GDP: A Greedy Based Dynamic Power Budgeting Method for Multi/Many-Core Systems in Dark Silicon

Hai Wang, Diya Tang, Ming Zhang, Sheldon X.-D. Tan, *Senior Member, IEEE*, Chi Zhang, He Tang, *Member, IEEE*, and Yuan Yuan

Abstract—Dark silicon phenomenon is significant in today's multi/many-core systems manufactured using new generation technology. In order to enhance performance of dark silicon systems, power budget constrained dynamic optimizations are performed in various ways including dynamic voltage and frequency scaling (DVFS) and task scheduling. However, power budgets given by existing methods are generally over pessimistic, which greatly limit the capability of dynamic performance optimization methods. In order to resolve this problem, we propose a dynamic power budgeting method, called Greedy based Dynamic Power (GDP). Different from existing methods, which are steady state based and ignore active core distributions, GDP formulates the power budgeting problem as a thermal-constrained combinational power optimization problem. To efficiently solve this problem, we propose two new ideas: first, we transform the original power-optimization problem to an easier solving temperature-optimization problem; second, we employ a more efficient greedy based algorithm that finds a sub-optimal active core distribution which maximizes power budget. The new method can consider current temperature states and transient thermal effects, which were ignored by existing methods. Both theoretical studies and experimental results show that GDP outperforms existing methods by providing a higher and less pessimistic power budget with low computing cost and guaranteed thermal safety.

Index Terms—Power budget, dark silicon, multi/many-core system

1 Introduction

One of the most impressive technology advances in the past fifty years comes from integrated circuit (IC), since IC integration density has been increasing in an exponential speed constantly for a long time, as described by the famous Moore's law. However, power density begins to rise with integration density after the breakdown of Dennard scaling, resulting in serious thermal related problems in IC systems [1]. Especially in recent years, power density has increased beyond the power wall caused by limited heat dissipation capability of the chip, leading to the fact that not all components of the system can be activated or run at full speed at the same time. Such system is called dark silicon system or system in dark silicon [1], [2], [3], [4], [5].

For today's multi/many-core systems in dark silicon, although thermal constraint cannot be altered due to reliability concerns, several dynamic optimizations can be applied in order to enhance system performance [6], [7]. For example, the voltage and frequency levels (VF levels) of the active cores can be adjusted [8], [9], [10], and tasks can be scheduled according to the active core number and their VF levels [11], [12], [13], [14], [15], [16]. These optimizations are clearly constrained by the maximum power consumption allowed for each active core, because the VF level and tasks running on the core determine power consumption, which should never exceed such maximum power to avoid thermal emergencies.

Determining the maximum allowed power is the fundamental step in order to improve the performance of a multi/many-core system in dark silicon. Such maximum allowed power of a system under practical constraints (like thermal constraint) is called *power budget*, and the process of finding the power budget is called power budgeting. The mathematical power budgeting problem formulation will be given later in Section 4.1.

It is very challenging to compute power budget dynamically for dark silicon systems due to high computational cost and overhead, because there are a lot of potential active core distributions in dark silicon system. As a result, existing power budgeting methods like TDP [17] and TSP [18] have to consider the thermally worst distribution case, and compute the corresponding pessimistic power budgets to ensure

- *H. Wang, D. Tang, C. Zhang, and H. Tang are with State Key Laboratory of Electronic Thin Films and Integrated Devices, University of Electronic Science and Technology of China, Chengdu 610054, China, and also with the School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China. E-mail: {wanghai, zhangc, tanghe}@uestc.edu.cn, diya_1@163.com.*
- *M. Zhang is with Cadence Design Systems, Inc, Shanghai 201204, China. E-mail: zhangming3804@cadence.com.*
- *S. X.-D. Tan is with the Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521. E-mail: stan@ee.ucr.edu.*
- *Y. Yuan is with School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China. E-mail: yuany@uestc.edu.cn.*

Manuscript received 31 Mar. 2018; revised 27 Sept. 2018; accepted 7 Oct. 2018. Date of publication 15 Oct. 2018; date of current version 19 Mar. 2019. (Corresponding author: Hai Wang).
Recommended for acceptance by J. D. Bruguera.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2018.2875986

system's absolute safety. In addition, only steady state power and thermal conditions are considered in the existing methods. It is well known that chip temperature may change over $50^{\circ}C$ (for example, from $40^{\circ}C$ to $90^{\circ}C$) during everyday use. Current temperature of the chip has huge impact on how much power it can consume under the thermal constraint: obviously, a system at $40^{\circ}C$ has much higher power budget than that of the same system at $90^{\circ}C$ for a future time duration. Although performance boost (Intel calls it turbo boost [17] which allows the system to consume a power higher than TDP for a short period of time) can be used to partially relieve the problem, the true power budget may not be fully utilized as shown both logically and experimentally in [18].

In this work, we propose a Greedy based Dynamic Power budgeting method (GDP) to solve the problems in existing power bugeting methods for dark silicon. This work has the following major contributions.

- First, in order to reduce computational complexity to enable dynamic power budgeting, we develop a greedy based method to find the sub-optimal active core distribution (which maximizes the power budget) and its corresponding power budget at runtime.
- Second, instead of providing a pessimistic steady state power budget like existing works, we show how to systematically take transient thermal effect into consideration, and compute a higher but still safe power budget which is adaptive to system's current temperature conditions. Such power budget is very accurate since it is computed using the explicit expression of chip's future temperature without approximation.

The contributions above lead to many advantages of the newly proposed methods over existing methods.

- First of all, because active core distribution is considered, GDP is able to provide a higher and less pessimistic power budget compared with TSP which can only provide over pessimistic power budget according to the worst case active core distribution.
- In addition, as a greedy based method, GDP has low computational complexity, which makes it capable of providing accurate power budget at runtime.
- Last but not least, being able to consider the transient temperature effect, GDP provides accurate power budget which automatically adapts to current running conditions of the system. We show that transient effect from both chip and package has huge impact on power budget, and the power budget provided by GDP can be significantly higher than the one provided by steady state power budgeting method like TSP when the system is at low temperature.

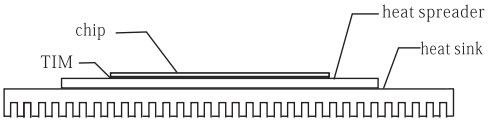
The remaining part of this article is organized as follows. We first review related work in power budgeting of IC systems in Section 2. In Section 3, modeling of the multi/many-core packaged IC system, which serves as prerequisite knowledge for power budgeting, is briefly introduced. Then, the new greedy based power budgeting method is presented in Section 4. Next, experiments used to verify the effectiveness and analyze the performance of the new method are demonstrated in Section 5. Finally, conclusion is drawn in Section 6.

2 Related Work

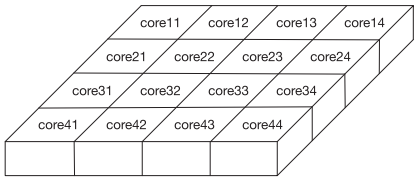
In this section, we review the important work in power budgeting of IC systems.

Since thermal issue has been the major performance limiting factor for many years, many dynamic thermal management (DTM) methods for IC systems have been proposed to optimize system performance under thermal constraint. There are methods that employ dynamic voltage and frequency scaling (DVFS) to limit the system power consumption in order to keep the temperature under thermal constraints for both single-core system [19], multi-core system [9], and 3D IC [20]. There are also researches that use task migration to switch heavy and light tasks to meet the temperature constraint [11], [12], [13], [14], [16], [21]. In [22], techniques in terms of core throttling and process migration policies have been proposed to perform DTM based on basic control methods. A distributed DTM scheme was proposed in [23] for thermally efficient on-chip network design. Work in [24] introduces a hardware/software co-design architecture to improve the DTM efficiency and accuracy. Real systems have implemented the DTM techniques mentioned above. For example, IBM POWER5 is equipped with a dual staged DTM using 24 digital temperature sensors [25]. IBM POWER7 offers per-core frequency scaling and multiple voltage islands to enable fine grained DTM in multi-core processor [26]. Rather than performing DTM relying on pure thermal sensor readings, there are works that perform thermal management based on the future temperature prediction of the processor [15], [27]. NADTM [14] was proposed to further consider the neighbor core's temperature impact in the DTM temperature prediction process for multi-core systems. However, these methods are not designed considering dark silicon, and may have problems when they are directly applied to dark silicon systems (for example, please see experiments using NADTM in Section 5.2). In order to reduce the large computing overhead, scalable DTM [28], power management [29], [30], and scheduling schemes [31] were recently proposed for many-core systems. Resource management technique was proposed in [32] for heterogeneous tiled multi-core systems by considering power density.

Power budgeting is important and challenging for dark silicon multi/many-core systems under temperature constraint [33]. TDP was proposed for general power budgeting which calculates the lowest total power of the system which may violate thermal constraint, and use such power as the power budget of the whole system [17]. Although being able to provide a safe power budget, over pessimistic is the main drawback of TDP, especially for multi-core dark silicon systems. This is because, in order to ensure the absolute safety, TDP has to consider multi-core dark silicon system at all running conditions (i.e., different number of active cores and all combinations of active core positions), and choose the lowest power budget from all these cases as the final TDP power budget. Clearly, such power budget is lower than the real allowable power for most running conditions. Performance enhancement technique by powering on more cores than the TDP allowed ones was proposed in [34] which also considers process variation. Different from TDP which provides only one power budget for all conditions, TSP puts active core number into consideration [18]. If the core distribution is given in advance, TSP calculates the maximum allowed



(a) Side view of the chip package structure



(b) Floorplan example of a 16-core dark silicon chip.

Fig. 1. Typical structure of a packaged multi/many-core dark silicon system.

power budget for this distribution. Otherwise, TSP provides a power budget by calculating the allowed maximum power for the thermally worst case active core distribution, so that all other active core distributions will also be safe under such power. Although TSP provides better power budgets compared to TDP, it is still over pessimistic, owing to two reasons. First, TSP only takes active core number and fixed active core distribution (if given in advance) into account, without considering different active core distributions. So the given power budget must be lower than the real maximum allowed powers. In addition, TSP, similar to TDP, is a static method, which considers only steady state power and thermal relationships. However, a system with low temperature, is actually able to consume a power higher than the steady state power budget, for a future time duration. In our experiment, we show the power budget difference can be huge if the transient temperature state of the system (with both chip and package) is considered. In [35], our previous work shows a heuristic based method to locate the active core position in order to maximize the power budget of the dark silicon systems. However, it requires large computing time and such computing time is not consistent since heuristic is used.

3 Modeling of Multi/Many-Core Dark Silicon System

In order to determine the power budget of a multi/many-core dark silicon system, power and temperature models of the system are presented in this section.

In this work, the multi/many-core dark silicon system is packaged in a common structure shown in Fig. 1a. Heat (power) generated from the chip is conducted through thermal interface material (TIM), heat spreader, heat sink, and finally dissipated to the air through convection. We ignore the secondary heat path structure here, since much fewer heat is dissipated through that path [36].

To perform thermal analysis for the packaged IC chip, we usually divide both the chip and its package into multiple blocks called thermal nodes, with partition granularity determined by the accuracy requirements. The power consumptions of the system are modeled as current sources, with currents flowing into the appropriate thermal nodes. For the dark silicon multi/many-core system (a 16-core chip's floorplan example is shown in Fig. 1b), we treat each core as a thermal node with a current source, because each core has very small area and highly correlated internal power distribution. The other thermal nodes from the package are divided according to the chip thermal nodes. Then, the thermal resistance and capacitance among all these thermal nodes are determined, which model the thermal transport and power response behaviors. Please note that multi/many-core system floorplans different from the one shown in Fig. 1b are fully compatible with this work, and each core's internal structures can also be modeled if necessary [36], [37].

By using the knowledge above, for an n-core system with m total thermal nodes, we can generate its thermal model as

$$\begin{aligned} GT(t) + C\dot{T}(t) &= BP(t), \\ T_c(t) &= B^T T(t), \end{aligned} \tag{1}$$

where $T(t) \in \mathbb{R}^m$ is the temperature rise vector, representing temperature rises (from the ambient temperature) at m places of the chip and package; $G \in \mathbb{R}^{m \times m}$ and $C \in \mathbb{R}^{m \times m}$ contain equivalent thermal resistance and capacitance information respectively; $B \in \mathbb{R}^{m \times n}$ stores the information of how chip powers are injected into the thermal nodes; $P(t) \in \mathbb{R}^n$ is the power vector, which contains power consumptions of n cores of the chip; $T_c(t) \in \mathbb{R}^n$ is the output temperature rise vector, containing temperature rises (from the ambient temperature) of the cores only.[1] For detailed structures of G, C, and B matrices, please refer to the thermal modeling works such as [36], [37], [38], [39].

With the model in (1), the task of power budgeting is to determine the appropriate P, with a given temperature threshold, as described next.

4 Greedy Based Power Budgeting for Multi/Many-Core Dark Silicon Systems

Power budgeting, by the name, provides a power budget, which serves as a guidance and regulation constraint for the system. One important property of power budgeting is that the given power budget should be conservative such that the system temperature does not violate the threshold by strictly following the power budget. Because of such conservative consideration and lack of dynamic budgeting capability, existing power budgeting methods are over pessimistic, with the given power budget as the maximum allowed *worst case* power under thermal constraint [18]. In this section, we show that power budget can be conservative but not over pessimistic, by introducing a dynamic power budgeting method GDP. As a dynamic method, GDP no longer considers the worst case, but directly provides maximum allowed power under thermal constraint, with low computing cost and conservative property.

First, we formulate the standard power budgeting problem and transform it into an equivalent problem of maximizing chip temperatures in Section 4.1. Next, we show

1. Temperature rises of the package thermal nodes are not outputted because we do not need them explicitly in this power budgeting problem. If they are explicitly needed (in some other applications), just simply output T.

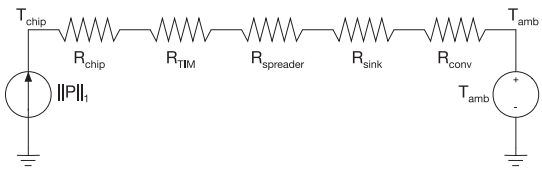


Fig. 2. The equivalent heat dissipation circuit for normally packaged IC chip.

how to perform the proposed greedy based power budgeting for steady state problem in Section 4.2. Since the new method is dynamic based, the steady state case serves mainly as demonstration for easier understanding. Then, the full dynamic version of GDP considering transient effects is presented in Section 4.3. Finally, how GDP adapts to some important practical situations is given in Section 4.7.

4.1 Power Budgeting Problem Formulation

4.1.1 Standard Power Budgeting by Maximizing Total Power

As discussed before, GDP, as a dynamic method, describes power budgeting task as finding the highest allowed total core power under temperature constraint. Some systems may have other constraints such as total power supply limit. But for dark silicon systems, which are extremely temperature limited, we focus on the major problem of thermal limits, and other constraints can be added with minor modification if needed. For simplicity, let us first consider the steady state case, and the power budgeting problem can be formulated as the following optimization problem

$$\begin{aligned} &\text{maximize} \quad \|P\|_1 \\ &\text{subject to} \quad \begin{cases} \text{card}(P) = n_a, \\ T_c \preceq T_{th}, \end{cases} \end{aligned} \tag{2}$$

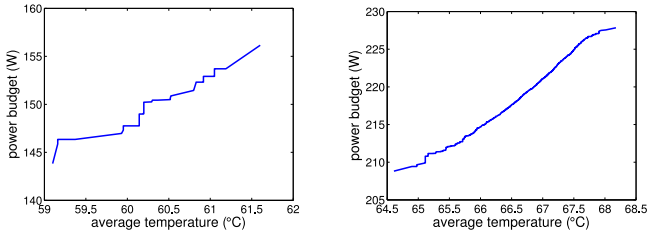
where $T_{th} \in \mathbb{R}^n$ is the temperature rise threshold vector containing the maximum allowed temperature rises from the ambient temperature; $\text{card}(P)$ means the cardinality or size of the vector P, which is defined as the number of nonzero components in P. In our case, $\text{card}(P) = n_a$ means there are n_a active cores.

4.1.2 Power Budgeting by Maximizing Chip Temperatures

However, the optimization problem above is a combinational problem, which is very difficult to solve because of the high computational complexity. To solve this problem with low complexity, we find out that we can formulate an easier alternative problem in which we treat core temperature rise T_c as a resource which is limited by temperature rise constraint T_{th}. In the new problem, we seek to allocate power budgets to cores, such that the temperature resource is fully used. In another word, we prefer the power budget induced temperatures of all active cores to reach the thermal constraint limits without violating it.

To show the two mentioned problems are equal, we have the following proposition:

Proposition 1. *Providing a normally packaged IC chip, a higher average temperature of the chip $T_{chip} = \frac{1}{n}\|T_c\|_1 + T_{amb}$ means a higher total power of the chip $\|P\|_1$, where T_{amb} represents the ambient temperature.*



(a) 9-core system with 4 active cores. All 126 possible active core distributions are plotted.

(b) 16-core system with 8 active cores. All 12870 posible active core distributions are plotted.

Fig. 3. Average temperature versus power budget plot for all possible active core distributions of two dark silicon systems. Both figures reveal that higher average chip temperature leads to higher power budget.

Proof. The proposition above is proved in the following way. The typical structure of a packaged chip is shown in Fig. 1a. Since most heat generated in chip is conducted through chip, thermal interface material, heat spreader, heat sink, and finally dissipated to the air through convection, we form the equivalent thermal resistor circuit by connecting several equivalent thermal resistors in series shown in Fig. 2. These thermal resistors represent thermal resistance in the chip (R_{chip}), TIM (R_{TIM}), heat spreader ($R_{spreader}$), heat sink (R_{sink}), and convection from heat sink to the air (R_{conv}). The two thermal nodes at the two ends of the thermal circuit denote average temperature of the chip ($T_{chip} = \frac{1}{n}\|T_c\|_1 + T_{amb}$) and average temperature of the ambient air (T_{amb}). Because heat is only generated at the chip and only dissipated into the air, we have an equivalent current source with current value $\|P\|_1$ (the total power of the chip) connected at T_{chip} to represent power generation. Also, an equivalent voltage source is connected at T_{amb} to stand for the ambient air temperature. Now if we increase the average temperature of the chip, i.e., increase the value of $T_{chip} = \frac{1}{n}\|T_c\|_1 + T_{amb}$, the current flow $\|P\|_1$ in the thermal resistor circuit has to increase due to the fixed ambient air temperature T_{amb}. In another word, a higher average temperature of the chip $T_{chip} = \frac{1}{n}\|T_c\|_1 + T_{amb}$ leads to a higher total power of the chip $\|P\|_1$, which proves the proposition. □

In addition to the proof, we also provide experimental evidence to support this proposition. For the 9-core system with 4 active cores, we plot the power budget versus average temperature for all possible active core distributions (there are totally $\binom{9}{4} = 126$ active core distributions) in Fig. 3a. We can see that the power budget increases monotonically with the average temperature of the chip. We also plot such figure for the 16-core system with 8 active cores with totally $\binom{16}{8} = 12870$ possible active core distributions in Fig. 3b. It also supports the proposition that higher average chip temperature leads to higher power budget.

We have shown that higher average temperature of the chip means a higher total power, and the highest chip temperature rise limited by the heat dissipation capability is T_{th}. So, instead of directly maximizing total power $\|P\|_1$, it is natural to maximize the average temperature of the chip by

making T_c as close to the temperature rise threshold T_{th} as possible, which leads to the following optimization problem:

$$\begin{aligned} &\text{minimize} \ \ \|T_{th} - T_c\|_2 \\ &\text{subject to} \ \begin{cases} \text{card}(P) = n_a, \\ T_c \preceq T_{th}, \end{cases} \end{aligned} \tag{3}$$

The optimization problem above is also a combinational problem, which means finding its optimal solution is still computationally expensive. Fortunately, such problem can be solved efficiently by finding a sub-optimal solution using greedy based method, as shown next.

4.2 Greedy Based Power Budgeting in Steady State

In Section 4.1, we have shown that we can allocate power budget by making the average temperature of the chip as high as possible to enhance performance. Such a strategy is summarized as optimization problem (3). However, remaining as a combinational problem, finding the solution of (3) is still very difficult. To be specific, for an n-core system with n_a active cores, finding the optimal solution of the optimization problem (3) requires solving $\binom{n}{n_a}$ sub-problems, where each sub-problem corresponds to minimizing $\|T_{th} - T_c\|_2$ with a possible active core distribution which satisfies $\text{card}(P) = n_a$. In this section, we use steady state example to show that we can use a greedy based method to efficiently find a sub-optimal solution of (3), which only requires solving n_a sub-problems. Please note that GDP is a dynamic power budgeting method, and we present its behavior in steady state mainly for the purpose of clarity.

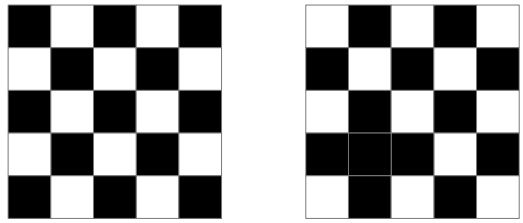
The steady state temperature rise of the chip can be calculated using model (1) by neglecting the differential term $C\frac{dT(t)}{dt}$, leading to

$$T_c = B^T G^{-1} B P. \tag{4}$$

Let $A = B^T G^{-1} B \in \mathbb{R}^{m \times m}$ to simplify notation, we plug (4) into the optimization problem (3) and get

$$\begin{aligned} &\text{minimize} \ \ \|T_{th} - AP\|_2 \\ &\text{subject to} \ \begin{cases} \text{card}(P) = n_a, \\ AP \preceq T_{th}. \end{cases} \end{aligned} \tag{5}$$

Finding the optimal solution of such optimization problem requires brute force search of all possible combinations of non-zero value positions in P which satisfies $\text{card}(P) = n_a$. For each combination, we can transform (5) into a constrained norm approximation problem. Although each constrained norm approximation problem can be solved efficiently, the combination number grows exponentially with the core and active core number, which makes solving (5) in traditional way impractical, as shown later in experiments. Another possible way of solving such problem is to eliminate the cardinality constraint by replacing the cost function by an l_1-norm regularized cost function $\|AP - T_{th}\|_2 + \gamma\|P\|_1$. By changing the value of γ, we can tune the sparsity in the solution P, and it is possible to find the one with $\text{card}(P) = n_a$ if we find the suitable γ [35]. Such method is faster than the traditional combination solution, however, it is still slow since a lot of values of γ have to be tested before we find the final solution.



(a) Optimal. (b) Sub-optimal.

Fig. 4. Optimal and sub-optimal active core distribution comparison using a 25-core system with 12 cores active. Cores in white are active and cores in black are off.

As discussed above, finding the optimal solution has high complexity which means it is not suitable for multi/many-core system with large number of cores. It is also noticed that for such systems, finding the optimal solution is *not* necessary. This is due to the fact that when core number is large, each core takes relatively small area, so there exist many sub-optimal active core distributions which only have slightly larger objective values (measured by cost function) than that of the optimal solution. For example, consider a 25-core system with 12 cores active. The optimal solution of such system is shown in Fig. 4a, and one sub-optimal solution is shown in Fig. 4b. Please note that white cores are active and black cores are off. We can see that the sub-optimal solution has a dark core cluster at one corner, but only takes a small chip area, meaning the sub-optimal solution has only slightly lower power budget than that of the optimal solution. This is also verified in our experiments by comparing the optimal power budget and sub-optimal power budget, as shown later.

Since a sub-optimal solution may have only slightly worse performance compared with the optimal solution, instead of finding the optimal solution using combinational method with high complexity, we seek for a fast method to find a sub-optimal solution.

For an n-core system with n_a active cores, the basic idea of finding such sub-optimal solution is described as follows: we first find the optimal solution for only one active core. Next, we *fix* the first active core position determined by the first step, and find the optimal solution of two cores, with the second active core position determined. Please note that although we say "optimal" in the second step, such solution is only the optimal solution with the first active core fixed at the position determined by the first step, but not the true optimal solution for general two active cores. Similarly, in the $(i+1)$th step, we look for the optimal solution for $i+1$ active cores with the positions of i active cores found in all previous steps remain fixed. By proceeding such strategy for n_a steps, we can arrive at a sub-optimal solution for n_a active cores.

Now let us explain the steps in details.

4.2.1 Find the Solution for One Active Core

For the first step, since we only need to find the optimal solution for one active core, the optimization problem is

$$\begin{aligned} &\text{minimize} \ \ \|T_{th} - AP\|_2 \\ &\text{subject to} \ \begin{cases} \text{card}(P) = 1, \\ AP \preceq T_{th}. \end{cases} \end{aligned} \tag{6}$$

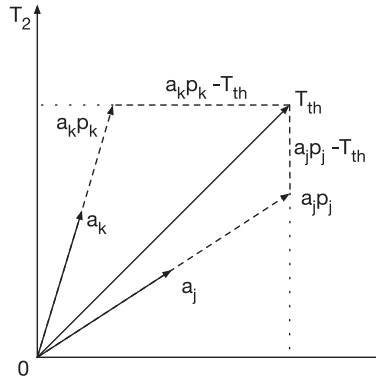


Fig. 5. Illustration of finding the first active core in the greedy based algorithm. For simplicity, only two cores (the jth core and the kth core) are compared in the figure. The T_1 and T_2 axes represent temperature rises at two positions of the chip, respectively.

We normalize columns of A to simplify discussions (please note we can reverse such operation on solution P to get the original solution), i.e., for $A = [a_1, a_2, \ldots, a_n]$, we make $\|a_1\|_2 = \|a_2\|_2 = \cdots = \|a_n\|_2 = 1$ without changing their directions.

In order to solve such problem, we first determine the optimal position for the single active core. Since there are n possible cores to be chosen as the first active core, we have to find a way to determine if one core is superior to the other one, measured by the cost function in (6).

One example of comparing two possible active core positions (we take the jth core and kth core as example) is shown in Fig. 5. Due to limitation of graph drawing, only temperature rises (denoted by T_1 and T_2 axes in the figure) at two positions on the chip are shown in the figure. By turning on the jth core only, the corresponding temperature rise of the chip is $a_j p_j$ (since all other cores are turned off with no power), where p_j is the jth element in P which should be determined by solving (6). Assume p_j is correctly computed, then the T_1 component (chip temperature rise at position 1) of $a_j p_j$ will just be the same as the threshold, and the T_2 component (chip temperature rise at position 2) of $a_j p_j$ will be lower than threshold. Instead, by turning on the kth core only, the corresponding solved power value p_k will heat the chip to $a_k p_k$, with its T_1 component being lower than threshold and its T_2 component just being equal to the threshold. Although both cases have temperature rise at one position reaching the threshold and temperature rise at the other position lower than threshold, we prefer to turn on the jth core, because its cost $\|T_{th} - a_j p_j\|_2$ (length of $a_j p_j - T_{th}$ in Fig. 5) in the optimization problem (6) is smaller than the cost $\|T_{th} - a_k p_k\|_2$ of turning on the kth core, as observed in Fig. 5. The physical meaning is: by turning on the jth core, the overall system temperature is closer to the temperature threshold, thus higher than that of turning on the kth core.

Actually, we do not even need to solve the corresponding power (p_j and p_k in this example) to compare two possible active cores. In Fig. 5, we can see that the jth core is preferred over the kth core, because a_j is closer to T_{th}, i.e., the angle between a_j and T_{th} is smaller (please note that all columns of A are normalized). As a result, in order to determine the optimal active core numerically, a good indicator to approximately measure such angle with low computing cost is the inner product of a column in A and T_{th}. In another word, we can simply compute the inner products of all columns of A and T_{th}, i.e., $\langle a_j, T_{th} \rangle$ for $j = 1, 2, \ldots, n$, and pick the core with the largest inner product as the first active core.

Now, with the first active core position fixed, we can compute the power budget of that core. Assume the jth core is picked, then optimization problem in (6) changes to

$$\begin{aligned} &\text{minimize} \ \ \|T_{th} - a_j p_j\|_2 \\ &\text{subject to} \ \ a_j p_j \preceq T_{th}. \end{aligned} \tag{7}$$

This optimization problem can be equivalently transformed into a quadratic programming (QP) problem

$$\begin{aligned} &\text{minimize} \ \ p_j^2 a_j^T a_j - 2 p_j T_{th}^T a_j + T_{th}^T T_{th} \\ &\text{subject to} \ \ a_j p_j \preceq T_{th}, \end{aligned} \tag{8}$$

which is then solved efficiently by calling the standard QP solving routines for the power budget p_j. Please note that if the total active core number is one ($n_a = 1$), then p_j is the final power budget. Otherwise, p_j is just a temporary value and will be updated in the subsequent procedures as shown in Sections 4.2.2 and 4.2.3.

4.2.2 Find the Solution for Two Active Cores

After finding the position and power budget for only one active core, we can proceed to the second step: find the solution for two active cores. As a greedy based algorithm, we fix the first active core position found in the previous step, and look for the second core position only. First, we subtract the first active core induced temperature rise from T_{th} to get the remaining temperature rise threshold as $T_{rm} = T_{th} - a_j p_j$, assuming the jth core is the active core already chosen in the first step. The physical meaning of T_{rm} is interpreted as: elements in T_{rm} with large values mean that the first active core has small temperature impact on these positions, and vice versa. As a result, in order to find the position of the greedy optimal second active core, we need to find the core which complements the impact of the first active core best. So we test the other $n - 1$ cores one by one through computing the inner products of its corresponding column in A and T_{rm}. The core with the largest inner product is chosen as the second active core.

Next, we need to compute the new power budget with the second active core position added. Assume the kth core is picked as the second active core, the power budget can be found by solving the following optimization problem:

$$\begin{aligned} &\text{minimize} \ \ \|T_{th} - [\, a_j \quad a_k \,] \begin{bmatrix} p_j \\ p_k \end{bmatrix} \|_2 \\ &\text{subject to} \ \ [\, a_j \quad a_k \,] \begin{bmatrix} p_j \\ p_k \end{bmatrix} \preceq T_{th}, \end{aligned} \tag{9}$$

by transforming it into the equivalent QP problem similar to (8). Please note that T_{th} is used here as the temperature threshold for power budget computing, whereas T_{rm} is only employed to locate the new active core position.

4.2.3 The General Iterative Steps

Previously, we have shown the steps to find the position and power budget for the first two active cores. Now, we

extend the previously introduced steps into the general iterative steps which are used to find the distribution of the n_a active cores. Assume we have already found i active cores. The corresponding i columns of A are collected into matrix $A_i \in \mathbb{R}^{i\times n}$, and power budget of these i cores are expressed as vector $P_i \in \mathbb{R}^{i\times 1}$. Then, we can form the following optimization problem to describe the power budgeting problem with these i active cores:

$$\begin{aligned} &\text{minimize} \ \ \|T_{th} - A_i P_i\|_2 \\ &\text{subject to} \ \ A_i P_i \preceq T_{th}. \end{aligned} \tag{10}$$

The power budget P_i is readily solved by changing (10) into the equivalent QP problem as

$$\begin{aligned} &\text{minimize} \ \ P_i^T A_i^T A_i P_i - 2T_{th}^T A_i P_i + T_{th}^T T_{th} \\ &\text{subject to} \ \ A_i P_i \preceq T_{th}. \end{aligned} \tag{11}$$

In order to find the position of the $(i+1)$th core, we subtract the temperature rise caused by power budget of i active cores from T_{th}, and obtain $T_{rm} = T_{th} - A_i P_i$. Next, in order to check which core left has the greatest potential in complementing the impact of the existing i active cores, we compute the inner products of the remaining columns of A and T_{rm}. The core with the largest inner product is picked. The position of the newly picked core is recorded and the corresponding column in A is added into A_i to form the new matrix A_{i+1}.

Then, the new power budget with $i+1$ active cores can be computed by solving the $i+1$ version of (11). If $i+1 = n_a$, we stop the iteration and output the positions of all n_a active cores and the computed power budget. If $i+1 < n_a$, we continue this iteration to find the $(i+2)$th active core.

4.3 Greedy Based Power Budgeting Considering Transient Effects

The power budgeting method introduced in Section 4.2 suits steady state condition and serves mostly for illustration of the basic ideas of GDP. It may work just fine on the aspect of reliability if the computed steady state power budget is followed strictly during the power management process. However, it still suffers from two issues, which are also shared by the existing power budgeting methods like TDP and TSP.

The first issue is the inability to correct or ease bad or false power management decisions which violate the steady state power budget. If the previous temperature threshold is already violated at current time because of previous bad or false power management, even performing correct power management following the steady state power budget may worsen the reliability. Obviously, the suitable way of dealing with such situation is to use a different power budget which is lower than the steady state one, or even shut down some/all cores if necessary.

The second issue is that the steady state power budget is over pessimistic for most of the time in real world, and as a result, performance will be sacrificed. This problem is significant when the chip previously ran at low performance mode, and suddenly requires high performance. Because the temperature of the chip is way below the emergency temperature at current time, the system is able to consume much more power than the steady state power budget without violating the temperature constraint.

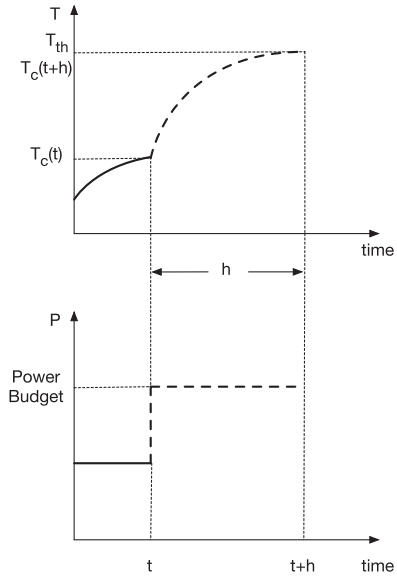


Fig. 6. Demonstration of how the impact of current temperature (at time t in the figure) is incorporated into the power budgeting method. The dashed lines represent power budget and the budget caused temperature rise of a core (assume this core is already chosen to be active by GDP).

In real world, it is possible that power management makes false decisions occasionally, and it is frequent that chip switches between low performance mode and high performance mode, so we develop a power budgeting method by considering current transient thermal/power behaviors as follows.

First, we use Fig. 6 to illustrate how GDP takes the current temperature impacts into account for power budgeting process. Note that only one active core's temperature and power budget are shown in the illustration for simplicity. At current time t, temperature rise of the core is $T_c(t)$, which is below temperature rise threshold T_{th}. GDP needs to find the power budget P (shown in dashed power line) for the future time duration h, such that by applying the budget to the core, temperature rise at $t+h$ (i.e., $T_c(t+h)$) should just reach temperature rise threshold T_{th} without violating it (shown in dashed temperature line).

From the previous illustration, we notice that the basic problem in dynamic power budgeting is to represent temperature rise $T_c(t+h)$ using the input power P and initial temperature rise $T(t)$. Once the formulation of $T_c(t+h)$ is available, we can put it into the basic optimization problem (3), and get the sub-optimal P using the similar greedy based method presented in Section 4.2.

However, representing $T_c(t+h)$ using P and $T(t)$ is not straightforward. This is because time domain convolution is required to express $T_c(t+h)$ explicitly, and practically, Euler method [37] and Runge-Kutta method [36] are used to compute $T_c(t+h)$ numerically instead of formulating the explicit $T_c(t+h)$. Fortunately, we are able to obtain the *closed-form* expression of $T_c(t+h)$ with P and $T(t)$, by taking advantage of the fact that P is constant for the future time duration h in power budgeting problem, as presented next.

It is well known that thermal model as a linear system in (1) has the following solution

$$T(t+h) = e^{-hC^{-1}G}T(t) + \int_t^{t+h} e^{-(t+h-\tau)C^{-1}G}C^{-1}BP(\tau)\,\mathrm{d}\tau. \quad (12)$$

Because the power budget is constant for the future power budgeting cycle duration h (i.e., from time t to $t+h$), we can simplify (12) into

$$T(t+h) = e^{-hC^{-1}G}T(t) + \int_0^{h} e^{-(h-\tau)C^{-1}G}C^{-1}B\,\mathrm{d}\tau P. \quad (13)$$

Let us denote

$$M(h) = e^{-hC^{-1}G}, \quad N(h) = \int_0^h e^{-(h-\tau)C^{-1}G}C^{-1}B\,\mathrm{d}\tau,$$

then (13) can be written as

$$T(t+h) = M(h)T(t) + N(h)P. \quad (14)$$

Please note that $M(h) \in \mathbb{R}^{m\times m}$ and $N(h) \in \mathbb{R}^{m\times n}$ are *constant* matrices which can be computed *offline* for a given time step h.

$M(h)T(t)$ and $N(h)P$ are the zero-input response and zero-state response of the linear system in (1), respectively, and there are many easy ways to compute $M(h)$ and $N(h)$ in both frequency domain and time domain. Here we only briefly show a time domain method presented in [40]. Let us denote $M(h) = [M_1, M_2, \ldots, M_m]$, $N(h) = [N_1, N_2, \ldots, N_n]$, $T(0) = [T_1, T_2, \ldots, T_m]^T$, and $P = [p_1, p_2, \ldots, p_n]^T$, where M_i and N_i represent the ith columns of $M(h)$ and $N(h)$, T_i and p_i represent the ith elements of $T(0)$ and P, respectively. With linear system theory, we have $M_i = T(h)$, if we let $P = [0, 0, \ldots, 0]^T$, $T_i = 1$, and $T_j = 0$ for all $j \neq i$. As a result, we can get the ith column of $M(h)$ by computing $T(h)$ using standard numerical integration of (1) with only the ith element of initial state $T(0)$ set to be one. Similarly, we have $N_i = T(h)$, if we let $T(0) = [0, 0, \ldots, 0]^T$, $p_i = 1$, and $p_j = 0$ for all $j \neq i$. In another word, the ith column of $N(h)$ can be obtained by computing $T(h)$ using numerical integration of (1) with only the ith element of power P set to be one.

Finally, we are able to express $T_c(t+h)$ using P and $T(t)$ as

$$T_c(t+h) = B^T T(t+h) = B^T M(h)T(t) + B^T N(h)P. \quad (15)$$

The cost function of the steady state power budgeting problem in (5) will be changed by using the new $T_c(t+h)$ for transient case as

$$\|T_{th} - B^T M(h)T(t) - B^T N(h)P\|_2. \quad (16)$$

In order to simplify notation, we denote a new vector

$$\bar{T}_{th} = T_{th} - B^T M(h)T(t), \quad (17)$$

where $\bar{T}_{th}$ has similar meaning of T_{th} in steady state case, and it also accounts for the transient thermal effects, i.e., current temperature's impact on power budget. Also, we denote a new matrix

$$\bar{A} = B^T N(h), \quad (18)$$

which works similarly as matrix A in steady state case, but accounts for transient thermal effect. The power budgeting optimization problem for transient case is updated as

$$\begin{aligned} &\text{minimize } \|\bar{T}_{th} - \bar{A}P\|_2 \\ &\text{subject to } \begin{cases} \mathrm{card}(P) = n_a, \\ \bar{A}P \preceq \bar{T}_{th}. \end{cases} \end{aligned} \quad (19)$$

Now dynamic GDP can be applied by following the steps in steady state GDP presented in Section 4.2, just with (5) replaced by (19).

4.4 The GDP Algorithm

Now, we summarize the GDP algorithm as Algorithm 1.

Algorithm 1. The GDP Algorithm

Input: n, n_a, A, T_{th} (for transient, replace A and T_{th} with $\bar{A}$ and $\bar{T}_{th}$, respectively)
Output: power budget vector P

```
T_rm = T_th, A_0 = [ ]
for i = 1 : n_a do
    idx(i) = 1
    for j = 2 : n do
        if ⟨a_j, T_rm⟩ > ⟨a_idx(i), T_rm⟩ then
            idx(i) = j
        end if
    end for
    A_i = [A_{i-1}, a_idx(i)]
    Compute P_i by solving the QP problem (11)
    T_rm = T_th − A_i P_i
end for
return P = P_{n_a}
```

Since GDP algorithm is elegant and concise, it can be easily integrated with many thermal modeling tools. Let us take the popular thermal modeling tool HotSpot [36], [38] as example. First, HotSpot will build the thermal model matrices G, C, and B. Then, GDP in Algorithm 1 can be easily implemented with the A matrix (or $\bar{A}$ and $\bar{T}_{th}$ matrices) formulated using the thermal model matrices.

4.5 Overhead Reduction of the Greedy Based Power Budgeting

Computing overhead is important for dynamic power budgeting. Although GDP already reduces the non-polynomial time complexity of original combination problem to polynomial time complexity, the overhead of GDP still grows as the active core number and core number increase. Specifically, for an n-core system with n_a active cores, GDP needs n_a iterations to locate the positions of all active cores. In each iteration, finding the max inner product $\langle a_j, T_{rm}\rangle$ takes approximately n^2 operations, and computing the new power budget for the next iteration takes approximately n_a^3 operations. As a result, the time complexity of GDP is $O(n^2 n_a + {n_a}^4)$.

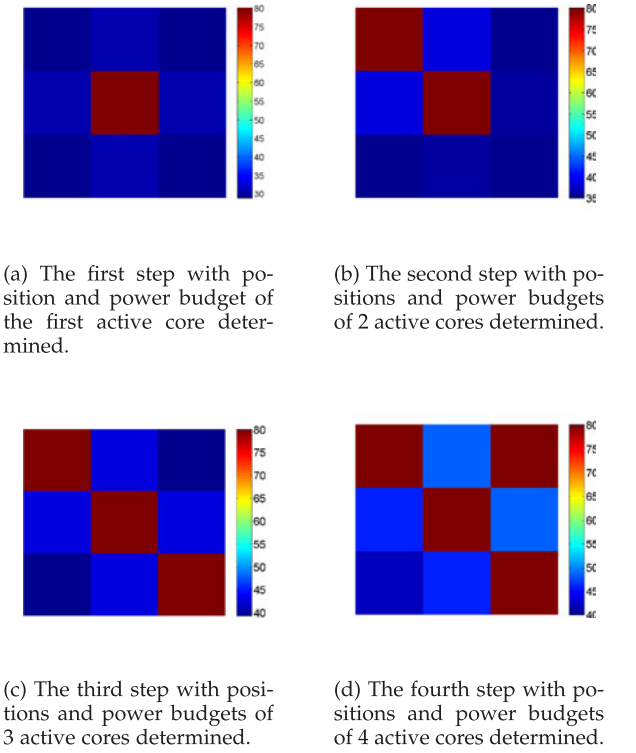
Please note that large computing overhead only appears when n and n_a are large. But since GDP is computed only on one of the n_a active cores, even such large overhead does not have significant impact on the total system throughput as shown in experiments. However, large overhead may still result in significant power budgeting delay. In this section, we propose two methods to relieve such overhead problem: one method is called reverse GDP for systems with large active core ratio, and another method is called parallel GDP for systems with large core number.

4.5.1 Reverse GDP to Locate Non-Active Cores

For a system with large active core ratio, GDP will locate the active cores one by one with a lot of iterations. For example, there will be 60 iterations for GDP to find the power budget for a 64-core system with 60 active cores. For such condition, we can actually locate the non-active cores instead of the active cores, in order to reduce the computing overhead. We call this method which aims to find non-active core positions rather than active core positions as *reverse GDP*.

The main idea of reverse GDP is to find the locations of non-active cores rather than active cores, in a greedy manner similar to the standard GDP. The steps of reverse GDP for one iteration are given as follows: First, calculate the power budget using the QP optimization (11), by assuming all cores are active except for the already located non-active cores in previous iterations (for initiation, simply assume all cores are active). Second, among all active cores, find the one (assume the jth core) with smallest $\|a_j p_j\|_1$. This core (the jth core) is the newly located non-active core in this iteration. At last, calculate the power budget for new distribution and go to the second step to start the next iteration until sufficient non-active cores are located.

Reverse GDP has lower computing overhead when the active core ratio is high. For the same 64-core system with 60 active cores, reverse GDP only needs 4 iterations compared to 60 iterations of standard GDP.



(a) The first step with position and power budget of the first active core determined.

(b) The second step with positions and power budgets of 2 active cores determined.

(c) The third step with positions and power budgets of 3 active cores determined.

(d) The fourth step with positions and power budgets of 4 active cores determined.

Fig. 7. Temperature distributions of the 9-core system with the power budget given by GDP's first four greedy steps.

4.5.2 Parallel GDP for Parallel Computing

The computing overhead of GDP also grows with core number. For the system with a large number of cores, we divide the original system into smaller sub-systems, and find the active core positions for each sub-system in parallel to reduce the computing overhead. We call this method as *parallel GDP*.

The main idea of parallel GDP is to divide the original system into several small sub-systems with moderate number of cores, such that the active cores can be located for each sub-system with standard GDP process in parallel. Since we only need to find the sub-optimal active core locations, the boundaries of each sub-system can be treated as adiabatic with tolerable error loss. The active core number in each sub-system can be set manually by assuming uniform active core distribution or by experience, as long as the total active core number is met. After all active core positions are located, we compute the power budget using the original system model (not the sub-system model anymore). As a result, the impact of the neighbor sub-systems is also accurately considered in parallel GDP. Please note that although the final power budget computing step is not a parallel process, it is still very fast because there is no iteration in this step.

Parallel GDP enables performing GDP even on a system with large number of cores. For example, if we apply parallel GDP to a 64-core system by dividing the original system into four 16-core sub-systems, the computing overhead is similar to applying standard GDP to a 16-core system.

4.6 Guidance on Estimating the Optimality Lower-Bound of GDP in Steady State

Since GDP provides sub-optimal solution, one natural question is how optimal is the GDP solution. However, deriving a theoretical optimality lower-bound for GDP is extremely difficult because the optimality of GDP differs with system core number, active core number, and the system thermal model (A matrix, etc.). Even checking the optimality of GDP is challenging because of the inability to obtain the optimal solution for just moderate sized multi-core systems: the computing complexity is so high that we cannot get the optimal solution for the 25-core system with 12 active cores even after over 12 hours' computing.

However, we still give a guidance on estimating the lower-bound of GDP in steady state. This guidance is based on the observation that the optimality of GDP increases with system core number n, simply because the active core area mismatch ratio between the GDP solution and the optimal solution is smaller with larger system core number n. For example, finding even one wrong active core position for a 9-core system can make a relatively large difference in power budget. However, finding even several wrong active core positions in a 1000-core system is not a big deal in power budget. As a result, we can seek for the largest difference between the GDP solution and the optimal solution in systems with small number of cores. Such difference could be the steady state optimality lower-bound for systems with different number of cores sharing the same package structure.

To be specific, for quad-core (2×2) system in steady state, the active core distribution given by GDP is absolutely optimal for any active core numbers. For the 3×3 core system, not all GDP solutions are optimal anymore because GDP is greedy based. Specifically, GDP solutions are non-optimal for active core number 2, 3, 4, as can be easily observed in Fig. 7. By comparing the steady state power budgets given by GDP and the optimal method, we find the 4 active core case has the largest power budget difference (1.5%). Thus, we can take such difference as the optimality

lower-bound of steady state GDP, which is supported experimentally by the data in Table 1.

4.7 Consideration of Practical Situation

The GDP method presented previously is designed with ideal assumptions. For practical situations, following such ideal GDP's suggestions may not lead to optimal performance.

One important example is that GDP determines active core positions every time it is activated. However, in practical situations, task migrations happen when active core position changes, with migration overheads. If task migration operation is performed too often, the aggregate overhead may grow too significant, making the game not worth the candle. In addition, a related practical situation is that we may prefer some communication extensive tasks to be performed in some adjacent cores, even though it leads to active core clusters with lower power budget (this is exactly what ideal GDP tries to avoid).

In order to take care of the aforementioned two practical situations, a forced active core map indicating which cores must be active can be given before the first iteration, thanks to GDP's greedy based iteration structure. For example, if migration overhead is significant for certain task, its current core should be added to the forced active core map, thus heavy migration cost can be avoided. If several tasks need to be performed in adjacent cores due to communication or other concerns, these adjacent cores should also be added to the forced active core map.

With the forced active core map provided (assume i cores are forced active), GDP will simply find the power budget P_i for these i active cores with their forced positions using (10) (or its dynamic version), and continue to find the remaining active cores (starting from the $(i+1)$th active core) following normal GDP iterations.

5 Experimental Results

In this section, we show the experimental results to verify the proposed power budgeting method and analyze its performance. All data are collected on a PC with Intel I5 4200U CPU and 4 GB memory. In the experiment, multi/many-core systems with the number of cores ranging from 9 to 100 are used, and each system is tested with different dark silicon ratios. The thermal models of these systems are extracted from HotSpot [36], [38] with default package and chip parameters. For all test cases, we set the ambient temperature as $20°C$, and the temperature constraint as $80°C$ (so the temperature rise constraint is $60°C$).

5.1 Effectiveness and Performance Tests for Steady State Cases

In order to test the effectiveness and performance of the new method, we first test it using a system with small core number (9 cores) and show its steady state behavior step in step. Please note that we show steps of the 9-core system because it is easier for the readers to verify the correctness of GDP using a system with small core number.

Now we demonstrate how GDP decides which cores to be active in a greedy manner for 4 active cores. For step one, GDP looks for the first active core position. This step is pretty easy even for human, as we can readily pick the center core, because its position has the best heat dissipation capability. GDP, unlike human who uses instinct and experience, picks the core with the largest inner product $\langle a_j, T_{th} \rangle$, which is exactly the center core. Then, GDP computes the power budget for such one active core case. We plot the steady state temperature distribution caused by the computed power budget in Fig. 7a. As expected, the only active core at the center has a temperature of $80°C$, which is just the provided thermal constraint value. Next, GDP searches for the second active core position with the first active core position fixed. Although all four cores at the corners can be chosen due to symmetry, the upper left one is picked simply because computer prefers the first. The updated power budget also leads to $80°C$ for the two active cores shown in Fig. 7b, as expected. For the third and fourth active cores, GDP locates their positions to be lower right corner and upper right corner, with power budget resulted temperature distributions shown in Fig. 7b and 7d, respectively.

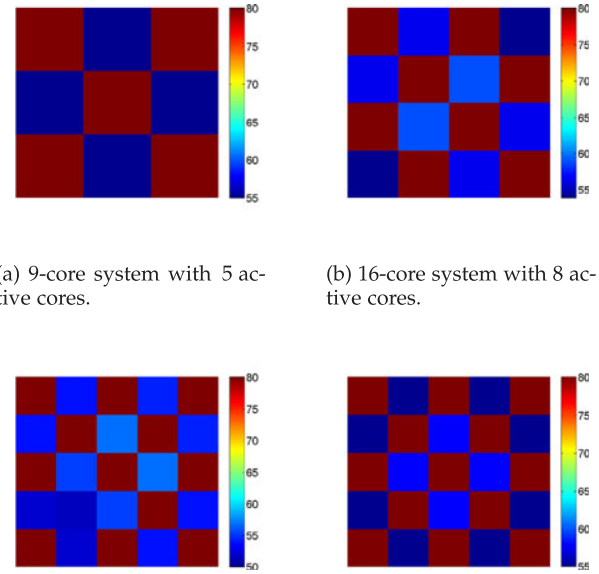
TABLE 1
Steady State Power Budget Comparison

Core #	Active #	GDP (W)	Optimal (W)	TSP (W)
9	2	95.0	96.4	89.9
	4	153.8	156.2	143.8
	7	202.8	203.6	196.5
16	3	130.3	131.8	114.8
	8	227.0	227.9	209.5
	13	269.0	269.8	261.8
25	5	114.2	114.9	100.2
	12	192.3	NA	173.9
	20	232.5	233.1	223.6
36	8	131.1	NA	110.5
	18	202.3		179.6
	28	236.8		224.7
64	12	122.7	NA	97.4
	32	205.5		178.5
	52	237.3		227.1
100	16	124.1	NA	103.1
	52	216.5		183.4
	76	235.4		220.5

"GDP", "Optimal", and "TSP" stand for power budgets given by GDP method, optimal solution with brute force search, and TSP method, respectively.

In addition to the case shown above, we also tested many other systems with different number of cores and active cores. The final power budget resulted temperature distributions of these cases are shown in Fig. 8. For all cases, GDP provides correct greedy based sub-optimal results.

Next, we test the power budget quality provided by GDP. The results are collected in Table 1. For comparison, we compute the optimal power budget by a brutal force search of all active core combinations, and the power budget given by the state-of-the-art method TSP [18]. Please note that the brute force search guarantees to find the global optimal solution but suffers from high computational complexity, and it is unable to deliver the results starting from the 25-core system with 12 active cores case, even after over 12 hours' computing. Although GDP only finds a sub-optimal solution, we find that



(a) 9-core system with 5 active cores.

(b) 16-core system with 8 active cores.

(c) 25-core system with 12 active cores.

(d) 25-core system with 13 active cores.

Fig. 8. Temperature distributions of systems with different core number and active core number.

the GDP computed power budget is very close to the global optimal one.

5.2 Effectiveness and Performance Tests for Transient Cases

GDP is a dynamic based method, meaning it is able to provide power budget adapting to transient running state of the multi/many-core dark silicon system.

GDP mainly focuses on computing the power budget dynamically for the multi/many-core dark silicon system. By using the power budget and active core distribution suggestion provided by GDP, many different thermal management methods can be designed and optimized, which is not the main focus of this work. In order to test the dynamic behavior of GDP for transient cases, the same simple task scheduling and dynamic voltage and frequency scaling strategy is used for all methods: a power matching determines task scheduling and DVFS is performed when the task on the core consumes more power than the provided power budget. Please note that task scheduling and DVFS are used here only on the purpose of showing power budgeting performance. Advanced task scheduling and DVFS methods may further boost performance but is out of scope of this work.

In the experiment, GDP is set to compute power budget dynamically for every 10 seconds ($h = 10$). State-of-the-art power budgeting method TSP [18] and neighbor-aware multi-core dynamic thermal management method NADTM [14] are used for comparison. Two SPEC benchmark applications are running on each active core by round-robin scheduling, with time slice set to be 50 ms [41]. At initiation, the applications are randomly assigned to the active cores. The computing overheads of all methods are considered in the experiments as throughput deduction and management latency. The power consumption is obtained by the power estimator Wattch [42]. There are 18 V/F levels (from 0.32V@140MHz to 1V@2GHz) for DVFS in our experiment, with DVFS action overhead set to be 10 μs by following the settings in [43]. The task migration overhead is set as 10 ms according to [44].

We first verify the effectiveness of TSP and GDP, i.e., we check whether the temperature will be constrained below the threshold if the given power budget is followed. We plot the transient temperature results with GDP and TSP in Fig. 9a and 9b, respectively. Because TSP is a static power budgeting method, we have to activate cores according to the worst case distribution to test if the system temperature is properly controlled with TSP power budget. For GDP, the cores are activated according to the sub-optimal distribution computed by the greedy based algorithm in GDP. From the figure, we can see that both power budgets provided by TSP and GDP are able to constrain the temperatures of all active cores below the user defined thermal threshold (80°C in our test case) with the simple task scheduling and DVFS.[2] In Fig. 9a, core temperature switches between high temperature and low temperature because GDP may switch active core positions dynamically.

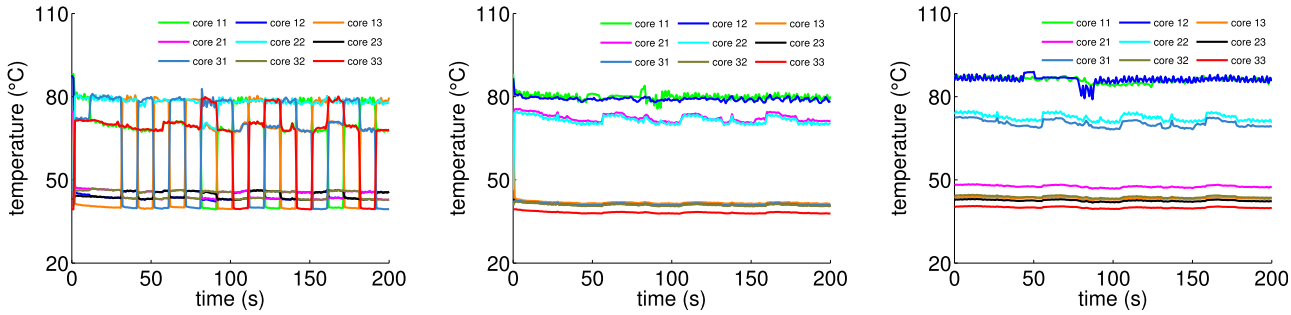
Next, we compare GDP with the neighbor-aware dynamic thermal management method NADTM [14], because both methods take neighbor core's temperature into account. The transient temperature results with NADTM is shown in Fig. 9c. We see that NADTM is not able to constrain the core temperature below the given thermal threshold for the dark silicon system. NADTM fails because it is *not* designed considering dark silicon properties. We provide the detailed discussion as follows.

The basic idea of NADTM is to use a linear model with three inputs (own current temperature, own increment factor, and neighbor increment factor) and three parameters[3] to predict the core's own temperature.

The major problem of using NADTM in dark silicon system is that there are only three inputs and three parameters (specifically, α, β, γ in NADTM paper) in temperature prediction, and only one input and one parameter (γ) among them is used to consider all four neighbor cores' impact. This is far from sufficient to consider the complex dark silicon temperature behaviors. To be specific, for dark silicon system, the neighbor cores could be inactive state or off state, and they impact the neighbor cores quite differently when in different state. If we put all neighbor cores' on-off combinations and different benchmarks into the training process, we end up with a large number of training samples with large diversity. For this overdetermined problem, the least square method will find a solution α, β, γ which works best for all samples. However, because of the large number and large diversity of the samples, the least square solution even has large error for the training samples. In another word, it is impractical to use a model with only three parameters to predict the complex temperature behavior of the dark silicon multi-core systems.

2. In the TSP test case, temperatures of the active cores will be significantly lower than threshold if active core distribution other than the worst case one is used. This is because TSP is a static power budgeting method, which has to provide over pessimistic (i.e., much lower than real) power budget to guarantee thermal safety of the system in all conditions including the worst one.

3. The three parameters are trained using least-square estimation in NADTM algorithm.



(a) Transient temperatures of the system with power budget provided by GDP. The cores are activated according to the sub-optimal distribution given by GDP.

(b) Transient temperatures of the system with power budget provided by TSP. We activate cores according to the worst case distribution given by TSP to verify the effectiveness of TSP.

(c) Transient temperatures of the system with NADTM. Temperature violation occurs since NADTM is not designed for dark silicon system.

Fig. 9. Transient temperatures of the 9-core system with simple task scheduling and DVFS following power budgets provided by TDP and TSP, and with dynamic thermal management method NADTM. Each line represents temperature of one core of the system. The temperature constraint is set as $80^{\circ}\mathrm{C}$ for all methods. The label "core ij" denotes the core located at the ith row and jth column in the 9-core system. The spikes in the transient temperature curves are due to the power variation between two temperature management actions, which can be reduced by using more frequent DVFS actions.

TABLE 2
Dynamic Power Budgeting and System Performance (in MIPS) Results Comparison

Core #	Active #	GDP Budget	GDP MIPS	Reverse GDP Budget	Reverse GDP MIPS	Parallel GDP Budget	Parallel GDP MIPS	Worst TSP Budget	Worst TSP MIPS	Random TSP Budget	Random TSP MIPS
9	2	100.8	180.0	100.2	179.7	NA	NA	89.9	173.3	94.9	176.4
	4	158.3	209.2	157.1	208.7			143.8	202.6	150.4	205.7
	7	206.6	228.7	207.8	229.1			196.5	224.9	200.5	226.4
16	3	136.9	199.4	137.5	199.6	NA	NA	114.8	188.0	127.2	194.5
	8	235.2	238.8	236.4	239.2			209.5	229.7	220.7	233.8
	13	273.2	251.0	274.6	251.4			261.8	247.4	264.9	248.4
25	5	118.2	189.8	118.5	190.0	NA	NA	100.2	179.7	111.4	186.1
	12	194.8	224.2	191.7	223.0			173.9	215.9	184.7	220.3
	20	237.8	239.6	237.6	239.6			223.6	234.8	229.4	236.8
36	8	134.9	198.4	135.6	198.7	NA	NA	110.5	185.6	127.3	194.6
	18	209.3	229.7	205.4	228.2			179.6	218.2	195.8	224.6
	28	242.6	241.2	240.4	240.5			224.7	235.2	232.4	237.8
64	12	128.2	195.0	128.9	195.4	124.6	193.2	97.4	178.0	118.4	189.9
	32	208.9	229.5	207.4	229.0	203.4	227.5	178.5	217.8	197.7	225.3
	52	239.2	240.1	238.5	239.9	238.2	239.8	227.1	236.0	233.9	238.3
100	16	128.4	195.1	129.3	195.6	121.5	191.6	103.1	181.4	114.6	187.9
	52	219.6	233.4	215.4	231.9	213.6	231.2	183.4	219.8	205.7	228.3
	76	240.1	240.4	239.5	240.2	239.3	240.1	220.5	233.7	232.0	237.7

"Budget" is short for power budget with the unit watt. Since GDP provides dynamic budget, the averaged value is provided here. For parallel GDP, the 64-core system and 100-core system are divided into four 16-core systems and four 25-core systems, respectively. " Worst TSP" denotes TSP performance with the worst case active core distribution, and "Random TSP" denotes TSP performance averaged from 10 random active core distributions.

Our experiment verifies this observation. Although working properly for traditional multi-core systems, NADTM is unable to make thermal prediction accurately by considering the impact of neighbor cores for dark silicon systems. As a result, NADTM may lead to temperature violation which puts the chip in danger.

We have shown that the system is thermally safe by following the power budgets given by both GDP and TSP, now we compare the performance of the two methods to see which one gives higher power budget and system performance. The power budget and system performance results are collected in Table 2. Except for showing TSP performance with the worst case active core distribution (denoted as "Worst TSP" in Table 2), we also demonstrate the performance given by TSP averaged from 10 random active core distributions (denoted as "Random TSP" in Table 2). For the transient case, GDP also provides higher power budget and better system performance than TSP, because GDP looks for active core distributions dynamically but TSP only finds the worst case one to ensure absolute thermal safety (for "Worst TSP") or follows a given active core distribution (for "Random TSP"). Moreover, the dynamic power budget provided by GDP is higher than its steady state power budget for the same multi/many-core dark silicon system by comparing data from Tables 1 and 2. This is because GDP automatically provides performance boost by letting the dynamic power budget to be higher than steady state power budget when temperature

is lower than threshold. This property will be shown in details later.

It is also interesting to see how GDP automatically provides "performance boost" when previous temperature is relatively low. We use the 9-core system with 4 cores active as the demonstration case, and provide test input power to its active cores. As shown in Fig. 10a, the test input power contains two non-zero parts. The first part is from 1 s to 25 s. In this part, a constant power, which just equals to the steady state power budget, is applied to the active cores, in order to see how power budget changes when system temperature changes from ambient to steady state value. The second part lasts from 60 s to 90 s. Different from the first part, at the beginning second of the second part (from 60 s to 61 s), we apply an input power which equals to the dynamic power budget provided by GDP (much higher than steady state power budget) to cores. This power is used to verify the accuracy of the dynamic power budget given by GDP. The input power, the resulted temperature, and the power budget provided dynamically by GDP for each future second ($h = 1$) of an active core are shown in Fig. 10.[4]

From the figure, we can see that at 0 s, since the system's temperature is at ambient $20°C$, the power budget for the future second is around 51 W for this core, which is much higher than the steady state power budget around 39 W. It means system's current temperature has huge impact on power budget, and a performance boost with *maximum* power consumption of 51 W can be executed for the future second without violating thermal constraint. Because the steady state power budget is constantly applied from 1 s to 25 s, temperature gradually rises to temperature constraint $80°C$. At the same time, power budget given by GDP decreases from 51 W to the steady state power budget 39 W, indicating the steady state power budget given by GDP is accurate.

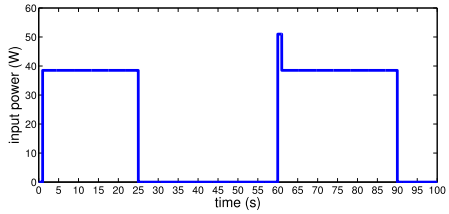
Then, from 25 s to 60 s, the input power is zero, so the system cools down from $80°C$ to ambient temperature. During this time, power budget increases from steady state value 39 W to 51 W for this core, as expected.

Next, input power equals to power budget given by GDP at 60 s (around 51 W) is applied from 60 s to 61 s, to simulate a performance boost consuming full power budget given by GDP. Because of this input power, temperature of the core rises fast and just reaches temperature threshold $80°C$ at 61 s. This matches the expected results illustrated in Fig. 6, meaning the dynamic power budget given by GDP is both accurate and safe.

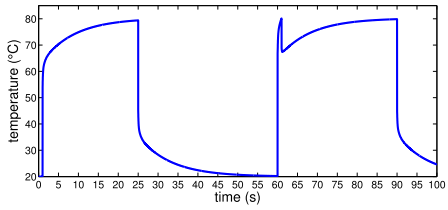
Finally, we apply input power equals to steady state power budget to active cores from 61 s to 90 s. It is interesting to see that temperature of the core drops from $80°C$ to around $68°C$ at first from 61 s to 62 s, then gradually rises back to $80°C$ from 62 s to 90 s. This phenomenon is explained as follows. Although cores of the system have already reached threshold $80°C$ at 61 s, system's package temperature is still much lower than its steady state value. Because of the relatively cool package, input power equals to steady state power budget (39 W) is too low to support the core temperature to stay at $80°C$. Actually, the power budget given by GDP at 61 s is around 48 W, which is able to consider the package effect and keep the core temperature to be $80°C$ at 62 s.

Readers may also notice that for the core temperature's rising process (from 1 s to 25 s) and dropping process (from 25 s to 60 s), power budget is different for the same temperature. To be more specific, for the same core temperature, rising process has a higher power budget than that of the dropping process. This is also due to the fact that package temperature changes much slower than core temperature, and will cool the core down in rising process and heat the core up in dropping process.

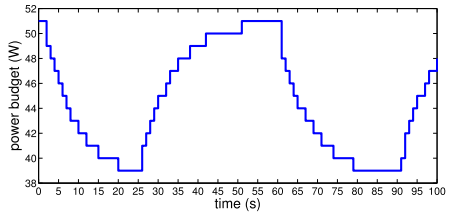
In summary, GDP shows that transient thermal effects from both chip and package have huge impact on power budget. By following the dynamic power budget given by



(a) Input power applied to an active core. It contains two non-zero parts with details explained in manuscript.



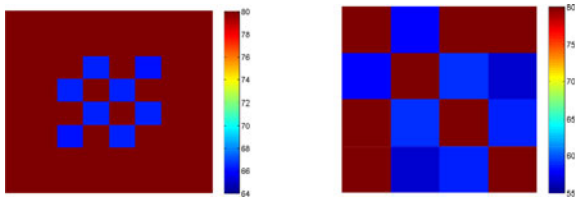
(b) Temperature caused by input power given above.



(c) Power budget calculated dynamically by GDP for each future second. Power budget is largest when the system is at ambient $20\,°C$, it equals to steady state power budget when temperatures of active cores are at thermal threshold $80\,°C$ and the system reaches steady state. Power budget is rounded to integers in this figure for better demonstration.

Fig. 10. Demonstration of GDP's dynamic power budgeting ability which handles performance boost automatically. Please note that input power is artificially given and is not effected by the calculated power budget in this example.

4. Please note that in this example, the input power does *not* relate to the calculated power budget. Input power is only used to generate a time varying temperature to see how power budget changes with current temperature. In addition, power budget is directly related to system's current temperature, but is *not* directly related to input power.



(a) Temperature distribution of the 64-core system with 56 active cores located by reverse GDP.

(b) Temperature distribution of the 64-core system with 32 active cores located by parallel GDP. Only the upper left 16-core subsystem is shown here due to symmetry.

Fig. 11. Examples of reverse GDP and parallel GDP.

TABLE 3
Computing Time Comparison of GDP, Reverse GDP, and Parallel GDP

Core #	Active #	GDP (ms)	Reverse GDP (ms)	Parallel GDP (ms)
64	12	2.7	8.5	0.24
	32	5.5	6.7	0.58
	40	6.2	5.1	0.67
	52	15	3.4	0.97
	60	16	2.1	1.1
100	24	5.3	22	0.44
	52	16	17	0.92
	64	17	14	1.2
	84	19	7.0	1.7
	92	22	4.1	1.8

For parallel GDP, the 64-core system and 100-core system are divided into four 16-core systems and four 25-core systems, respectively.

GDP, system performance can be boosted with guaranteed thermal safety.

5.3 Performance of the Overhead Reduction Techniques

In this part, we test the performance of the two overhead reduction techniques (reverse GDP and parallel GDP) proposed in Section 4.5.

The power budget and system performance results of reverse GDP and parallel GDP are given in Table 2, and the computing time results (for two systems with large core number) are shown in Table 3.

First, let us look at the results of reverse GDP, which is proposed to reduce the GDP computing overhead for systems with high active core ratio. One active core allocation example of reverse GDP for the 64-core system with 56 active cores is given in Fig. 11a. System performance wise, reverse GDP is very close to the standard GDP as seen from Table 2. This is because the active core distribution found by reverse GDP is as optimal as standard GDP. At the same time, reverse GDP brings significant computing time reduction for systems with high active core ratio as shown in Table 3, since less iterations are performed in reverse GDP.

Then, we analyze the results of parallel GDP, which is introduced to reduce the GDP computing time for systems with large core number. Two systems with large core number are tested: the 64-core system and 100-core system are divided into four 16-core systems and four 25-core systems for parallel GDP, respectively. As seen from Table 2, system performance with parallel GDP is a little lower than that of the standard GDP. This is expected since the active core distribution found by parallel GDP is less optimal than the one found by standard GDP (one example of active core distribution found by parallel GDP is given in Fig. 11b, for the 64-core system with 32 active cores). On the other side, the computing time is drastically reduced by parallel GDP as shown in Table 3, meaning the system is safer with shorter power budgeting computing delay.

6 Conclusion

In this article, we have demonstrated the new dynamic power budgeting method called GDP. The new method formulates the power budgeting problem as a constrained power optimization problem. Then, it employs a greedy based algorithm to efficiently find a sub-optimal active core distribution, and computes the corresponding power budget considering current thermal condition of the system. As a dynamic method for dark silicon multi/many-core systems, GDP provides a higher and less pessimistic power budget than the existing methods under the same thermal constraint, which means users can harness more performances from the dark silicon computing system than existing power budgeting methods. Both theoretical studies and experimental results show that GDP outperforms TSP, which is the state-of-the-art power budgeting technology for dark silicon multi-core systems.

Acknowledgments

This research is supported in part by National Natural Science Foundation of China under grant No. 61404024, in part by the Fundamental Research Funds for the Central Universities under grant No. ZYGX2016J043, in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

References

[1] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, May 2012.

[2] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, no. 4, pp. 6–15, Jul. 2011.

[3] N. Goulding-Hotta, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, P.-C. Huang, M. Arora, S. Nath, V. Bhatt, J. Babb, S. Swanson, and M. Taylor, "The GreenDroid mobile application processor: An architecture for silicon's dark future," *IEEE Micro*, vol. 31, no. 2, pp. 86–95, Mar./Apr. 2011.

[4] M. Taylor, "Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse," in *Proc. Des. Autom. Conf.*, 2012, pp. 1131–1136.

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

[6] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *Proc. Des. Autom. Conf.*, May 2013, pp. 1–9.

[7] H. Khdr, S. Pagani, M. Shafique, and J. Henkel, "Thermal constrained resource management for mixed ILP-TLP workloads in dark silicon chips," in *Proc. Des. Autom. Conf.*, 2015, pp. 1–6.

[8] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2003, pp. 2–13.

[9] R. Jayaseelan and T. Mitra, "A hybrid local-global approach for multi-core thermal management," in *Proc. Int. Conf. Comput. Aided Des.*, 2009, pp. 314–320.
[10] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. De Micheli, and R. Gupta, "Processor speed control with thermal constraints," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 56, no. 9, pp. 1994–2007, Sep. 2009.
[11] M. Powell, M. Gomaa, and T. Vijaykumar, "Heat-and-Run: Leveraging SMT and CMP to manage power density through the operating system," in *Proc. Int. Conf. Archit. Support Program. Lang. Operating Syst.*, Oct. 2004, pp. 260–270.
[12] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," in *Proc. Des. Autom. Conf.*, Jun. 2010, pp. 579–584.
[13] T. Chantem, S. Hu, and R. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 10, pp. 1884–1897, Oct. 2011.
[14] G. Liu, M. Fan, and G. Quan, "Neigbor-aware dynamic thermal management for multi-core platform," in *Proc. Eur. Des. Test Conf.*, Mar. 2012, pp. 187–192.
[15] R. Ayoub and T. Šimunić Rosing, "Predict and act: Dynamic thermal management for multi-core processor," in *Proc. Int. Symp. Low Power Electron. Des.*, Aug. 2009, pp. 99–104.
[16] T. Ebi, M. Al Faruque, and J. Henkel, "TAPE: Thermal-aware agent-based power economy for multi/many-core architectures," in *Proc. Int. Conf. Comput. Aided Des.*, 2009, pp. 302–309.
[17] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, Mar./Apr. 2012.
[18] 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. Hardware/Softw. Codes. Syst. Synthesis (CODES+ISSS)*, 2014, Art. no. 10.
[19] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware computer systems: opportunities and challenges," *IEEE Micro*, vol. 23, no. 6, pp. 52–61, Nov./Dec. 2003.
[20] A. K. Coskun, J. L. Ayala, D. Atienza, T. S. Rosing, and Y. Leblebici, "Dynamic thermal management in 3D multicore architectures," in *Proc. Eur. Des. Test Conf.*, Apr. 2009, pp. 1410–1415.
[21] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Proc. Eur. Des. Test Conf.*, Apr. 2007, pp. 1659–1664.
[22] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2006, pp. 78–88.
[23] L. Shang, L.-S. Peh, A. Kumar, and N. K. Jha, "Temperature-aware on-chip networks," *IEEE Micro*, vol. 26, no. 1, pp. 130–139, Jan./Feb. 2006.
[24] O. Khan and S. Kundu, "Hardware/software co-design architecture for thermal management of chip multiprocessors," in *Proc. Eur. Des. Test Conf.*, Apr. 2009, pp. 952–957.
[25] J. Clabes, J. Friedrich, M. Sweet, J. DiLullo, S. Chu, D. P. J. Dawson, P. Muench, L. Powell, M. Floyd, B. Sinharoy, M. Lee, M. Goulet, J. Wagoner, N. Schwartz, S. Runyon, G. Gorman, P. R. R. Kalla, J. McGill, and S. Dodson, "Design and implementation of the POWER5 microprocessor," in *Proc. Des. Autom. Conf.*, 2004, pp. 670–672.
[26] M. Ware, K. Rajamani, M. Floyd, B. Brock, J. C. Rubio, F. Rawson, and J. B. Carter, "Architecting for power management: The IBM POWER7 approach," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.*, 2010, pp. 1–11.
[27] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *Proc. Des. Autom. Conf.*, 2008, pp. 734–739.
[28] H. Wang, J. Ma, S. X.-D. Tan, C. Zhang, H. Tang, K. Huang, and Z. Zhang, "Hierarchical dynamic thermal management method for high-performance many-core microprocessors," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1:1–1:21, Jul. 2016.
[29] A. Pathania, H. Khdr, M. Shafique, T. Mitra, and J. Henkel, "Scalable probabilistic power budgeting for many-cores," in *Proc. Eur. Des. Test Conf.*, Mar. 2017, pp. 864–869.
[30] A. Pathania, H. Khdr, M. Shafique, T. Mitra, and J. Henkel, "QoS-aware stochastic power management for many-cores," in *Proc. Des. Autom. Conf.*, Jun. 2018, Art. no. 69.
[31] A. Pathania, V. Venkatramani, M. Shafique, T. Mitra, and J. Henkel, "Optimal greedy algorithm for many-core scheduling," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 6, pp. 1054–1058, Jun. 2017.
[32] H. Khdr, S. Pagani, É. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel, "Power density-aware resource management for heterogeneous tiled multicores," *IEEE Trans. Comput.*, vol. 66, no. 3, pp. 488–501, Mar. 2017.
[33] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The EDA challenges in the dark silicon era," in *Proc. Des. Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.
[34] M. Shafique, D. Gnad, S. Garg, and J. Henkel, "Variability-aware dark silicon management in on-chip many-core systems," in *Proc. Eur. Des. Test Conf.*, Mar. 2015, pp. 387–392.
[35] H. Wang, M. Zhang, S. X.-D. Tan, C. Zhang, Y. Yuan, K. Huang, and Z. Zhang, "New power budgeting and thermal management scheme for multi-core systems in dark silicon," in *Proc. IEEE Int. Syst.-on-Chip Conf.*, Sep. 2016, pp. 344–349.
[36] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.
[37] H. Wang, S. X.-D. Tan, D. Li, A. Gupta, and Y. Yuan, "Composable thermal modeling and simulation for architecture-level thermal designs of multi-core microprocessors," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 2, pp. 28:1–28:27, Mar. 2013.
[38] 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. Comput.*, vol. 57, no. 9, pp. 1277–1288, Seo. 2008.
[39] V. Hanumaiah, S. Vrudhula, and K. Chatha, "Performance optimal online DVFS and task migration techniques for thermally constrained multi-core processors," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 11, pp. 1677–1690, Nov. 2011.
[40] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "TILTS: A fast architectural-level transient thermal simulation method," *J. Low Power Electron.*, vol. 3, no. 1, pp. 13–21, Apr. 2007.
[41] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 9th ed. Hoboken, NJ, USA: Wiley, 2012.
[42] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2000, pp. 83–94.
[43] Z. Lu, J. Lach, M. R. Stan, and K. Skadron, "Improved thermal management with reliability banking," *IEEE Micro*, vol. 25, no. 6, pp. 40–49, Nov./Dec. 2005.
[44] D. Cuesta, J. Ayala, J. Hidalgo, D. Atienza, A. Acquaviva, and E. Macii, "Adaptive task migration policies for thermal control in MPSoCs," in *Proc. IEEE Annu. Symp. VLSI*, 2010, pp. 110–115.

Hai Wang received the BS degree from Huazhong University of Science and Technology, China, and the MS and PhD degrees from the University of California, Riverside, in 2007, 2008, and 2012, respectively. He is currently an associate professor with the University of Electronic Science and Technology of China. His research interests include electrical/thermal verification and optimization of VLSI circuits and systems. He has served as technical program committee member of several international conferences including DATE, ASP-DAC and ISQED, and also served as reviewer of many journals including the *IEEE Transactions on Computers*, the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, and the *ACM Transactions on Design Automation of Electronic Systems*.

Diya Tang received the bachelor's degree from the University of Electronic Science and Technology of China, in 2017. Currently, she is working toward the master's degree at UESTC. Her current research interests include thermal analysis, power analysis, and thermal management of integrated circuit.

Ming Zhang received the BS and MS degrees from the University of Electronic Science and Technology of China. He is now a software engineer with Cadence. His research interests include place and route of power grid, mainly on the algorithm development and optimization.

Sheldon X.-D. Tan (S'96-M'99-SM'06) received the BS and MS degrees in electrical engineering from Fudan University, Shanghai, China, in 1992 and 1995, respectively and the PhD degree in electrical and computer engineering from the University of Iowa, Iowa City, in 1999. He is a professor with the Department of Electrical Engineering, University of California, Riverside, California. He also is a cooperative faculty member with the Department of Computer Science and Engineering at UCR. He is an associate director of Computer Engineering Program at UC Riverside. He was a visiting professor of Kyoto University as a JSPS fellow in 2017. His research interests include VLSI reliability modeling, optimization and management at circuit and system levels, thermal modeling, optimization and dynamic thermal management for many-core processors, statistical modeling, simulation and optimization of mixed-signal/RF/analog circuits, parallel circuit simulation techniques based on GPU and multicore systems. He received Outstanding Oversea Investigator Award from the National Natural Science Foundation of China (NSFC) in 2008. He received NSF CAREER Award in 2004. He received the Best Paper Award from 2007 IEEE International Conference on Computer Design (ICCD'07), the Best Paper Award from 1999 IEEE/ACM Design Automation Conference. He also receives three Best Paper Award Nomination from IEEE/ACM Design Automation Conferences in 2005, 2009 and 2014 and one Best Paper Award nomination from ASP-DAC in 2015. He now is serving as an editor-in-chief for integration, The *VLSI Journal*. He is also serving as an associate editor for three journals: the *IEEE Transaction on VLSI Systems (TVLSI)*, the *ACM Transaction on Design Automation of Electronic Systems (TODAES)* and the *Microelectronics Reliability*. He is a senior member of the IEEE.

Chi Zhang received the bachelor's degree from Taiyuan University of Science and Technology, and the master's degree from Microelectronics Research Institute of Chinese Academy of Sciences, in 1994 and 2003. He is currently working toward the PhD degree at the University of Electronic Science and technology of China. His main research directions are mixed-signal integrated circuit design, EDA technology, multi-mode biometrics technology.

He Tang (M'09) received the BSEE degree from the University of Electronic Science and Technology of China, Chengdu, China, the MS degree in electrical and computer engineering from the Illinois Institute of Technology, Chicago, and the PhD degree in electrical engineering from the University of California, Riverside, in 2005, 2007, and 2010. From 2010 to 2012, he was with OmniVision Technologies Inc., in Santa Clara, California, as an analog IC designer, where he worked on high-speed I/O interface. Since 2012, he has been an associate professor and subsequently a professor with the University of Electronic Science and Technology of China, Chengdu, China. He has authored or coauthored more than 40 papers. His research interests include data converters and analog/mixed-signal IC designs. His past work includes high-speed high-resolution pipelined ADCs with digital calibration and high-performance ultra-low-power SAR ADCs. He has served on IEEE CAS Analog Signal Processing Technical Committee (ASPTC) since 2013. He is a member of the IEEE.

Yuan Yuan received the BS and MS degrees from the University of Electronic Science and Technology of China, in 1992 and 2005, respectively. He is currently an associate professor with the University of Electronic Science and technology of China. His main research directions are electronic measuring equipment design, computer based measuring technology, embedded system, etc. He has published more than 10 research papers in international conferences and journals.