T-TSP: Transient-Temperature Based Safe Power Budgeting in Multi-/Many-Core Processors

Sobhan Niknam, Anuj Pathania, Andy D. Pimentel
Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands
{s.niknam,a.pathania,a.d.pimentel}@uva.nl

***Abstract*—Power budgeting techniques allow thermally safe operation in multi-/many-core processors while still allowing for efficient exploitation of available thermal headroom. Core-level power budgeting techniques like Thermal Safe Power (TSP) have allowed for more efficient operations than chip-level power budgeting techniques like Thermal Design Power (TDP) since the finer granularity permits operations closer to the threshold temperature without thermal violations.**

State-of-the-art TSP bases its power budgeting calculations on the long-term steady-state temperature of cores while ignoring trends in their short-term transient temperature. In this paper, we propose a new power budgeting technique called T-TSP (Transient-Temperature-based Safe Power) that bases its calculation on the current temperature of the core, a detail ignored by TSP. T-TSP provides a dynamic power budget to a core, which inversely correlates with the core's thermal headroom. Dynamic power budgeting with T-TSP allows cores to reach the threshold temperature faster than TSP and operate safely close to it in perpetuity. Therefore, it provides the same thermal guarantees as TSP but enables even more efficient exploitation of thermal headroom.

We integrate T-TSP with a state-of-the-art thermal interval simulation toolchain. Our detailed evaluations show that benchmarks execute faster by up to 17.94% and 8.37% on average when we do power budgeting with T-TSP instead of the state-of-the-art TSP. Finally, we make T-TSP publicly available in both its integrated and stand-alone forms.

***Index Terms*—Multi-/Many-Core Systems, Dark Silicon, Power Budgeting, Thermal/Power Management.**

I. Introduction

Technology advancements have led to a steady increase in transistor density in chips. Consequently, a chip nowadays integrates more and more processing cores within it. This integration has led to the emergence of now common-place multi-/many-core processors. These processors are primarily suitable for multi-threaded applications whose performance enhances through parallel processing. However, with the breakdown of Dennard scaling, on-chip power density for the processors has been increasing dramatically with the miniaturization of transistor size. This increased power density leads to overheating issues wherein thermal hotspots emerge on the processor even when only some of the available cores operate near their full potential. These hotspots threaten the safety and reliability of the system that deploys a multi-/many-core processor. Therefore, to ensure a thermally safe system operation, the power consumption of the active cores is constrained using

This research received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871259 (ADMORPH project).

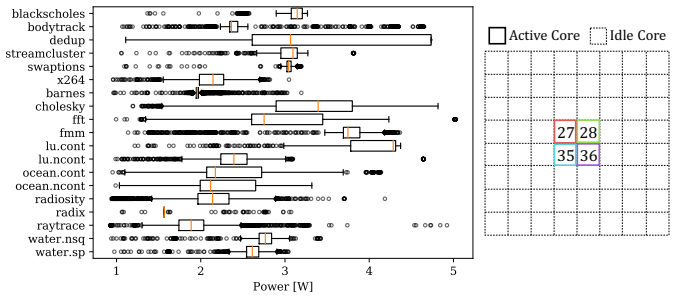

Fig. 1. (a) The variations in the power consumption in some benchmarks from *PARSEC* and *SPLASH-2* suites running on (b) an 8×8 many-core at the highest frequency. The numbers in the squares represent the active cores' id.

power budgets. A power budget for a core is the maximum power that it dissipates with the help of the accompanying cooling solution, i.e., the heat spreader, the heat sink, the cooling fan, etc. This power budget is not determined just by a core's activity but also by the activities of other cores.

Nevertheless, to prevent any thermal violation, processors also come with Dynamic Thermal Management (DTM). DTM is triggered when the processor (or a part of it) heats up above a threshold temperature. DTM controls the temperature by various means, such as powering down the cores, gating their clocks, reducing their supply voltage and frequency, boosting the fan speed, etc. Although DTM ensures the thermally safe operation of a processor, it also leads to substantial degradation in the performance of the overlying system. Therefore, for systems deployed in thermally constrained environments, the primary objective is to maximize performance (and avoid triggering DTM) through techniques like power budgeting.

Thermal Design Power (TDP) [1] is the most commonly used power budget in practice. TDP is a single and constant chip-level power budget that the manufacturer derives at the time of design. Therefore, TDP is unaware of the number and spatial alignment of active cores. Consequently, TDP is often either too pessimistic (wherein significant thermal headroom wastes) or is too optimistic (wherein frequent triggering of DTM occurs).

The authors of [2] present an efficient alternative for TDP called Thermal Safe Power (TSP). TSP considers the spatiality of active cores to provide a per-core budget that is often much higher than TDP while still providing guarantees for thermally safe operation. TSP, by design, derives the per-core power budget for the steady-state. It provides the power budget for a core

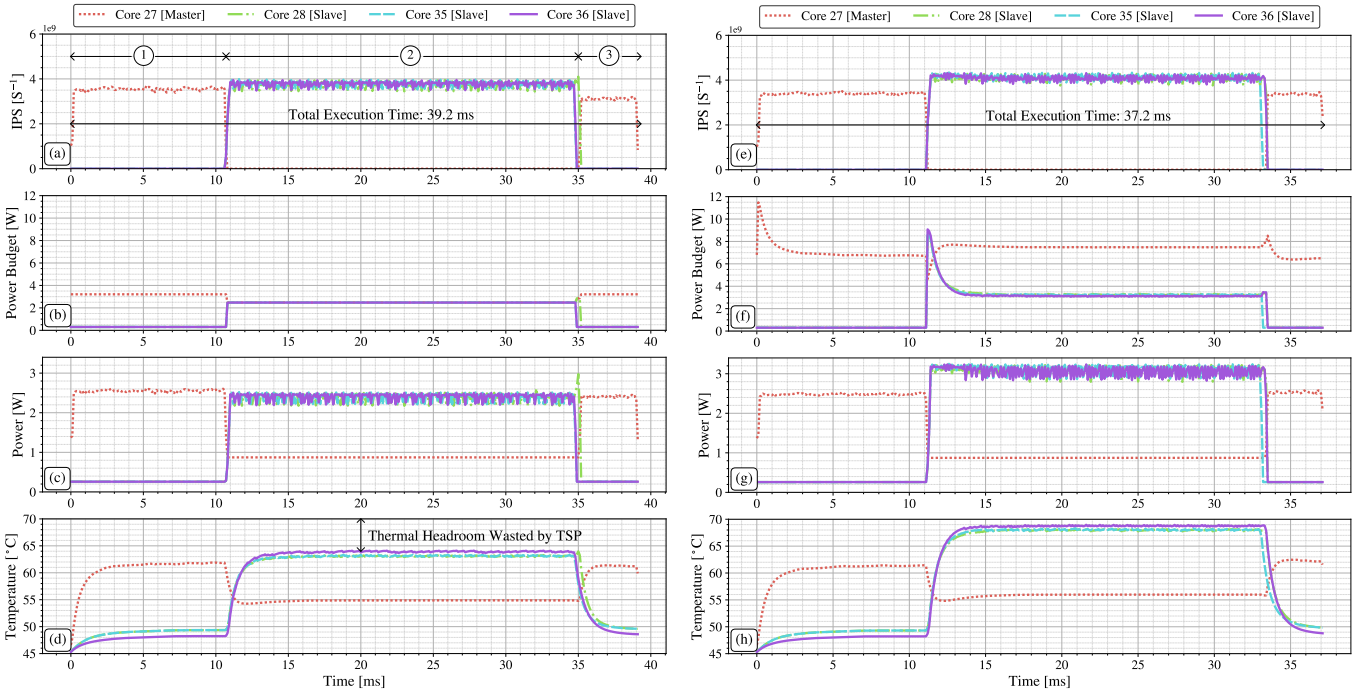

Fig. 2. This figure demonstrates the Instructions per Second (IPS), power budget, actual power consumption, and temperature of the active cores (from our motivational example) running the four-threaded *blackscholes* in (a)-(d) and (e)-(h) wherein the power budgeting is being done over the course of time using TSP and T-TSP, respectively.

that ensures a core's steady-state temperature – the long-term equilibrium temperature a core attains with constant power consumption – does not exceed the temperature threshold. Therefore, by operating at a much fine-grained granularity, core-level TSP offers more opportunities to safely improve system performance – using techniques such as Dynamic Voltage and Frequency Scaling (DVFS) – than chip-level TDP.

However, the power consumption of cores is rarely steady (constant) in practice. A complex application projects a high entropy in the power consumption of its underlying core(s). Figure 1 illustrates this entropy in four-threaded instances of different benchmarks executing on four central cores of an 8×8 many-core at a fixed (highest) voltage and frequency. Consequently, heat-flows occurring between the cores of a processor are almost always in a state of flux. Therefore, cores rarely attain their steady-state temperatures. TSP, therefore, by operating solely on the premise of steady-state, still leaves significant thermal headroom unexploited. The following example illustrates the shortcomings of such traditional *transient temperature-agnostic* use of TSP.

Motivational Example: Figure 1(b) shows the mapping of a four-threaded *blackscholes* benchmark application onto the four central cores of an 8×8 many-core.[1] All other cores in the many-core are idle. The ambient temperature is 45 °C. The temperature threshold at which DTM gets triggered, denoted as T_{DTM}, is 70 °C. The granularity for power budgeting (and DVFS) is 0.1 ms.

Figures 2(a)-(d) show the execution of *blackscholes* wherein TSP does the power budgeting. Figure 2(a) shows the threads' Instructions per Second (IPS) corresponding to the execution. The figure shows that the master-thread execution of *blackscholes* has three main phases. In Phase ①, only the master thread runs. Phase ② starts when the master thread spawns the slave threads. In this phase, the slave threads process the data (prepared by the master thread in Phase ①) while the master thread is near idle. After the slave threads have finished execution and left the system, the master thread resumes completing the task execution in Phase ③.

Figure 2(b) shows the power budget assigned to master and slave threads by TSP for the execution. In both Phases ① and ③, only the master thread is active. Therefore, TSP assigns the power budget of 3.21 W to the core with the master thread, while it assigns an idle core budget of 0.3 W to all other cores. However, in Phase ②, all threads (master and slaves) execute simultaneously on the center cores. Therefore, TSP assigns all active cores a lower uniform power budget of 2.473 W. Since TSP operates at the spatiality of the active cores (which remains unchanged during a phase), the per-core power budgets assigned by TSP are constant over a phase.

Figure 2(c) shows the actual power consumption of the active cores during the execution. A run-time control algorithm with the help of DVFS ensures the execution of a thread close to the assigned power budget. However, the actual power consumption of the cores is inherently unstable due to the ever-changing mix of instructions they are executing from *blackscholes*. Furthermore, due to the discrete nature of DVFS, it is not always possible to attain a target power budget for a core exactly. Consequently, the control algorithm oscillates between two frequencies when the core operates close to the

[1]Refer to Section IV-A for the full details of our experimental setup.

power budget. These induced frequency oscillations also add to the fluctuations in the observed power consumption.

These fluctuations prevent the temperatures of the cores from ever reaching the temperature threshold as expected by TSP. Figure 2(d) shows the core temperatures corresponding to the execution. The figure shows that the temperatures of the cores (even during the compute-intensive Phase ②) never come close to the temperature threshold temperature of 70 °C. The temperature of the cores fluctuates at most close to 64 °C. Therefore, TSP wastes at least a thermal headroom of 6 °C. This wastage highlights the shortcomings of TSP.

We introduce a new power budgeting technique called T-TSP (Transient-Temperature Based Safe Power Budgeting) in this work to address this shortcoming. Most importantly, T-TSP works on the premise of transient temperature in contrast to steady-state temperature-based TSP. The fundamental idea behind T-TSP is to compute a thermally safe power budget for a core at fixed periods (every 0.1 ms in this example) in consideration of its current thermal headroom.

Figures 2(e)-(h) show the execution of *blackscholes* done under T-TSP. Figure 2(f) shows the power budgets assigned under T-TSP for the corresponding execution. In contrast to the static power budgets assigned under TSP (Figure 2(b)), power budgets assigned under T-TSP are dynamic even within a phase. The power budgets assigned by T-TSP have an inverse relation with the current core temperatures. Initially, when a core is idle, its transient temperature is low, and therefore the available headroom is larger. Consequently, T-TSP assigns the core a higher budget when it is just starting with the execution. The assigned power budget gets reduced over time as the core's transient temperature increases and the corresponding thermal headroom reduces.

Figure 2(g) shows the real power consumption of execution under T-TSP. Compared to the power consumption of the execution under TSP (Figure 2(c)), the master thread seems to have a similar power consumption profile under T-TSP as it had under TSP. The memory-intensive nature of the master thread explains this similarity. Its nature prevents it from consuming even the power budget assigned by TSP, let alone the much higher power budget under T-TSP. Nevertheless, the compute-intensive slave threads have no problems consuming the higher T-TSP assigned budgets to boost performance.

Figure 2(h) shows that the slave threads under T-TSP can exploit the entire thermal headroom. Therefore, they operate very near to the temperature threshold of 70 °C. This exploitation leads to faster execution of Phase ②. In conclusion, the execution time for *blackscholes* under T-TSP at 37.2 ms (Figure 2(e)) is 5.1% lower than under TSP (Figure 2(a)) at 39.2 ms.

An interesting observation from Figure 2(f) is that even when the temperatures of the cores running the slave threads are very close to the temperature threshold, their power budget under T-TSP (3.1 W) remains much higher than their power budget under TSP (2.473 W). We can attribute this observation to the fact that the variable core power consumption (even close to the temperature threshold) can result in different transient thermal headrooms over time. This variability prevents the budget assigned to it by T-TSP from dropping to the level of the corresponding budget assigned to it by TSP.

Our Novel Contributions: Based on the above discussion, this paper makes the following novel contributions.

- We propose a new power budgeting technique, called T-TSP, which computes safe power constraint values as a function of *the spatial alignment* of active cores and their *current temperature*. The use of T-TSP leads to better exploitation of the processor's thermals and therefore improved performance in a thermally constrained environment. Furthermore, T-TSP guarantees a thermally safe operation wherein DTM is never triggered.
- We implement and evaluate the proposed technique in the state-of-the-art interval thermal simulation toolchain *HotSniper* [3] over a set of benchmarks from both *PARSEC* [4] and *SPLASH-2* [5] benchmark suites. Our detailed simulations show that execution under power budgeting done with T-TSP results in a significant performance boost over the same execution with power budgeting done with TSP for nearly all tested benchmarks.

Open-Source Contributions: The source code for T-TSP as 1) a stand-alone tool and 2) a *HotSniper* plugin is available for download at https://github.com/sobhanniknam/ttsp.

II. Background

A. System and Application Model

In this work, we consider a many-core processor with n homogeneous cores. All cores in the processor have the same micro-architecture and share the same memory address space. The processor has a per-core DVFS capability which allows it to change the voltage and frequency of each core independently. The cores are thermally constrained, and their temperature should be kept below a given temperature threshold, i.e., T_{DTM}, to avoid triggering DTM. In this work, we consider multi-threaded applications as the system workload. The applications map on multiple cores under a one-thread-per-core model. Multiple applications can execute on the processor in parallel through multi-programming.

B. Thermal Model

In this work, we use the well-established RC thermal model from [6] that has a basis in the well-known duality between the thermal behavior and electrical circuits. In this model, we build an RC thermal network with N thermal nodes for a many-core system. The first n nodes in the model correspond to the cores, while the remaining $N - n$ thermal nodes correspond to the cooling system. Thermal conductances interconnect the thermal nodes. Each thermal node is also associated with a thermal capacitance (except for the thermal node corresponding to the ambient temperature) to model transient temperature. Ambient temperature, denoted as T_{amb}, is considered to be a constant. In this network, the power consumption of the active cores acts as a heat source. Note that any thermal modeling tool, e.g., *HotSpot* [6] and *MatEx* [7], can be used to derive such an RC thermal network for a chip, if detailed information about the

chip's floorplan, technological and cooling system parameters, etc., is known. As this information might be proprietary to the chip manufacturers, one can obtain such an RC thermal model experimentally based on the temperature readings from the thermal sensors on the chip using a methodology similar to those presented in [8], [9].

With the above considerations, we can compute the temperature of each thermal node (a function of its power consumption, the temperature of its neighboring thermal nodes, and the ambient temperature) through a set of N first-order differential equations given by

$$\mathbf{AT}' + \mathbf{BT} = \mathbf{P} + T_{\text{amb}}\mathbf{G}, \tag{1}$$

where $\mathbf{A} = [a_{i,j}]_{N\times N}$ contains the thermal capacitances, $\mathbf{B} = [b_{i,j}]_{N\times N}$ contains the thermal conductivity between neighboring nodes, $\mathbf{T} = [T_i(t)]_{N\times 1}$ contains the temperature on every node at time t, $\mathbf{T}' = [T_i'(t)]_{N\times 1}$ contains the first-order derivative of the temperature on every node concerning time, $\mathbf{P} = [p_i]_{N\times 1}$ contains the power consumption of every node, and $\mathbf{G} = [g_i]_{N\times 1}$ contains the thermal conductivity between every node and the ambient temperature. By defining matrix $\mathbf{C} = -\mathbf{A}^{-1} \times \mathbf{B}$, we can rewrite Equation (1) in a standard form given below.

$$\mathbf{T}' = \mathbf{CT} + \mathbf{A}^{-1}\mathbf{P} + T_{\text{amb}}\mathbf{A}^{-1}\mathbf{G}. \tag{2}$$

Furthermore, in a steady-state, when the temperatures of cores are stable, Equation (1) becomes

$$\mathbf{T}_{\text{steady}} = \mathbf{B}^{-1}\mathbf{P} + T_{\text{amb}}\mathbf{B}^{-1}\mathbf{G} \tag{3}$$

where $\mathbf{T}_{\text{steady}} = [T_{\text{steady}_i}]_{N\times 1}$ contains the steady-state temperature of nodes and $\mathbf{B}^{-1} = [\tilde{b}_{i,j}]_{N\times N}$ is the inverse of matrix $\mathbf{B}$. Note that $\tilde{b}_{i,j} \cdot p_j$, from $\mathbf{B}^{-1}\mathbf{P}$, represents the amount of heat contributed by node j into the steady-state temperature of node i, i.e., T_{steady_i}.

III. T-TSP

The T-TSP's primary objective is to derive the accurate power budget values of the active cores using their transient temperature. T-TSP, therefore, more efficiently exploits the available thermal headroom in the system to improve system performance. The main step towards this objective is to solve Equation (2) to attain the relation between the transient temperature of the cores and their power consumption. As initial conditions are needed to solve any differential equation, we define the column matrix $\mathbf{T}_{\text{init}} = [T_{\text{init}_i}]_{N\times 1}$ as the initial temperature of nodes at time $t = 0s$. Assume $\mathbf{P}$ is the new power vector. After any change in power consumption of one or more cores that happens at time $t = 0$, an analytical solution of Equation (2) (by using the well-studied matrix exponentials [7]) is

$$\mathbf{T}(t) = \mathbf{T}_{\text{steady}} + e^{\mathbf{C}t}(\mathbf{T}_{\text{init}} - \mathbf{T}_{\text{steady}}), \tag{4}$$

where we derive the temperature of the cores at time t as a function of $\mathbf{T}_{\text{init}}$, $\mathbf{T}_{\text{steady}}$ (that depends on the new vector $\mathbf{P}$ according to Equation (3)), and $\mathbf{C}$. In Equation (4), $e^{\mathbf{C}t} = [e^{\mathbf{C}t}{}_{i,j}]_{N\times N}$ is defined as the matrix exponential and can be analytically computed as $e^{\mathbf{C}t} = \mathbf{VDV}^{-1}$, where $\mathbf{V} = [v_{i,j}]_{N\times N}$ represents a matrix containing the eigenvectors of matrix $\mathbf{C}$, $\mathbf{V}^{-1} = [\tilde{v}_{i,j}]_{N\times N}$ is the inverse of matrix $\mathbf{V}$, and

$$\mathbf{D} = \begin{pmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_N t} \end{pmatrix}$$

is a diagonal matrix where $\lambda_1, \lambda_2, \cdots, \lambda_N$ are the eigenvalues of matrix $\mathbf{C}$. Thus, every element $e^{\mathbf{C}t}{}_{i,j}$ of matrix $e^{\mathbf{C}t}$ can be computed as $\sum_{k=1}^{N} v_{i,k} \cdot e^{\lambda_k t} \cdot \tilde{v}_{k,j}$. Since matrix $\mathbf{C}$ is hardware dependent, it is only needed to compute matrix $e^{\mathbf{C}t}$ once for every chip.

Having Equation (4), we are now ready to compute the power budget value of the cores for their current temperature. Note that we compute the power budget value of cores at a fixed period, called power budgeting epoch, with the length of τ. τ is, therefore, the time steps between every power budget computation. In this way, we can assume that the initial temperature of cores sets to the transient temperature of cores at the beginning of each epoch at time t, i.e., $\mathbf{T}_{\text{init}} = \mathbf{T}(t)$. Furthermore, to exploit the available thermal headroom, the transient temperature of cores should reach the temperature threshold by the end of the epoch, i.e., $\mathbf{T}(t+\tau) = [T_{\text{DTM}}]_{N\times 1}$. Therefore, we can compute $\mathbf{T}_{\text{steady}}$ (the only variable in Equation (4)) as

$$\mathbf{T}_{\text{steady}} = (\mathbf{I} - e^{\mathbf{C}\tau})^{-1}(\mathbf{T}(t+\tau) - e^{\mathbf{C}\tau}\mathbf{T}(t)) \tag{5}$$

where $\mathbf{I}$ is the identity matrix of size N. Note that we can rewrite the expression $(\mathbf{I} - e^{\mathbf{C}\tau})^{-1}$ as

$$\begin{aligned}\mathbf{Q} = (\mathbf{I} - e^{\mathbf{C}\tau})^{-1} &= (\mathbf{I} - \mathbf{VDV}^{-1})^{-1} = (\mathbf{VIV}^{-1} - \mathbf{VDV}^{-1})^{-1} \\ &= (\mathbf{V}(\mathbf{I} - \mathbf{D})\mathbf{V}^{-1})^{-1} = \mathbf{V}(\mathbf{I} - \mathbf{D})^{-1}\mathbf{V}^{-1}\end{aligned} \tag{6}$$

where

$$(\mathbf{I} - \mathbf{D})^{-1} = \begin{pmatrix} \frac{1}{1-e^{\lambda_1\tau}} & 0 & \cdots & 0 \\ 0 & \frac{1}{1-e^{\lambda_2\tau}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{1-e^{\lambda_N\tau}} \end{pmatrix}$$

as both $\mathbf{I}$ and $\mathbf{D}$ are diagonal matrices. Similarly, we can rewrite the expression $(\mathbf{I} - e^{\mathbf{C}\tau})^{-1}e^{\mathbf{C}\tau}$ as

$$\begin{aligned}\mathbf{R} = (\mathbf{I} - e^{\mathbf{C}\tau})^{-1}e^{\mathbf{C}\tau} &= \mathbf{V}(\mathbf{I} - \mathbf{D})^{-1}\mathbf{V}^{-1}\mathbf{VDV}^{-1} \\ &= \mathbf{V}(\mathbf{I} - \mathbf{D})^{-1}\mathbf{DV}^{-1}\end{aligned} \tag{7}$$

where

$$(\mathbf{I} - \mathbf{D})^{-1}\mathbf{D} = \begin{pmatrix} \frac{e^{\lambda_1\tau}}{1-e^{\lambda_1\tau}} & 0 & \cdots & 0 \\ 0 & \frac{e^{\lambda_2\tau}}{1-e^{\lambda_2\tau}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{e^{\lambda_N\tau}}{1-e^{\lambda_N\tau}} \end{pmatrix}.$$

Using the above defined auxiliary matrices, we can rewrite Equation (5) as

Algorithm 1: Power budgeting of cores under T-TSP

Input: Floorplan, T_{amb}, T_{DTM}, $\mathbf{T}_{\text{init}}$, p_{idle}, τ, and AC
Output: Matrix $\mathbf{P}$

/* Design-time phase */
1 **forall** $i = 1, 2, \cdots, N$ **do**
2 **forall** $j = 1, 2, \cdots, N$ **do**
3 $q_{i,j} = \sum_{k=1}^{N} v_{i,k} \times \frac{1}{1-e^{\lambda_k \tau}} \times \tilde{v}_{k,j}$
4 $r_{i,j} = \sum_{k=1}^{N} v_{i,k} \times \frac{e^{\lambda_k \tau}}{1-e^{\lambda_k \tau}} \times \tilde{v}_{k,j}$
/* Run-time phase */
5 $\mathbf{P} = [p_i = 0]_{n \times 1}$
6 **forall** $core_i \in AC$ **do**
7 $T_{\text{steady}_i} = T_{\text{DTM}} \times \sum_{j \in AC} q_{i,j} - \sum_{j \in AC} r_{i,j} \times T_{\text{init}_j}$
8 **forall** $core_i \notin AC$ **do**
9 $p_i = p_{idle}$
10 **forall** $core_i \in AC$ **do**
11 $\sum_{j \in AC} \tilde{b}_{i,j} \times p_j = T_{\text{steady}_i} - T_{\text{amb}} \times \sum_{j=1}^{N} \tilde{b}_{i,j} \times g_j - \sum_{j \notin AC} \tilde{b}_{i,j} \times p_j$
12 **return** Matrix $\mathbf{P}$

$$\mathbf{T}_{\text{steady}} = \mathbf{Q}\mathbf{T}(t+\tau) - \mathbf{R}\mathbf{T}(t). \tag{8}$$

Now, by substituting the computed $\mathbf{T}_{\text{steady}}$ (by using Equation (8)) into Equation (3) (wherein the matrix $\mathbf{P}$ is the only variable), we can compute the new matrix $\mathbf{P}$ using the Gaussian elimination algorithm. $\mathbf{P}$ now contains the power budget value of cores for the next epoch.

Algorithm 1 presents the pseudo-code for computing the power budget values under T-TSP. This algorithm takes as inputs floorplan including hardware-dependent matrices (computed once for every chip at design time), an ambient temperature T_{amb}, a thermal threshold T_{DTM}, an initial temperature matrix $\mathbf{T}_{\text{init}}$, the power consumption of an idle core p_{idle}, an epoch length of τ, and a set AC of active cores. The algorithm returns as output a matrix $\mathbf{P}$ containing the power budget values of the cores on the chip. At first, the algorithm computes the auxiliary matrices of $\mathbf{Q}$ and $\mathbf{R}$ (in lines 1-4) at design time using Equations (6) and (7), respectively. Then, at run time, the algorithm first initializes matrix $\mathbf{P}$ to zero in line 5. Then, it computes the new value of T_{steady_i} only for active cores, having the major heat contributions on each others' steady-state temperature, in lines 6-7. In lines 8-9, the algorithm sets the power budget of idle cores to p_{idle}, i.e., the maximum power consumed by a core's associated LLC bank even though the core is idle. The algorithm then computes the power budget of p_i of each active core i using Equation (3) in lines 10-11. Note that in this equation, the heat contribution of each idle core j with the assigned budget of p_{idle} is also considered on the steady-state of an active core i using $\tilde{b}_{i,j} \cdot p_j$.

A. Run-Time Overhead

So far, we assume that the power budget values are computed and applied instantly at the beginning of each epoch. This assumption is, however, not feasible in practice. The execution of Algorithm 1, to compute new power budget values, has a run-time overhead (t_{ov}). This overhead implies that the newly computed power budget values in the current epoch, beginning at time t, are available (and effectively applied) at the time $t + t_{\text{ov}}$. Meanwhile (during $[t, t + t_{\text{ov}}]$), the active cores can still consume power according to their assigned power budget in the previous epoch. However, the cores may have different temperatures at time $t+t_{\text{ov}}$ compared to time t, with which Algorithm 1 originally computes the new power budget values. Thus, the newly computed power budget values are no longer safe to apply at time $t+t_{\text{ov}}$, as they may result in thermal violations.

One possible solution to avoid any thermal violation is: 1) at first, predict how the initial temperature of cores at the beginning of each epoch at time t evolves until time $t + t_{\text{ov}}$, during which cores are still consuming the old power budget values. Equation (4) predicts the transient temperature of cores at time $t + t_{\text{ov}}$ as follows:

$$\mathbf{T}(t + t_{\text{ov}}) = \mathbf{T}_{\text{steady}} + e^{\mathbf{C}(t+t_{\text{ov}})}(\mathbf{T}_{\text{init}}(t) - \mathbf{T}_{\text{steady}})$$

where $\mathbf{T}_{\text{init}}(t)$ contains the initial temperatures of cores at time t and $\mathbf{T}_{\text{steady}}$ contains the steady-state temperature of the cores according to the power budget values of cores assigned in the previous epoch. 2) Then, we can use the predicted temperature of cores at time $t + t_{\text{ov}}$ in Algorithm 1 instead of their (measured/computed) initial temperature at time t to compute the new safe power budget values of cores. Each epoch is theoretically considered in period $[t, t + \tau]$, whereas Algorithm 1 assigns the computed power budgets to cores in period $[t + t_{\text{ov}}, t + t_{\text{ov}} + \tau]$.

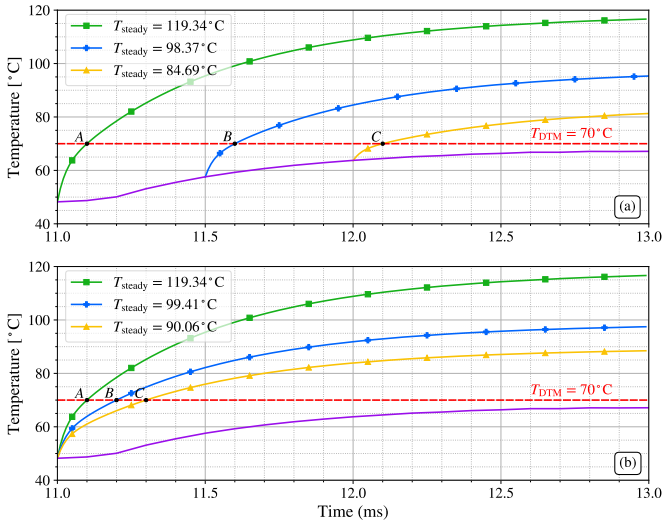

Fig. 3. An example showing how the power budget of a core is computed under T-TSP with respect to (a) its transient temperature and (b) the epoch length, when the overhead is assumed to be negligible.

B. Working Example

Figures 3 and 4 present an example to provide insights into the periodic computation of power budget by Algorithm 1. For the sake of simplicity, we assume the run-time overhead to be

negligible in Figure 3. This figure zooms in the sharp temperature rise of core 36, shown in Figure 2(h) with a solid (violet) curve, from 48.22 °C to 67.20 °C within the time interval of [11, 13] ms. In Figure 3(a), we demonstrate the effect of increasing the transient temperature of the core on its computed steady-state temperature and the corresponding power budget value. At time t =11 ms, the transient temperature of the core is 48.22 °C. Therefore, the available thermal headroom is about 21.78 °C when considering a thermal threshold of 70 °C. By targeting the thermal headroom to be exploited by the end of the upcoming epoch $[t, t+\tau]$, T-TSP computes the steady-state temperature and the corresponding power budget of the core. Assuming the epoch length of τ =100 μs, T-TSP computes the steady-state temperature of 119.34 °C. Using this temperature, the computed transient temperature of the core (shown as a solid (green) curve with square markers in Figure 3(a)) according to Equation (4) reaches the temperature threshold, at point *A*, by the end of the upcoming epoch at time $t + \tau$ =11.1 ms. The transient temperature of the core at the beginning of the other two epochs of [11.5,11.6] ms and [12,12.1] ms are 57.54 °C and 63.64 °C, respectively. This increase in transient temperature of the core decreases the available thermal headroom on the core. By targeting the thermal headroom to be exploited by the end of the epochs at 11.6 ms and 12.1 ms, T-TSP computes the steady-state temperature of 98.37 °C and 84.69 °C, respectively. As a result, the transient temperature of the core (according to Equation (4)) reaches the temperature threshold at points *B* and *C* for the second and third epochs, respectively. The corresponding power budget value of the core for the computed steady-state temperatures of the three (non-consecutive) epochs are 8.57 W, 5.85 W, and 4.08 W, respectively.

In Figure 3(b), we demonstrate the effect of increasing the length of the power budgeting epoch on the computed steady-state temperature and the power budget value of the core. As explained above, T-TSP computes the steady-state temperature of 119.34 °C at time t =11 ms when assuming the epoch length of τ =100 μs. Following the same approach, assuming different epoch lengths of 200 μ and 300 μ results in different steady-state temperatures of 99.41 °C and 90.06 °C for which the computed transient temperature of the core (according to Equation (4)) reaches the temperature threshold, at the points *B* and *C*, respectively. The corresponding power budget values of the computed steady-state temperatures in the order of increasing epoch length are 8.57 W, 6.11 W, and 4.95 W, respectively. This observation implies that the longer is the epoch length, the lower is the computed steady-state temperature and power budget value of the core.

Let us now assume that the run-time overhead is non-negligible, i.e., t_{ov} = 65 μs according to our experiments in Section IV-B. Figure 4 further zooms in the temperature rise of core 36, shown in Figure 2(h) within the time interval of [11, 11.5] ms. The transient temperature of the core at the beginning of the first epoch [11, 11.1] ms is 48.22 °C. The dotted (green) line with a square marker in this figure represents the temperature rise of the core during the period

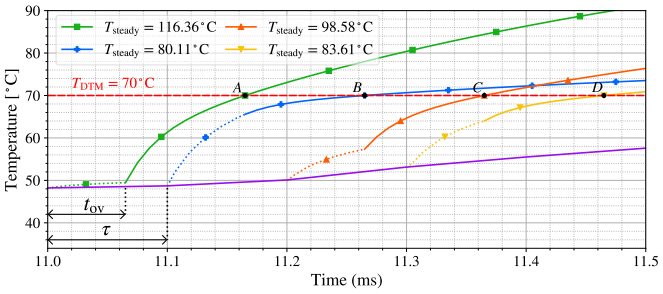

Fig. 4. An example showing how the power budget value of a core under T-TSP is computed when considering the run-time overhead.

of $[t, t + t_{ov}] = [11, 11.065]$ ms when the core consumes the power according to its assigned budget in the previous epoch, i.e., 0.3 W as the core was idle. In this way, the transient temperature of the core (according to Equation (4)) is expected to reach at most to 49.46 °C at time $t + t_{ov}$=11.065 ms, thereby leaving the thermal headroom of 20.54 °C. T-TSP then computes the steady-state temperature of 116.36 °C. Using this temperature, the computed transient temperature of the core (shown as a solid (green) curve with square markers in Figure 4) reaches the temperature threshold (70 °C) at point A at time $t + t_{ov} + \tau$ = 11.165 ms. The corresponding power budget value of the core for the next epoch is then 8.28 W. However, this high assigned power budget causes a pessimistic prediction of the core's transient temperature of 65.58 °C at the time 11.165 ms. Consequently, it leaves only a tiny thermal headroom of 4.4 °C. The dotted (blue) line (with a plus sign marker) in Figure 4 shows the rise in temperature of the core. The rise shown is from its actual transient temperature at 11.1 ms to the predicted transient temperature at 11.165 ms. For the next epoch, this pessimism results in the pessimistic steady-state temperature and power budget of 80.11 °C and 3.66 W, respectively. Similarly, the computed steady-state temperature and power budget values for the core in the third and fourth (consecutive) epochs are (98.56 °C, 6.01 W) and (83.61 °C, 4.1 W), respectively.

In contrast to T-TSP, TSP always considers the temperature threshold (70 °C) as the steady-state temperature of the core. Consequently, this consideration results in the (constant) power budget of 2.473 W when using Equation (3) and considering the mapping shown in Figure 1(b). This example shows that T-TSP, compared with TSP, can compute a much higher yet safe power budget value for the core.

IV. Experimental Evaluation

In this section, we first describe the experimental setup used for evaluating T-TSP. In the next part, we describe different experiments performed to quantify the benefits of the proposed T-TSP approach compared with the state-of-the-art TSP approach [2].

A. Experimental Setup

This work employs state-of-the-art open-source Electronic Design Automation (EDA) tools for experimentations. We use

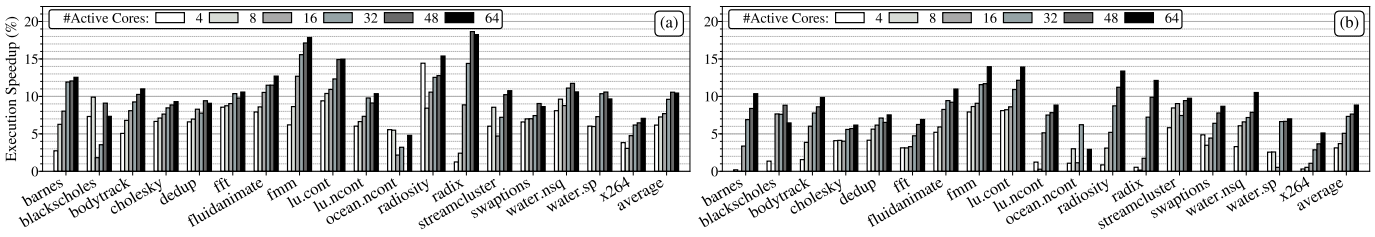

Fig. 5. The effect of increasing the number of active cores on the execution speedup of running tasks when power budgeting under T-TSP (considering τ=100 μs) compared to TSP for (a) T_{DTM}=65 °C and (b) T_{DTM}=70 °C.

the *HotSniper* [3] thermal interval simulation toolchain to simulate the execution of multi-threaded applications. *HotSniper* tightly integrates the *Sniper* [10] many-core interval simulator, *McPAT* [11] Power Modeling framework, and *HotSpot* [6] thermal modeling tool in a unified toolchain.

For the hardware platform, we simulate a many-core system with 64 out-of-order cores located in an 8×8 grid and communicating over a 2D Network-on-Chip (NoC), as shown in Figure 1(b). The NoC uses the conventional XY routing with a latency of 1.5 ns (corresponds to 6 CPU cycles at 4 GHz) per-hop and a link bandwidth of 256 bits per cycle. Cores have *Intel Gainestown* micro-architecture with *x86* Instruction Set Architecture (ISA). Each core has private L1 data and instruction caches of size 16 KB each and an access latency of 3 cycles. We assume an 8 MB shared LLC using S-NUCA policy where each core holds a 128 KB LLC bank with an access latency of 8 cycles. We assume the use of a 14 nm technology node for fabrication of the many-core. Each core, including caches, has an area of 0.79 mm^2. We assume per-core DVFS with a frequency range of 1 GHz up to 4 GHz with steps of 100 MHz and DVFS epoch of 100 μs. The most recent technology node supported by *McPAT* is 22 nm. Therefore, we estimate the power consumption at 22 nm and scale it to 14 nm using scaling factors [12]. We used *HotSpot* and *MatEx* [7] to compute the hardware-dependent matrices needed to create the thermal model of the many-core. We set the ambient temperature to 45 °C and set the value of thermal threshold T_{DTM} to 65 °C or 70 °C in our experiments. We set the power consumption of an idle core p_{idle} to 0.3 W.

For the system's software workload, we use 19 benchmarks taken from both *PARSEC* [4] and *SPLASH2* [5] multi-threaded benchmark suites with *small* inputs as they are large enough to stress the caches. However, they are also still small enough to allow the simulation to finish in a reasonable time.

B. Results and Analysis

In this section, we compare the power budgeting efficacy of our T-TSP to the state-of-the-art approach TSP [2] (described in the motivational example in Section I). The average execution time of the concurrently running multi-threaded tasks on the many-core system act as the comparison metric. To prevent long-running tasks from dominating the average execution time, we compare the geometric means of the execution times. Note that we observe no thermal violation when power budgeting either using TSP or T-TSP.

In the first set of experiments, we assume that the run-time overhead of Algorithm 1 running in parallel with the executed tasks on a single core is negligible. The experimental results under this assumption can provide some insights into the maximum execution speedup possible under our T-TSP vis-a-vis TSP. Figure 5 presents the speedup over average execution time for different tasks with power budgeting under T-TSP (considering τ=100 μs) compared to TSP for a different number of active cores. In these experiments, we run multiple instances of the given benchmark, each of which has a different number of threads such that they utilize all the active cores. The active cores are also selected from the center of the many-core to evaluate the performance of both power budgeting approaches under the worst-case thermal condition. Figures 5(a) and 5(b) show the result for T_{DTM}=65 °C and T_{DTM}=70 °C, respectively. We observe an increase in execution speedup for most benchmarks when the number of active cores increases. We can observe the same trend by comparing Figures 5(a) and 5(b) when reducing thermal threshold. The decrease in the non-uniform per-core power budget (to avoid any thermal violations) with the increase in active cores or thermal threshold decrease explains the trend. As a result, the benefit of power budgeting under our T-TSP compared to TSP can be more significant when the active cores have more constrained power budgets. Figures 5(a) and 5(b) show the execution time of tasks under T-TSP (compared to TSP) can be reduced by up to 18.25% and 10.4% on average when T_{DTM}=65 °C and all cores are active. Similarly, this performance gain when T_{DTM}=70 °C is up to 13.96% and 8.8% on average. In our remaining experiments, we only consider the worst-case thermal scenario wherein all the cores are active. We also now fix T_{DTM} to one value of 65 °C.

We further analyze the performance of our T-TSP under different epoch lengths. The results for this experiment, considering negligible run-time overhead, are demonstrated in Figure 6(a). In this figure, we can see that the benefit of using T-TSP compared to TSP decreases with the increase in the epoch length for most of the benchmarks. This decrease is mainly because the power budgets are computed more conservatively for a longer period. Consequently, the effect of considering the transient temperature gradually cancels out with an increase in the epoch length. Therefore, T-TSP will

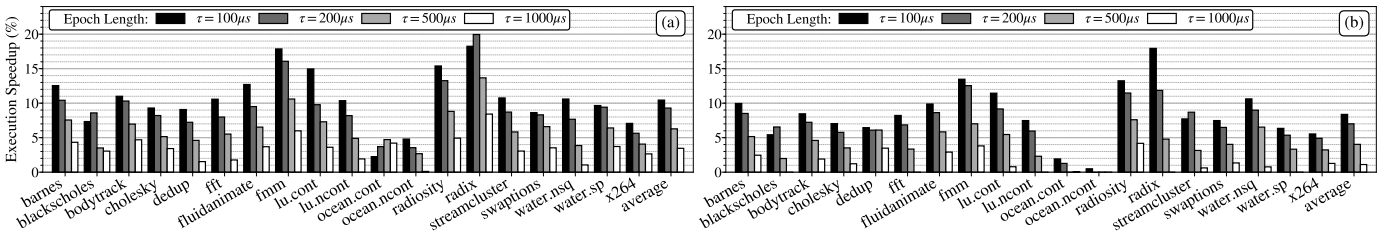

Fig. 6. The effect of increasing the epoch length, τ, on the execution speedup of tasks running on different number of active cores when power budgeting is done under T-TSP compared to TSP with (a) negligible run-time overhead assumption and (b) with measured run-time overhead t_{ov} of 65 μs in consideration.

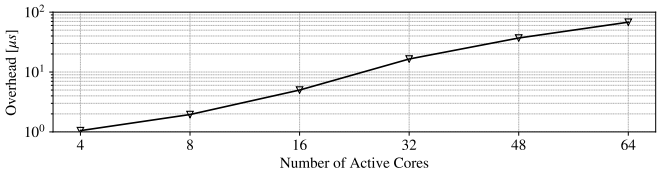

Fig. 7. Average run-time overhead t_{ov} of Algorithm 1 with different number of active cores.

result in the same power budget values as TSP for long enough epochs. Figure 6(a) shows the average execution speedup under T-TSP compared to TSP decreases from 10.4% to 9.3%, 6.2%, and 3.4% for an increase in epoch length τ from 100 μs to 200 μs, 500 μs, and 1 ms, respectively.

Now let us consider the impact of the run-time overhead of Algorithm 1, running in parallel to the executed tasks, on the execution time of the tasks. The run-time overhead of Algorithm 1 plays an important role in the computed power budget of active cores under T-TSP. As shown in Section III-B, T-TSP computes the power budgets conservatively when considering the run-time overhead of Algorithm 1 to avoid any thermal violations. Figure 7 shows the run-time overhead of Algorithm 1 for different numbers of active cores in the system. The overhead reported is averaged over 1000 runs. According to this figure, the average overhead increases with the number of active cores from 1 μs for a single active core to 65 μs when all 64 cores are active.

Considering the run-time overhead of 65 μs, Figure 6(b) demonstrates the execution speedup of our T-TSP compared to TSP. When considering epoch length τ=100 μs, the average execution time of tasks under T-TSP can speed up by 17.94% and by 8.37% on average. Comparing Figure 6(a) and Figure 6(b), the execution speed of T-TSP compared to TSP is reduced by on average 2% and by up to 4.4% when considering run-time overhead compared to the negligible overhead scenario. These results show that such run-time overhead has only a minor impact on the performance of our T-TSP. Figure 6(b) also demonstrates the effect of increasing the epoch length on the performance of our T-TSP. Following the similar trend as in Figure 6(a), we see a decrease in the speedup under our T-TSP compared to TSP in Figure 6(b). The average speedup decreases from 8.37% (for $\tau = 100\,\mu$) to 7%, 4%, and 1.11%, with an increase in epoch length τ to 200 μs, 500 μs, and 1 ms, respectively.

V. Related work

Power budgeting is quintessential in thermal management of multi-/many-cores. TDP [1], a chip-level power budget, is an inefficient yet popular technique used extensively to perform power budgeting. The authors in [13] propose an algorithm that greedily distributes the available power budget, under TDP, to all active cores according to their Instructions per Cycle (IPC). However, TDP is now increasingly replaced by more advance and fine-grained power budgeting techniques.

The authors in [2] propose a per-core power budgeting technique called TSP. TSP primarily depends on active cores and their spatial alignment. Apart from online power budgeting under TSP for a particular thread mapping, the authors in [2] also compute the offline power budgets under TSP for the worst-case mappings of active cores. Worst-case mapping gives the lowest power budget for a given number of active cores. TSP is significantly better in exploiting thermal headroom compared to TDP. However, by being agnostic to transient temperature, it is still too pessimistic. Consequently, it also fails to exploit the entire thermal headroom. *The transient-temperature aware technique T-TSP introduced in this work address this shortcoming.*

The authors in [14] propose a run-time refinement to TSP that reallocates the excessive power budget of the idle or memory-intensive threads to more compute-intensive threads at run-time. This run-time approach can also be applied with T-TSP to further boost [15] the system performance by assigning the power budgets more intelligently to active cores when one or multiple threads are idle/memory intensive. The authors further extended the idea to work with thread (task) migrations in [16]. However, T-TSP (similar to TSP) in its default form is application-independent.

The authors in [17] propose a power-temperature stability and safety analysis technique for multi-core processors. The proposed analysis considers the positive feedback between the leakage power and the temperature. It then computes the stable fixed point at which the power-temperature trajectory of a core is eventually converged. The analysis forms the foundation for the computation of the fixed point (i.e., steady-state) temperature and maximum thermally-safe *dynamic* power consumption at run-time based on the current temperature of the cores. Although this technique is more fine-grained than TSP, it has the same shortage as TSP and only considers the steady-state temperature for power budgeting.

The authors in [18] propose a heuristic algorithm, called GDP, to find near-optimal threads mapping that results in high per-core power budgets on multi-/many-core systems using both the spatial alignment of the active cores and their transient temperature. By solving Equation (2) numerically, both the power budgeting and task re-mapping are performed periodically at each mapping epoch. Authors in [18] presume a long mapping epoch (in the order of seconds) to avoid high run-time overhead due to frequent thread re-mapping. However, such a coarse-grained epoch length (considering the results shown in Figure 6) can entirely cancel out the benefit of considering the transient temperature in the power budgeting computations. Moreover, the approach in [18] lacks the consideration of run-time overhead from power budgeting (as described in Section III-A) on the computed power budgets. Without this consideration, the approach in [18] may result in thermal violations when applied at a more fine-grained epoch length. In contrast to [18], in this work, we focus on power budgeting at a fine-grained epoch length (in the order of a few microseconds) for a given mapping while still taking the run-time overhead of power budget computation into account. Moreover, in contrast to the approach in [18], we perform a more extensive and realistic analysis for T-TSP. Authors in [18] only evaluate their approach using *HotSpot* and that also solely with single-threaded benchmarks. We, on the contrary, evaluate T-TSP using detailed interval thermal simulations of multi-threaded benchmarks using *HotSniper* [3].

The authors in [19] propose an empirical model for computing the maximum sustainable per-chip power budget concerning the current temperature and spatial alignment of the active cores on the chip. They use *HotSpot* for deriving the model. To derive the model, the uniform per-core power budget for limited sets of active cores, each having a different initial temperature, is computed at design time. Authors do this by increasing the power budget from a low value as long as the peak chip temperature, simulated by *HotSpot*, remains below the threshold temperature. The authors further extend the model to support non-uniform per-core power budgets in [20]. In contrast to analytical T-TSP, this empirical approach is prone to errors that lead to incorrect power budgeting. Consequently, the errors lead to thermal headroom wastage/thermal violation.

VI. Conclusion

In this paper, we propose T-TSP as a novel power budgeting technique for multi-/many-core systems. The fundamental feature of T-TSP is to incorporate the transient temperature of the cores in power budgeting calculations, a detail ignored by state-of-the-art power budgeting techniques. This feature enables T-TSP to provide a dynamic power budget to a core, which inversely correlates with the core's thermal headroom. With efficient exploitation of such transient thermal headroom, dynamic power budgeting with T-TSP can result in a significant system performance boost in a thermally constrained environment. Executing cores at any power consumption below their power budgets under T-TSP ensures thermally safe operation wherein the temperature of cores remains below the threshold temperature. Our experiments show that benchmarks execute faster by 17.94% and 8.37% on average when power budgeting with T-TSP instead of TSP.

References

[1] T. Mitra. Heterogeneous multi-core architectures. *Information and Media Technologies*, 10(3):383–394, 2015.

[2] 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 *International Conference on Hardware/Software Codesign and System Synthesis*, pages 1–10, 2014.

[3] A. Pathania and J. Henkel. Hotsniper: Sniper-based toolchain for many-core thermal simulations in open systems. *IEEE Embedded Systems Letters*, 11(2):54–57, 2018.

[4] C. Bienia, S. Kumar, J.P. Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In *17th international conference on Parallel architectures and compilation techniques*, pages 72–81, 2008.

[5] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. The splash-2 programs: Characterization and methodological considerations. *ACM SIGARCH computer architecture news*, 23(2):24–36, 1995.

[6] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *IEEE Transactions on very large scale integration (VLSI) systems*, 14(5):501–513, 2006.

[7] S. Pagani, J.J. Chen, M. Shafique, and J. Henkel. Matex: Efficient transient and peak temperature computation for compact thermal models. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1515–1520. IEEE, 2015.

[8] F. Beneventi, A. Bartolini, A. Tilli, and L. Benini. An effective gray-box identification procedure for multicore thermal modeling. *IEEE Transactions on Computers*, 63(5):1097–1110, 2012.

[9] G. Bhat, G. Singla, A. K Unver, and U. Y Ogras. Algorithmic optimization of thermal and power management for heterogeneous mobile platforms. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(3):544–557, 2017.

[10] T.E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2011.

[11] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi. The mcpat framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(1):1–29, 2013.

[12] J. Henkel, H. Khdr, S. Pagani, and M. Shafique. New trends in dark silicon. In *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015.

[13] Z. Chen and D. Marculescu. Distributed reinforcement learning for power limited many-core system performance optimization. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1521–1526. IEEE, 2015.

[14] M. Rapp, M. Sagi, A. Pathania, A. Herkersdorf, and J. Henkel. Power- and cache-aware task mapping with dynamic power budgeting for many-cores. *IEEE Transactions on Computers*, 69(1):1–13, 2019.

[15] M. Rapp, B. Sikal, H. Khdr, and J. Henkel. Smartboost: Lightweight ml-driven boosting forthermally-constrained many-core processors. In *58th ACM/IEEE Design Automation Conference (DAC)*, 2021.

[16] M. Rapp, A. Pathania, T. Mitra, and J. Henkel. Neural network-based performance prediction for task migration on s-nuca many-cores. *IEEE Transactions on Computers*, 2020.

[17] G. Bhat, S. Gumussoy, and U. Y Ogras. Power-temperature stability and safety analysis for multiprocessor systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):1–19, 2017.

[18] H. Wang, D. Tang, M. Zhang, S.X.D Tan, C. Zhang, H. Tang, and Y. Yuan. Gdp: A greedy based dynamic power budgeting method for multi/many-core systems in dark silicon. *IEEE Transactions on Computers*, 68(4):526–541, 2018.

[19] X. Hu, Y. Xu, J. Ma, G. Chen, Y. Hu, and Y. Xie. Thermal-sustainable power budgeting for dynamic threading. In *51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2014.

[20] G. Chen, Y. Xu, X. Hu, X. Guo, J. Ma, Y. Hu, and Y. Xie. Tsocket: Thermal sustainable power budgeting. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 21(2):1–22, 2016.