified to simulate a practical running environment: a typical 3-D processor installed in a personal computer. Since the default power scaling parameter in HotSniper will lead to low power consumptions that require no thermal management, we adjusted it to 4 in order to achieve a reasonable power consumption with the V/f level ranging from ($1.0\,\mathrm{GHz}$, $0.8\,\mathrm{V}$) to ($5.0\,\mathrm{GHz}$, $1.6\,\mathrm{V}$). We also set the power consumption of the inactive cores to zero in HotSniper to model the power gated inactive cores in the dark silicon system. The frequency increase constraint in the original HotSniper, which limits the maximum frequency increase in each DVFS action to be $1\,\mathrm{GHz}$, is removed. The other settings in HotSniper are kept as default, including the ambient temperature and the threshold temperature set as $45\,^\circ\mathrm{C}$ and $80\,^\circ\mathrm{C}$, respectively. In this test, the power budgeting cycle is $10\,\mathrm{ms}$. Thread migration is performed every $100\,\mathrm{ms}$ or at the next $10\,\mathrm{ms}$ when the active core number changes.

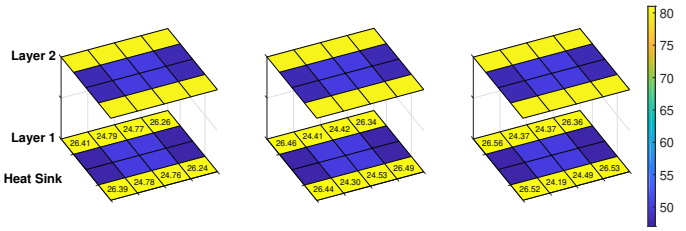
The experiments are performed on a server with Intel i9-13900K CPU and $128\,\mathrm{GB}$ memory.

B. Comparison of active core distributions and power budgets

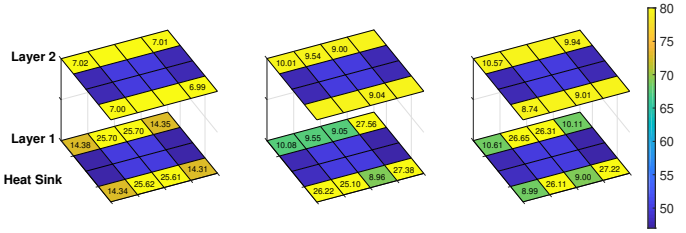
First, we analyze the behavior of the new method, TPAVA [23] and 3D-DNaPE [29] on the identical processor. The identical processor's active core distribution and power budget results with the three methods for different active core number are given in Fig. 4.



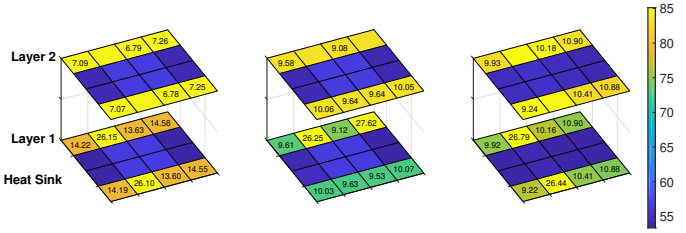
(a) The new method (left), TPAVA (middle), and 3D-DNaPE (right) with 4 active cores.



(b) The new method (left), TPAVA (middle), and 3D-DNaPE (right) with 8 active cores.



(c) The new method (left), TPAVA (middle), and 3D-DNaPE (right) with 12 active cores.

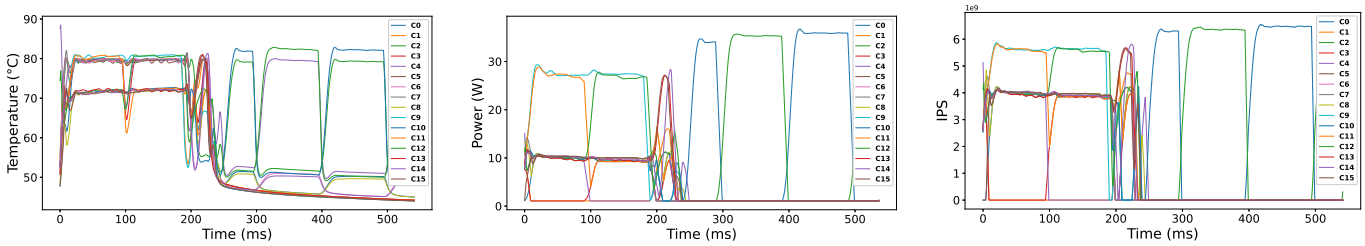


(d) The new method (left), TPAVA (middle), and 3D-DNaPE (right) with 14 active cores.

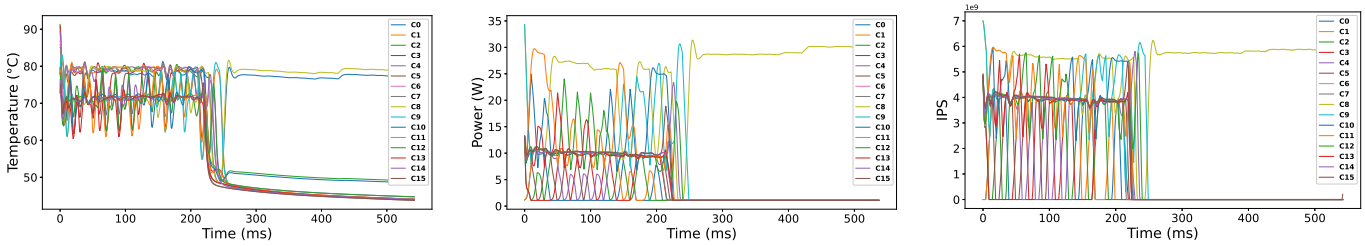
Fig. 4: The power (with unit W) and temperature (with unit $^\circ$C) distribution of the 16-core identical processor determined by the new method, TPAVA, and 3D-DNaPE with different number of active cores. The benchmark used here is *swaptions*. In each subfigure, the numbers in the active cores are their power values, and the cores without any number are inactive.

When the active core number is less than 8 ($n_a < 8$), all methods activate only the first layer cores to gain more power and higher performance, as shown in Fig. 4a. The difference is that our method and 3D-DNaPE achieve a better active core distribution in the first layer without forming any active core cluster.

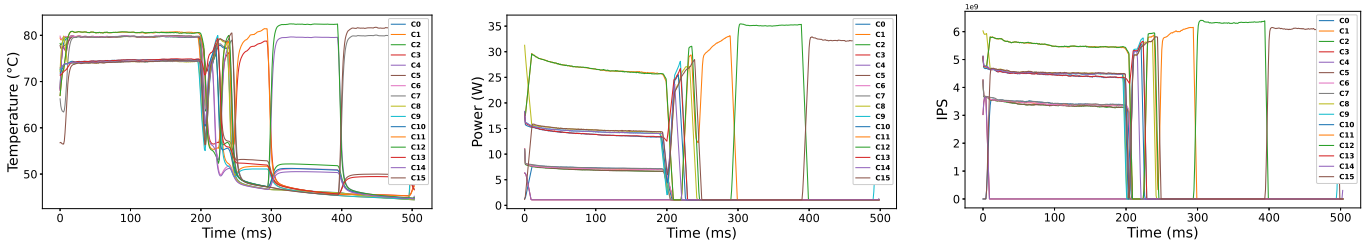
When the active core number is 8, the new method and the existing methods lead to similar results, as shown in Fig. 4b. All methods activate cores in the first layer because this layer



(a) Transient results of TPAVA [23].



(b) Transient results of 3D-DNaPE [29].



(c) Transient results of the new method.

Fig. 5: Transient temperature, power and throughput (IPS) comparison of the new method and the existing methods [23], [29] on the identical processor running *swaptions* on 15 active cores. Temperature threshold is $80\,^{\circ}\mathrm{C}$.

has the best heat dissipation condition.

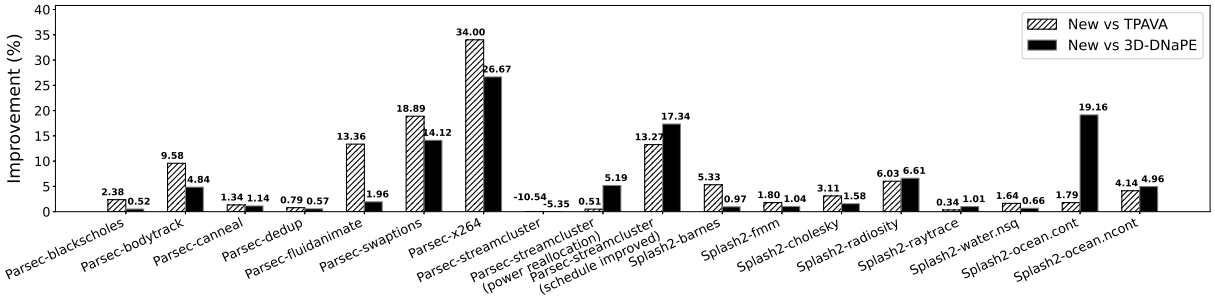
When the active core number is larger than 8, the new method shows significant advantage, as observed from Fig. 4c to Fig. 4d. The advantage comes from two aspects. First, with the new method, active core clusters in the second layer are avoided, leading to a better heat dissipation condition. Second, the new method assigns the power budgets between two layers according to the optimal power ratio computed offline. As shown in Fig. 4d, with the new method, all the overlapping active core pairs have a power ratio around $1:2$. On the other hand, for the existing methods, the recent 3D-DNaPE method performs a little better than the classical TPAVA, because it avoids active core clusters by sensing the neighbor cores' temperatures dynamically. However, both existing methods adjust the V/f levels of the overlapping active cores with an equal ratio, resulting in a similar power budget in the overlapping cores. Such optimized power budget assignment in the new method eventually leads to higher system performance as shown next.

C. Transient temperature, power, IPS comparison

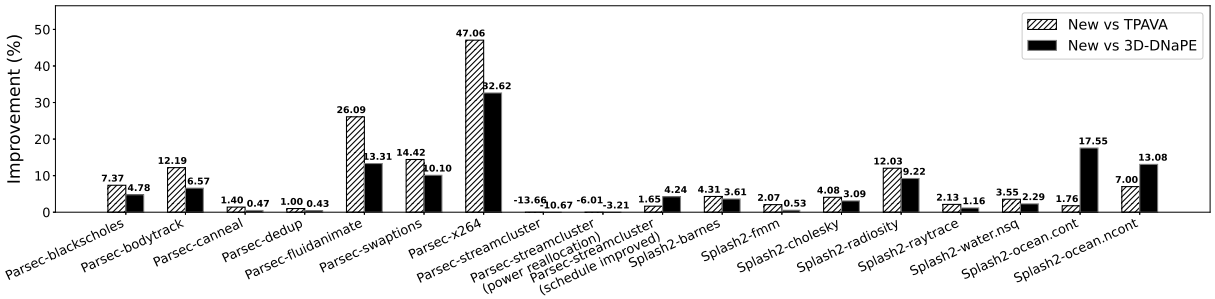
The transient temperature, power, and throughput (measured as instructions per second (IPS)) comparison results of the new method and the two existing methods [23], [29] on the identical processor are plotted in Fig. 5. In this test, benchmark *swaptions* is used with 15 threads (running on 15 active cores).

From the figures, we can see that with all methods, the maximum temperatures of the processors follow the threshold ($80\,^{\circ}\mathrm{C}$) tightly, indicating that all methods can keep the processor thermally safe.

For the throughput (IPS) and power traces in Fig. 5, let us focus on the parallel running part (from the beginning to around $200\,\mathrm{ns}$) of the benchmark. With TPAVA as shown in Fig. 5a, the active cores are generally divided into two categories: the active cores in category one have IPS of around 6×10^9, corresponding to the ones with no other active core stacked vertically. The active cores in category two have IPS of around 4×10^9, corresponding to the ones with another active core vertically stacked. Note that in category two, the active cores in different layers have similar IPS because the power budget is evenly divided across layers, as shown in the power trace figure of Fig. 5a. For 3D-DNaPE shown in Fig. 5b, although the active cores are still divided into two categories similar to TPAVA, there are significant temperature fluctuations of the active cores. These temperature fluctuations are caused by the task migrations of moving the threads



(a) Performance improvement of the identical processor.



(b) Performance improvement of the mirrored processor.

Fig. 6: Performance improvement results of the multithreaded PARSEC/SPLASH2 benchmarks running on the two 3-D processors (identical and mirrored) with the new method and two existing methods.

from the high temperature cores to their low temperature neighbors, resulting in a slightly higher system performance than TPAVA. In contrast, with the new method as shown in Fig. 5c, there are three kinds of active cores: the first kind with around 6×10^9 IPS is the same as the first category of the existing method. The remaining two kinds of active cores have IPS of around 4.8×10^9 and 3.7×10^9, respectively. These are the ones with another vertically stacked active core, but divided into two kinds because they are in two different layers with different power budget assigned according to the optimal power ratio as shown in the power trace figure of Fig. 5c. Since $4.8 \times 10^9 + 3.7 \times 10^9$ is larger than $2 \times 4 \times 10^9$, the throughput (total IPS) of the processor with the new method is clearly higher than the one with TPAVA, showing the advantage of the intelligent power assignment strategy of the new method.

Another important observation is that the average temperature of the processor's bottom/first layer is much higher with the new method than with the existing methods. This observation explains the power/performance advantage brings by the new method: with a similar maximum temperature compared with the existing methods, the new method explores the power budget/performance potential of the 3-D system by assigning power across layers intelligently according to the optimal power ratios, which rises the lower layers' (the bottom/first layer in this example) temperatures.

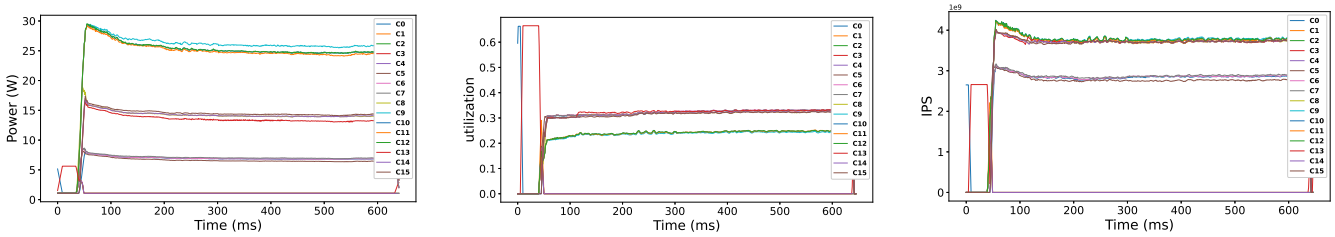
D. Test on different PARSEC/SPLASH2 benchmarks

The performance improvement results of different PARSEC/SPLASH2 benchmarks running on the two 3-D processors (identical and mirrored) with the new method and the existing methods are shown in Fig. 6. In this test, the thread number of benchmarks is set to 13, except for the following set to the largest supported value: 9 for "fluidanimate" and "x264", 8 for "ocean.count" and "ocean.ncount". All benchmarks execute the simlarge dataset, except for "canneal", "dedup", and "x264": "canneal" and "dedup" run simsmall due to their excessive simulation time with simlarge, and "x264" executes simmedium because of the inability to run simlarge.

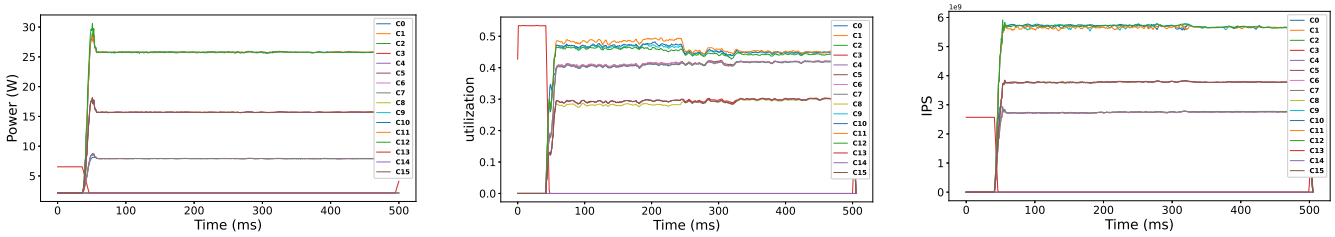
In general, the new method outperforms both existing methods. Specifically, it has a greater performance lead over TPAVA than 3D-DNaPE on most benchmarks, because 3D-DNaPE improves from TPAVA with a better active core distribution by sensing the neighbor cores' temperatures.

It is noted that the new method performs similarly to the existing methods on some benchmarks like "canneal". This is simply because "canneal" is not a computing-intensive benchmark, whose power consumption is low. Thus, the processors will remain cool by running "canneal" even at the maximum frequency, and no dynamic thermal management will be triggered to throttle the processor performance, leading to a similar performance.

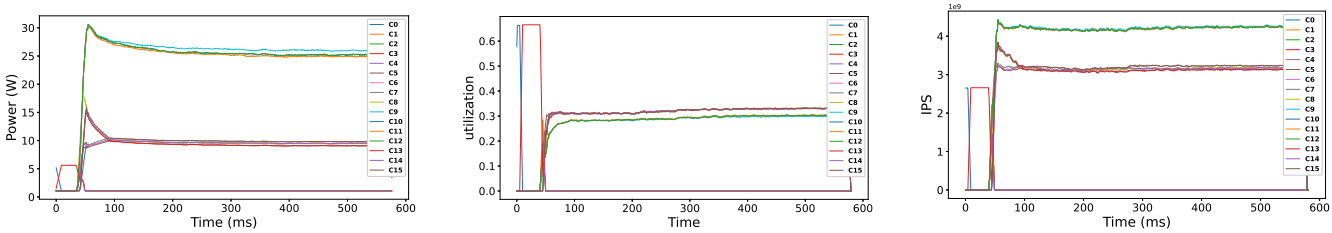
The only exception is on "streamcluster" that the existing methods actually perform better. After investigation, we discovered that the active cores running the "streamcluster" benchmark in parallel have large diversity in utilization ratio (i.e., activity factor) with the new method as shown in Fig. 7a. This diversity in utilization ratio is caused jointly by the diversity in power budget provided by the new method and the one core one thread scheduling. Specifically, in "streamcluster" the fast threads tend to wait for the slow threads, so the high power budget cores will run into a low efficiency state with the one core one thread scheduling: they will raise frequency



(a) Transient results of the new method, with the one core one thread scheduling.



(b) Transient results of the new method, by scheduling one more thread to the high power budget cores (the ones with over 25 W) using SMT.



(c) Transient results of the new method with power reallocation, which reallocates power from the layers with underutilized cores to layers with more active cores until the utilization difference falls below 10%.

Fig. 7: Results of two policies (scheduling improvement and power reallocation) to mitigate the uneven utilization of the new method. In this test, *streamcluster* benchmark is running on 13 active cores of the identical processor.

trying to consume the high power budget, but at the same time wait for the slow threads on the low power budget cores, leading to low utilization. This problem is more serious with the new method where the power budget difference among cores is larger than the existing methods. This is because the new method optimizes the *potential* system performance, with the assumption that the utilization is the same for all cores, which breaks down in this case.

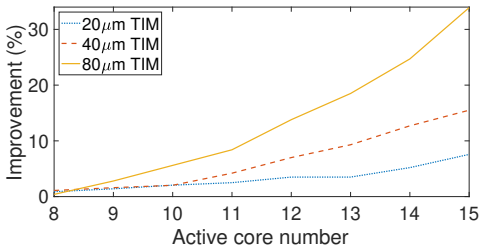
This problem can be alleviated by either scheduling improvement or power reallocation. Scheduling improvement removes the one core one thread restriction to utilize the power budget more efficiently. For example, with the same power budget provided by the new method, if we just schedule one more thread to the cores with the high power budget (the ones with over 25 W) by enabling simultaneous multithreading (SMT) in HotSniper, the processor performance can be significantly boosted (although still not fully optimized), as shown in Fig. 7b.

For power reallocation, if the utilization of the l-th layer is significantly higher than that of the first layer, we iteratively increase the power ratio $R[n_a][l] = \frac{P_{c_l}}{P_{c_1}}$ toward 1. This systematically reallocates power from the layers with underutilized cores to layers with more active cores until the utilization difference falls below a preset threshold. Results of this mechanism with the 10% utilization difference threshold are presented in Fig. 7c. We observe that the power reallocation successfully mitigates the uneven utilization problem of the proposed method. It should be noted, however, that this approach yields slightly lower performance compared to the scheduling improvement (Fig. 7b). This occurs because power reallocation shifts the power ratio away from its theoretical optimal point in order to balance utilization. We conclude that power reallocation serves as an excellent supplementary technique in scenarios where an advanced scheduling scheme is unavailable.

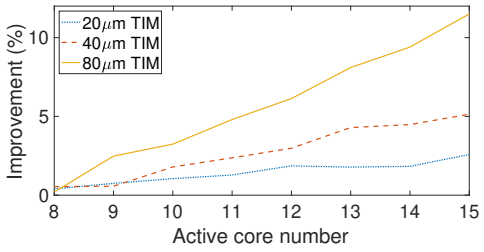
With the two schemes above, the new method now leads to a similar or even better system performance compared with the existing methods on the "streamcluster" benchmark, as shown in Fig. 6 with labels "Parsec-streamcluster (schedule improved)" and "Parsec-streamcluster (power reallocation)".

E. System performance improvement with different TIM thicknesses

The thickness of the TIM between the 3-D layers has a significant impact on the power/performance advantage brings by the intelligent power budget assignment across layers. The



(a) Power improvement.



(b) IPS improvement.

Fig. 8: The power and performance (IPS) improvements with the new method over TPAVA on the identical processor with different TIM thicknesses and different active core number running *swaptions* benchmark.

performance and power improvements with the new method against the existing method with different TIM thicknesses are plotted in Fig. 8, with different active core number.

We observe that the improvement is more significant with a thicker TIM: when 16 cores are active, the new method achieves over 10% performance improvement with $80\,\mu\mathrm{m}$ thick TIM, whereas the improvement is less than 3% with $20\,\mu\mathrm{m}$ thick TIM. This indicates that the 3-D processor with a worse vertical heat conduction actually has a higher demand for the intelligent power assignment across layers.

F. Computing overhead analysis and behavior on many-core system

The computing overhead results of the new method are collected in Table I, where "mapping time" is the overhead of executing the full algorithm in Algorithm 1 including the active core locating process and "budgeting time" is the power budget updating overhead without changing the active core distribution.

We can see that the "mapping time" grows with the active core number because the greedy iteration number equals the active core number. Overall, this overhead is acceptable with one ms delay considering the extra power budget and performance the new method brings.

The "budgeting time" is extremely small because only lines 15 and 18 of Algorithm 1 are executed without any iteration to locate the active cores. This overhead also increases with the active core number because the sizes of the matrices in line 15 are related to the active core number.

From this test, we see the way to run the new power budgeting method efficiently is to perform the full algorithm only when the active core number changes (i.e., when threads are scheduled to some new active cores or when the threads on some active cores terminate) in order to find the optimal active core distribution. In the other cases, just update the power budget with the existing active core distribution.

We also test the performance of the new method on many-core systems, by building a 2-layer 64-core identical 3-D processor, with 32 cores in each layer. The behavior of the new method compared against TPAVA and 3D-DNaPE is given in Fig. 9, which looks similar to that of the 16-core 3-D processor tested before. In this test, the average performance improvement of the new method is 16% against TPAVA and 12% against 3D-DNaPE.

The computing overhead of the new method on the 64-core 3-D processor is given in Table II. As expected, we see that "mapping time" on the 64-core processor becomes larger than that on the 16-core processor shown before in Table I, because there are now 32 cores in each layer. However, note that the "mapping time" overhead of the new method is mainly sensitive to the core number in each layer, but not to the total core count. So for the many-core 3-D processors with many layers but few cores per layer, the new method's computing overhead will not increase significantly.

On the other hand, the "budgeting time" is generally the same for both the 16-core and 64-core processors, with the same active core number. By analyzing the equation for power budget computing (Eq. (15) or Line 15 of Algorithm 1), we know this is because the power budgeting process is mainly governed by n_v, and is irrelevant to the total core number.

Finally, we also tested the computational cost of the offline optimal power ratio computation algorithm (Algorithm 2) presented in Section III-F. In our experiment, the nonlinear optimization problem always converges by using the "fmincon" function with "active-set" algorithm in MATLAB. As an offline step, the computational cost of this algorithm is moderate: it takes around 5 to 10 minutes to complete in our test cases.

V. Limitations and Future Work

The new power budgeting method for homogeneous 3-D processors in dark silicon is not perfect. Here we list its major known limitations and the future research directions to overcome these limitations.

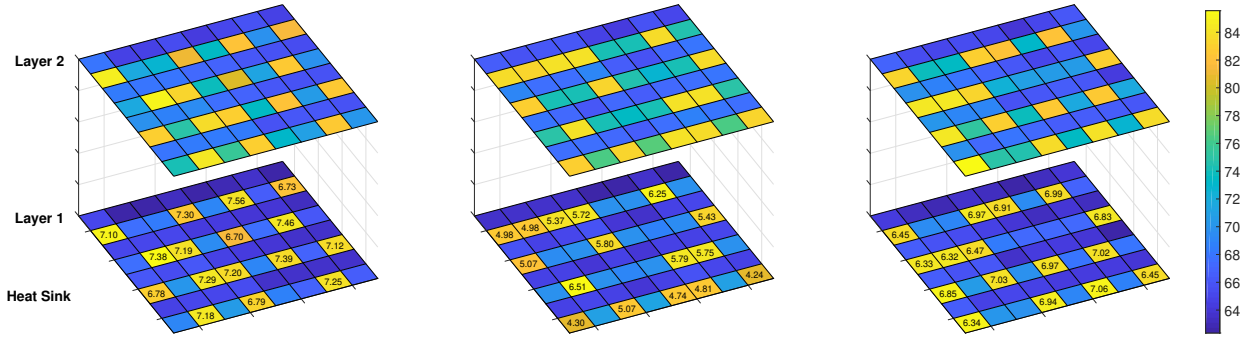
- Although the new method optimizes processor's potential performance, using the new method alone does not always guarantee good performance under different runtime operating conditions, unless it is accompanied by a good scheduling scheme. Specifically, it implicitly assumes equal activity factor in the potential performance optimization, which may break down if threads are not scheduled properly. Next, we plan to develop an advanced scheduling method which utilizes the provided power budget efficiently, to fully release the performance potential brought by the new method.
- The new method is currently verified only by simulation. In the future, we would like to test it on some real processors. We expect to improve the power model using a machine learning based model and integrate an advanced

TABLE I: The computing overheads of the new method on the 16-core identical and mirrored processors. In this table and also Table II, "Mapping time" means the overhead of executing the full algorithm in Algorithm 1 including the active core locating process. "Budgeting time" is the power budget updating overhead without changing the active core distribution.

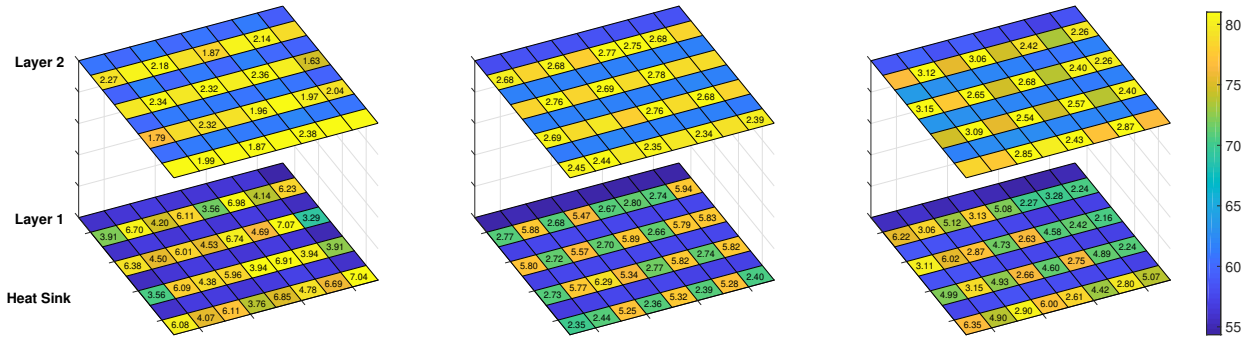
Active core #		1	3	5	7	9	11	13	15
identical	budgeting time (µs)	0.63	3.5	3.8	5.0	5.9	6.2	6.3	6.4
	mapping time (µs)	3.0	13	26	40	20	64	109	158
mirrored	budgeting time (µs)	0.62	3.4	3.9	4.5	4.9	6.2	6.3	7.1
	mapping time (µs)	5.0	23	43	59	156	298	426	530

TABLE II: The computing overheads of the new method on the 64-core identical processor.

Active core #	1	6	11	16	21	26	31	36	41	46	51	56	61
budgeting time (µs)	0.56	4.0	4.7	5.5	8.5	9.5	12	15	15	16	16	17	17
mapping time (µs)	3.8	33	65	99	133	173	212	116	284	348	478	609	746



(a) The new method (left), TPAVA (middle), and 3D-DNaPE (right) with 16 active cores.



(b) The new method (left), TPAVA (middle), and 3D-DNaPE (right) with 48 active cores.

Fig. 9: The power (with unit W) and temperature (with unit °C) distribution of the 64-core identical processor determined by the new method, TPAVA, and 3D-DNaPE with different number of active cores. The benchmark used here is *swaptions*. In each subfigure, the numbers in the active cores are their power values, and the cores without any number are inactive.

scheduler to enhance the performance/power prediction accuracy in real running environment.

- Although the computing overhead of the new method is low on the multi-core 3-D processors, the scalability of its mapping process is not very good for some many-core 3-D processors, because the mapping overhead grows with the active core number and the core number in each layer. Developing a distributed version of the mapping algorithm is a way to solve this problem in the future.

VI. Conclusion

In this article, we have presented a power budgeting method for homogeneous 3-D processors in dark silicon. A greedy iterative algorithm in the new method finds the optimal 3-D active core distribution and power budget at runtime. In addition, the impact of power assignment across 3-D layers is investigated, and an intelligent power assignment method with optimal power ratios computed offline is proposed. Experiments on two 3-D processors with different homogenous architectures using multi threaded tasks show that the new method outperforms the existing methods in system performance.

Acknowledgment

The authors acknowledge the use of generative AI for grammar checking of this manuscript. All technical content and results were developed and verified by the authors.

REFERENCES

[1] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration," *Proc. of the IEEE*, vol. 89, no. 5, pp. 602–633, May 2001.

[2] R. Weerasekera, L.-R. Zheng, D. Pamunuwa, and H. Tenhunen, "Extending systems-on-chip to the third dimension: performance, cost and technological tradeoffs," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, November 2007, pp. 212–219.

[3] K. Cao, J. Zhou, T. Wei, M. Chen, S. Hu, and K. Li, "A survey of optimization techniques for thermal-aware 3D processors," *Journal of System Architecture*, vol. 97, pp. 397–415, August 2019.

[4] D. H. Kim, K. Athikulwongse, M. B. Healy, M. M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y.-J. Lee, D. L. Lewis, T.-W. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. H. Loh, H.-H. S. Lee, and S. K. Lim, "Design and analysis of 3D-MAPS (3D massively parallel processor with stacked memory)," *IEEE Trans. on Computers*, vol. 64, no. 1, pp. 112–125, January 2015.

[5] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschwiler, "3D-ICE: A compact thermal model for early-stage design of liquid-cooled ICs," *IEEE Trans. on Computers*, vol. 63, no. 10, pp. 2576–2589, October 2014.

[6] M. Taylor, "A landscape of the new dark silicon design regime," *IEEE MICRO*, vol. 33, no. 5, pp. 8–19, October 2013.

[7] H. Wang, D. Tang, M. Zhang, S. X.-D. Tan, C. Zhang, H. Tang, and Y. Yuan, "GDP: A greedy based dynamic power budgeting method for multi/many-core systems in dark silicon," *IEEE Trans. on Computers*, vol. 68, no. 4, pp. 526–541, April 2019.

[8] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Multicore thermal management with model predictive control," in *Proc. European Conference on Cicuit Theory and Design*, August 2009, pp. 90–95.

[9] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 3, pp. 329–339, August 2010.

[10] A. Schranzhofer, J.-J. Chen, and L. Thiele, "Dynamic power-aware mapping of applications onto heterogeneous MPSoC platforms," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 4, pp. 692–707, November 2010.

[11] X. Wang, X. Fu, X. Liu, and Z. Gu, "PAUC: Power-aware utilization control in distributed real-time systems," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 3, pp. 302–315, August 2010.

[12] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 3, pp. 316–328, August 2010.

[13] H. Wang, L. Hu, X. Guo, Y. Nie, and H. Tang, "Compact piecewise linear model based temperature control of multi-core systems considering leakage power," *IEEE Trans. on Industrial Informatics*, vol. 16, no. 12, pp. 7556–7565, December 2020.

[14] H. Wang, X. Guo, S. X.-D. Tan, C. Zhang, H. Tang, and Y. Yuan, "Leakage-aware predictive thermal management for multi-core systems using echo state network," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 7, pp. 1400–1413, July 2020.

[15] D. Huang, A. Pahlevan, L. Costero, M. Zapater, and D. Atienza, "Reinforcement learning-based joint reliability and performance optimization for hybrid-cache computing servers," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 12, pp. 5596–5609, December 2022.

[16] H. Khdr, M. E. Batur, K. Zhou, M. B. Sikal, and J. Henkel, "Multi-agent reinforcement learning for thermally-restricted performance optimization on manycores," in *Proc. Design, Automation and Test in Europe Conf. (DATE)*, March 2024.

[17] B. Dietrich, H. Khdr, and J. Henkel, "Centralized training and decentralized control through the actor-critic paradigm for highly optimized multicores," in *Proc. Design Automation Conf. (DAC)*, June 2025.

[18] S. Lee, K. Kang, and C.-M. Kyung, "Runtime thermal management for 3-D chip-multiprocessors with hybrid SRAM/MRAM L2 cache," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 3, pp. 520–533, March 2015.

[19] A. Asad, O. Ozturk, M. Fathy, and M. R. Jahed-Motlagh, "Optimization-based power and thermal management for dark silicon aware 3D chip multiprocessors using heterogeneous cache hierarchy," *Microprocessors and Microsystems*, vol. 51, pp. 76–98, June 2017.

[20] L. Siddhu, R. Kedia, and P. R. Panda, "CoreMemDTM: Integrated processor core and 3D memory dynamic thermal management for improved performance," in *Proc. Design, Automation and Test in Europe Conf. (DATE)*, March 2022, pp. 1377–1382.

[21] S. Niknam, Y. Shen, A. Pathania, and A. D. Pimentel, "3D-TTP: Efficient transient temperature-aware power budgeting for 3D-stacked processor-memory systems," in *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, June 2023.

[22] H. Wang, W. Li, W. Qi, D. Tang, L. Huang, and H. Tang, "Runtime performance optimization of 3-D microprocessors in dark silicon," *IEEE Trans. on Computers*, vol. 70, no. 10, pp. 1539–1554, October 2021.

[23] C.-H. Liao, C. H.-P. Wen, and K. Chakrabarty, "An online thermal-constrained task scheduler for 3D multi-core processors," in *Proc. Design, Automation and Test in Europe Conf. (DATE)*, March 2015.

[24] H. Wang, D. Huang, R. Liu, C. Zhang, H. Tang, and Y. Yuan, "STREAM: Stress and thermal aware reliability management for 3-D ICs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 11, pp. 2058–2071, November 2019.

[25] X. Zhou, J. Yang, Y. Xu, Y. Zhang, and J. Zhao, "Thermal-aware task scheduling for 3D multicore processors," *IEEE Trans. on Parallel and Distributed Systems*, vol. 21, no. 1, pp. 60–71, 2010.

[26] A. K. Coskun, J. L. Ayala, D. Atienza, T. S. Rosing, and Y. Leblebici, "Dynamic thermal management in 3D multicore architectures," in *Proc. Design, Automation and Test in Europe Conf. (DATE)*, April 2009, pp. 1410–1415.

[27] 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 *Proc. Int. conf. on Hardware/software codesign and system synthesis (CODES+ISSS)*, 2014.

[28] H. Wang, W. He, Q. Yang, X. Peng, and H. Tang, "DBP: Distributed power budgeting for many-core systems in dark silicon," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 12, pp. 5727–5731, December 2022.

[29] M. S. Mohammed, A. Al-Dhamari, M. Hamdan, A.-M. H. Y. Saad, A. S. H. Abdul-Qawy, and M. N. Marsono, "3D-DNaPE: Dynamic neighbor-aware performance enhancement for thermally constrained 3D many-core systems," *IEEE Access*, vol. 11, pp. 131 964–131 978, 2023.

[30] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *Proc. Design Automation Conf. (DAC)*, June 2015.

[31] H. Wang, J. Wan, S. X.-D. Tan, C. Zhang, H. Tang, Y. Yuan, K. Huang, and Z. Zhang, "A fast leakage-aware full-chip transient thermal estimation method," *IEEE Trans. on Computers*, vol. 67, no. 5, pp. 617–630, May 2018.

[32] H. Sultan and S. R. Sarangi, "A fast leakage-aware Green's-function-based thermal simulator for 3-D chips," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 11, pp. 2342–2355, November 2020.

[33] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[34] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *Proc. Int. Symp. on Computer Architecture (ISCA)*, June 1995, pp. 24–36.

[35] W. Huang, K. Sankaranarayanan, K. Skadron, R. J. Ribando, and M. R. Stan, "Accurate, pre-RTL temperature-aware processor design using a parameterized, geometric thermal model," *IEEE Trans. on Computers*, vol. 57, no. 9, pp. 1277–1288, 2008.

[36] J. Meng, K. Kawakami, and A. K. Coskun, "Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints," in *Proc. Design Automation Conf. (DAC)*, June 2012, pp. 648–655.

[37] A. Pathania and J. Henkel, "HotSniper: Sniper-based toolchain for many-core thermal simulations in open systems," *IEEE Embedded Systems Letters*, vol. 11, no. 2, pp. 54–57, June 2019.

Hai Wang received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China in 2007, and the M.S. and Ph.D. degree from the University of California, Riverside, USA in 2008 and 2012, respectively.

He is now a full professor with the University of Electronic Science and Technology of China, Chengdu, China. His research interests include design automation of VLSI circuits and systems, software/hardware codesign of computer systems, and artificial intelligence algorithm and hardware.

Dr. Wang was a recipient of the Best Paper Award nomination from Asia and South Pacific Design Automation Conference (ASP-DAC), the First Class Award of Science and Technology Progress Award of Sichuan Province, and the Huawei Excellent Collaboration Project Award.

He has served as an Organizing Committee Member of ICCD, Technical Program Committee Member of ICCAD, DATE, ASP-DAC, IGSC and ISQED, and also served as a Reviewer of many journals including the IEEE TC, IEEE TCAD, IEEE TPDS, and ACM TODAES.

Yanhong Luo received the B.S. degree from Sichuan University, Chengdu, China, in 2024. He is currently pursuing the M.S. degree at the University of Electronic Science and Technology of China (UESTC), Chengdu, China. His current research interests include power analysis and thermal management of integrated circuits.

Jincheng Guo received the M.S degree from school of integrated circuit science and engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2024. His reseach focuses is on thermal analysis, power analysis, and design for test.

Yu Chen received the B.S. degree in Thermophysics, in 2004 and M.S. degree, in 2006, from Beihang University, Beijing, China. He is an expert of Huawei, responsible for Thermal Integration Architecture Design and related technology research, with about 20 years of professional experience.

Jiming Li received M.S. degree in Thermal Energy and Power Engineering from Northwestern Polytechnical University, Xi'an, China in 2012. He is a Senior Thermal Architect in Huawei Hisilicon. He mainly engaged in research on chip temperature reconstruction, thermal architecture definition, and dynamic thermal management.