DBP: Distributed Power Budgeting for Many-Core Systems in Dark Silicon

Hai Wang, *Member, IEEE,* Wenjun He, Qinhui Yang, Xizhu Peng, and He Tang

***Abstract*—Power budget is an important power constraint provided to guarantee the thermal reliability of an integrated system. In this work, we present DBP, a DistriButed Power budgeting method, for dark silicon many-core systems. In DBP, there are two new techniques proposed to bring accurate and optimized power budget in a distributed way. First, a distributed active core locating technique is developed to find an active core distribution that leads to a high power budget. Second, a distributed power budget computing technique is introduced which computes the power budget for each active core accurately. Experiments show DBP outperforms the state-of-the-art power budgeting methods thermal safe power (TSP) and greedy dynamic power (GDP) on many-core dark silicon systems by providing a high and accurate power budget with low overhead and good scalability.**

***Index Terms*—Power budget, distributed computing, thermal modeling, many-core system.**

I. Introduction

As technology advances, the high power density of the integrated circuit (IC) systems leads to serious thermal related problems. Recently, the broke of the Dennard scaling makes the problems even worse and finally results in the dark silicon phenomenon in today's multi/many-core systems, where only part of the system components are powered on at the same time to avoid thermal violation [1], [2].

In order to guarantee thermal safety, power budget, which is the highest power allowed to be consumed under the temperature constraint, is used as the power constraint for the IC systems. For the dark silicon systems, since there are different active core numbers and distributions at runtime, the power budget provided by the traditional power budgeting method like thermal design power (TDP) can be overly pessimistic [3]. To mitigate the shortcomings of TDP in dark silicon systems, thermal safe power (TSP) [3] was proposed which provides a higher power budget than TDP by considering the active core number. Recently, greedy dynamic power (GDP) [4] was proposed by further considering the active core distribution and transient thermal effects. Based on the dark silicon power budgeting methods mentioned above, many thermal/power management techniques were introduced for the multi-core dark silicon systems [5], [6], [7].

However, power budgeting for the dark silicon many-core systems remains an open problem. Existing methods such as TSP and GDP have limitations when applied to the many-core systems. To be specific, TSP does not have the ability to optimize the active core distribution and consider transient effects accurately, which leads to a conservative power budget and suppressed performance of the many-core systems. GDP, although does not have the previous problems of TSP, has a computational cost related to the system core number as a centralized method. Thus, it has a large power budget computing delay if applied to the system with massive cores.

In this brief, we present DistriButed Power (DBP) budgeting method for many-core systems in dark silicon. With a new distributed active core locating algorithm and a new local power budget computing technique, DBP is able to find a good active core distribution and provide an accurate power budget considering transient thermal effects, with $\mathcal{O}(1)$ computational complexity locally for each active core.

II. The Distributed Power Budgeting Method

The flow of DBP has two stages. In the first stage, each active core is located in a distributed way which leads to a high power budget, as presented in Section II-A. In the second stage, we compute the power budget for each located active core in a distributed way based on a local thermal model, as presented in Section II-B. Finally, we analyze the computational complexity of DBP in Section II-C.

A. Distributed active core locating

Locating active cores is important in power budgeting of dark silicon systems because a better active core distribution leads to a higher power budget. In the existing power budgeting method GDP [4], a sub-optimal active core distribution is found by solving a power budget maximizing problem approximately. Such a centralized active core locating algorithm has a computational complexity that depends on the core number, making it inapplicable to the systems with an extremely large number of cores. To solve this problem, we propose a distributed active core locating algorithm for DBP with a computational complexity of $\mathcal{O}(1)$ for each active core.

Before presenting the new algorithm, we define the *adjacent* cores of the core in the ith row and jth column (denote this location as (i, j)) as the 4 cores that locate at $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$, respectively, as shown in Fig. 1.

The new algorithm is based on an important observation from the previous studies in power budgeting of dark silicon systems [3], [4]: a more uniform active core distribution generally leads to a higher power budget. This observation can be explained that the power budget of an active core will

This research is supported by National Natural Science Foundation of China under Grant 61974018 and Grant 61801087.

H. Wang, W. He, Q. Yang, X. Peng, and H. Tang are with School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 610054 China. H. Wang is also with Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou, 313000 China.

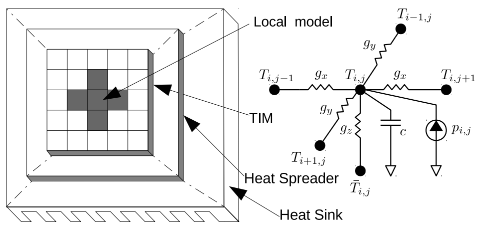
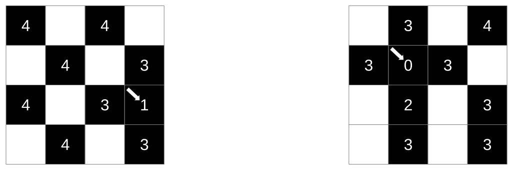


Fig. 1: The structure of a packaged many-core system and the local thermal model of the core at the ith row and jth column (location (i,j)). The cores at the locations $(i-1,j)$, $(i+1,j)$, $(i,j-1)$, and $(i,j+1)$ are defined as its adjacent cores.



(a) The lower bound $n_m = 1$ appears in the most uniform distribution.

(b) We can always find candidate with $n_a \le n_m$ in any other distributions.

Fig. 2: Illustration of finding the lower bound of n_{th} in a 16-core system with 7 existing active cores. The white blocks represent active cores. The number in each candidate (black block) is its n_a.

be suppressed if its adjacent cores are also active due to the thermal coupling effect. According to this observation, the goal of our distributed active core locating algorithm is simplified to distributing active cores evenly to avoid active core clusters.

To achieve this goal, the basic idea of the new algorithm is to locate an active core randomly to an inactive location (called *candidate*), with some criteria to avoid forming active core clusters. Specifically, to locate an active core, we first pick a candidate randomly. If this candidate's existing adjacent active core number (denoted as n_a) is no larger than a threshold n_{th} (indicating no active cluster will be formed if we activate this candidate), then we will locate the active core to this candidate.

Two problems need to be solved before we can realize the basic idea above. First, if the randomized candidate does not satisfy $n_a \le n_{th}$, how do we proceed? This problem can be solved with the similar strategy used in the Hash table conflict resolving: we simply test the next candidate if the current candidate fails, until an eligible candidate is found.

Second, how do we set a proper threshold n_{th}? Clearly, n_{th} depends on the existing active core number: it should be larger if there are more existing active cores. So, for a given existing active core number, there is a lower bound of n_{th}, denoted as n_m: by setting $n_{th} < n_m$, there may not exist any candidate that satisfies $n_a \le n_{th}$. The lower bound n_m can be found as the smallest adjacent active core number of the candidates in the most uniform active core distribution, with an example shown in Fig. 2a. Then, setting the threshold as $n_{th} = n_m$ guarantees to find a candidate in all distributions, because a candidate with $n_a \le n_m$ always exists in a less uniform distribution, with an example shown in Fig. 2b.

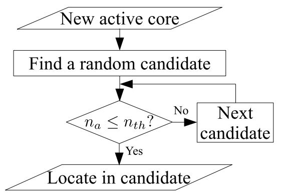
However, setting $n_{th} = n_m$ can be too strict practically, because there may exist only very few eligible candidates, which are difficult to find in a random-based algorithm. On the contrary, it is also undesirable to use a large n_{th}, because many unwanted active core clusters may appear. As a result, one can adjust n_{th} within $[n_m, 4]$ to balance the active core locating quality and computing cost.

The distributed active core locating flow is given in Fig. 3a.

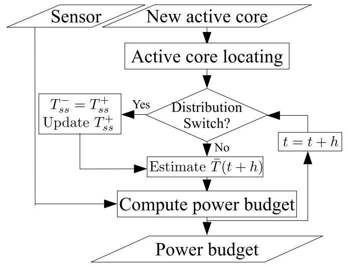
B. Distributed power budget computing

With the active cores located, now we present the new power budget computing method for each active core.

1) The local thermal model: First, we show the local thermal model. Although there are some existing models (like [8], [9]), our target is to build a local model that specially works for the dark silicon power budgeting, with small computing overhead and without power consumption information.



(a) The active core locating flow.



(b) The whole DBP flow with the active core locating flow shown in Fig. 3a.

Fig. 3: The flow of the distributed power budgeting method (DBP) for dark silicon many-core systems.

For a core in a many-core system, its temperature is determined by the temperatures of its adjacent cores, the temperature of the package, and the power of itself.

Specifically, for the core in the ith row and jth column of the many-core system, let us denote its temperature as $T_{i,j}$, its power as $p_{i,j}$, and its corresponding package node temperature as $\bar{T}_{i,j}$. By using the well known duality between thermal system and electrical circuit system, the equivalent local thermal circuit of this core is drawn in Fig. 1, where g_x, g_y, g_z are the thermal conductance in the x, y, z directions, respectively, c is the thermal capacitance of each core.

According to the local thermal circuit, we can write the local thermal model in ordinary differential equation as

$$c\frac{\mathrm{d}T(t)}{\mathrm{d}t} = -g_l T(t) + p_a(t) + g_z \bar{T}(t) + p(t), \tag{1}$$

where $g_l = 2g_x + 2g_y + g_z$, and we count the thermal influence from the adjacent cores ($T_{i-1,j}$, $T_{i+1,j}$, $T_{i,j-1}$, $T_{i,j+1}$) as a power source $p_a(t)$:

$$\begin{aligned} p_a(t) = & g_x T_{i,j-1}(t) + g_x T_{i,j+1}(t) \\ & + g_y T_{i-1,j}(t) + g_y T_{i+1,j}(t), \end{aligned} \tag{2}$$

and use $T(t)$, $\bar{T}(t)$, $p(t)$ to represent $T_{i,j}(t)$, $\bar{T}_{i,j}(t)$, $p_{i,j}(t)$ respectively to simplify notation.

2) Formulation of the local power budget computing: The problem of power budgeting is to compute the maximum power allowed to be consumed in a future time span called the power budgeting time step (denoted as h).

To describe the distributed power budgeting problem mathematically, we first discretize the original local thermal model according to the power budgeting time step h as

$$\begin{aligned}(\frac{c}{h}+g_l)T(t+h) =&\frac{c}{h}T(t)+p_a(t+h)\\ &+g_z\bar{T}(t+h)+p(t+h),\end{aligned} \tag{3}$$

In the power budgeting problem, p is the power budget to be solved, which should be treated as a constant during each power budgeting step. In order to solve the power budget p, the core temperature at the end of the step $T(t+h)$ should be set as the threshold temperature: $T(t+h)=T_{th}$.

Now, we can rewrite (3) for the distributed power budgeting problem as

$$(\frac{c}{h}+g_l)T_{th}=\frac{c}{h}T(t)+p_a(t+h)+g_z\bar{T}(t+h)+p. \tag{4}$$

Then, p, the power budget of the local core, can be solved from the equation above.

In equation (4), T_{th} is provided by the user. $T(t)$, which is the current temperature of the local core, can be read from the thermal sensor. $p_a(t+h)$, the thermal influence from the adjacent cores, is generally unknown, but can be estimated using the current temperature values of the adjacent cores ($T_{i-1,j}(t)$, $T_{i+1,j}(t)$, $T_{i,j-1}(t)$, $T_{i,j+1}(t)$). Since temperature does not change drastically thanks to the low pass filter property of the thermal system, these current temperatures lead to a good approximation.

Except for p to be solved, the only unknown variable in (4) is the temperature of the package node $\bar{T}$, because there is usually no thermal sensor located in the package. Since the package temperature has a significant impact on the power budget of the core [4], we need to estimate $\bar{T}$ in (4) in order to compute the power budget accurately, as shown next.

3) Package temperature estimation: At runtime, the dark silicon pattern (active core distribution) may change when tasks are assigned to new active cores or when tasks are completed in some active cores. The package temperature will also change with the active core distribution. In this part, we show how to estimate the package node temperature $\bar{T}(t+h)$, in order to solve equation (4) for power budget p.

Since the package temperature is influenced by the die temperature, it will reach a steady state value if the die temperature is steady for a period of time. We observe that for each active core distribution, the steady state package node temperature is mainly determined by the adjacent active core number of this local core, the target temperature value T_{th}, and the ratio of active cores to total cores. As a result, we can estimate the steady state temperature of the package node, denoted as $\bar{T}_{ss}$, using the following equation

$$\bar{T}_{ss}=\alpha_1 n_a+\alpha_2 T_{th}+\alpha_3 r, \tag{5}$$

where n_a is the adjacent active core number of the local core, r is the ratio of active cores to total cores, α_1, α_2, and α_3 are parameters fitted through training.

With the steady state package temperature $\bar{T}_{ss}$ available, we can estimate the transient package node temperature $\bar{T}(t+h)$. To illustrate the idea of the package temperature estimation, we plot the transient temperature of the package node in Fig. 4. From the figure, $\bar{T}(t+h)$ can be represented as

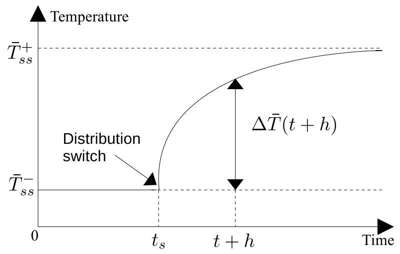


Fig. 4: Illustration of the transient curve of a local package node temperature $\bar{T}(t)$. The active core distribution switches at time t_s.

$$\bar{T}(t+h)=\bar{T}_{ss}^{-}+\Delta\bar{T}(t+h), \tag{6}$$

where $\bar{T}_{ss}^{-}$ is the steady state package temperature with the previous active core distribution, $\Delta\bar{T}(t+h)$ is the temperature change from the previous steady state package temperature.

According to equation (6), we still need to estimate $\Delta\bar{T}(t+h)$ in order to get $\bar{T}(t+h)$. Fortunately, with the previous steady state temperature ($\bar{T}_{ss}^{-}$) and the current steady state temperature ($\bar{T}_{ss}^{+}$) available, we can estimate $\Delta\bar{T}(t+h)$ according to the system thermal dynamics.

Specifically, we can write the following full thermal model of the many-core system with the local package node temperature $\bar{T}(t)$ (i.e., $\bar{T}_{i,j}(t)$) as the output:

$$\begin{aligned}GY(t)+C\frac{\mathrm{d}Y(t)}{\mathrm{d}t}&=BP(t)\\ \bar{T}(t)&=LY(t),\end{aligned} \tag{7}$$

where $Y(t)$ is the thermal vector of all temperature nodes; $P(t)$ is the power vector of all power sources; G and C are the thermal conductance matrix and capacitance matrix, respectively; B is the power mapping matrix; L is an output selection vector which selects the local package node temperature out of the thermal vector.

Using the full thermal model (7), we can get $\Delta\bar{T}(t+h)$ as

$$\begin{aligned}\Delta\bar{T}(t+h)=&\bar{T}(t+h)-\bar{T}(t_s)\\ =&L(e^{-(t+h-t_s)C^{-1}G}Y(t_s)\\ &+\int_{t_s}^{t+h}e^{-(t+h-\tau)C^{-1}G}C^{-1}BP\mathrm{d}\tau-Y(t_s))\\ =&L(I-e^{-(t+h-t_s)C^{-1}G})(G^{-1}BP-Y(t_s)).\end{aligned} \tag{8}$$

Please note that because $\bar{T}_{ss}^{+}=LG^{-1}BP$ and $\bar{T}_{ss}^{-}=LY(t_s)$, $\Delta\bar{T}(t+h)$ can be approximated as

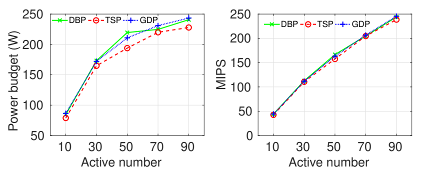
$$\Delta\bar{T}(t+h)\approx(1-e^{-\beta(t+h-t_s)})(\bar{T}_{ss}^{+}-\bar{T}_{ss}^{-}), \tag{9}$$

where β is a fitting parameter. Please note that higher order approximation is also possible with more fitting parameters.

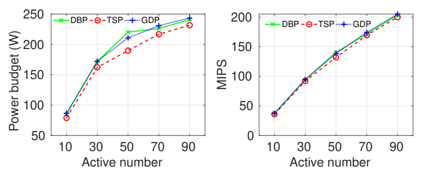
Finally, according to (6), $\bar{T}(t+h)$ can be estimated as

$$\bar{T}(t+h)\approx\bar{T}_{ss}^{-}+(1-e^{-\beta(t+h-t_s)})(\bar{T}_{ss}^{+}-\bar{T}_{ss}^{-}). \tag{10}$$

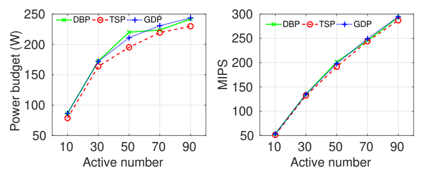
With the estimated package node temperature $\bar{T}(t+h)$, the power budget of the local core can be solved from equation (4).



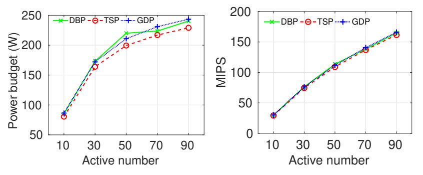
(a) bzip2



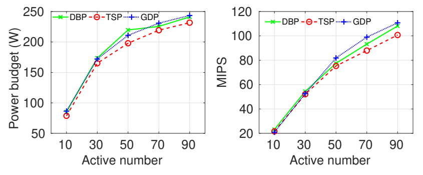
(b) gobmk



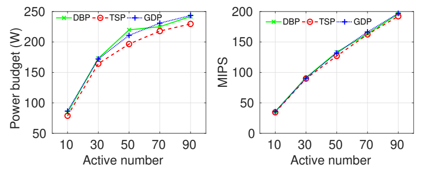
(c) hmmer



(d) lbm



(e) mcf



(f) milc

Fig. 5: The power budget and the average performance (MIPS) of the 100-core system with the three power budgeting method.

The whole flow of the distributed power budgeting method DBP is summarized in Fig. 3b.

C. Computational complexity analysis

The new method mainly contains the active core locating and the power budget computing. For each active core, the locating algorithm has a computational complexity of $\mathcal{O}(1)$, because it only requires one random trial and several extra trials if the previous trial fail, which is irrelevant to the problem size (the total core number and the active core number).

The power budget computing for each local active core also has a complexity of $\mathcal{O}(1)$, because both the package node temperature estimation in (10) and the power budget solving in (4) are irrelevant to the problem size.

III. Experimental Results

We implemented DBP using Matlab and performed experiments on a PC with Intel i7-9750H CPU and 8 GB memory.

To show the advantage of DBP, we compare it with two state-of-the-art power budgeting methods for dark silicon systems: thermal safe power (TSP) [3] and greedy dynamic power (GDP) [4]. In this experiment, TSP provides the power budget for a given active core distribution. Since TSP does not have the ability to locate the active cores, it uses the active core distribution determined by DBP unless mentioned explicitly.

The experiments are performed mainly on a 100-core system, with scalability/runtime analysis also performed on a 49-core system and a 144-core system, assuming mesh NoC is used such that the communication latency between the adjacent cores is low. We use HotSpot [10] to build the system thermal model, which is used to perform thermal simulation and extract the local thermal model for DBP. The power budgeting time step is set as 10 ms for all methods. For the 100-core system, we set $n_{th} = 0, 1, 2, 3, 4$ for the existing active core number range of $[0, 35]$, $[36, 48]$, $[49, 57]$, $[58, 74]$, $[75, 100]$, respectively, which balances the active core locating quality and computing speed.

First, we test the accuracy of $\bar{T}_{ss}$ trained through linear regression. By randomly activating the cores in the 100-core system, 1×10^6 error samples of $\bar{T}_{ss}$ are collected, with an average absolute error of 0.12 °C and variance of 0.024. We also tested the transient thermal estimation error of the local model (eq (4)) in a power budgeting problem with ambient temperature as 45 °C and threshold temperature as 80 °C. Through 60 simulations of 1 s duration (error sample step 10 ms) after a random active core map switching, we collected around 2.8×10^5 error samples, where the average absolute error is 1.2 °C and the variance is 1.6.

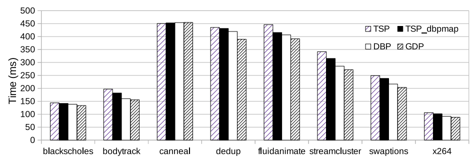
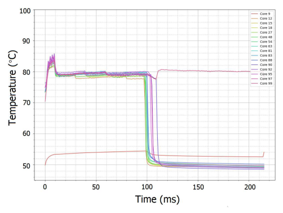


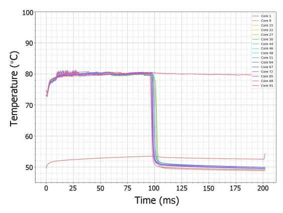
Fig. 6: The runtime of PARSEC benchmarks with different power budgeting methods. TSP uses the default first unused mapping and TSP_dbpmap shares the DBP mapping.

Then, we show the power budget and system performance (million instructions per second (MIPS)) results of the 100-core system with different methods in Fig. 5. In this experiment, the system power is obtained using Wattch by running the SPEC CPU 2006 benchmarks on each active core with Alpha architecture. The ambient temperature and the threshold temperature are set as 20 °C and 80 °C, respectively. We can see that DBP brings a similar power budget compared with GDP in all benchmarks. The power budget of TSP is significantly lower than the other two, because TSP, being the only static power budgeting method, can only provide a conservative power budget based on the steady state thermal state. Thanks to the higher power budgets, the system performance is also higher with DBP and GDP than with TSP, as shown in Fig. 5. Please note that DBP does not necessarily outperform GDP in system performance because DBP can be interpreted as a distributed approximation of the centralized GDP method.

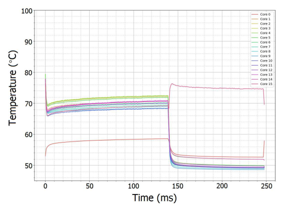
Next, we analyze the multi-threaded performance of the 100-core system with DBP. This experiment is performed using HotSniper [11] running PARSEC benchmarks with Intel X86 architecture and the other default settings. The ambient



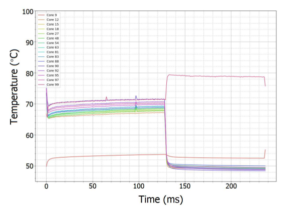
(a) DBP.



(b) GDP.



(c) TSP first unused mapping.



(d) TSP DBP mapping.

Fig. 7: Transient temperatures of the 100-core system running "swaptions" benchmark with different power budgeting methods, simulated using HotSniper with the threshold of $80\,^\circ\mathrm{C}$.

temperature and the threshold temperature are set as $45\,^\circ\mathrm{C}$ and $80\,^\circ\mathrm{C}$, respectively. TDP is $100\,\mathrm{W}$. Power budgeting is activated every $10\,\mathrm{ms}$ and when the active core number changes. We have tested all 8 supported PARSEC benchmarks with the largest recommended thread number, by running simlarge dataset except for "canneal" and "dedup" running simsmall due to their excessive simulation time and "x264" running simmedium because of the inability to run simlarge.

The runtime results of the PARSEC benchmarks are given in Fig. 6. GDP generally leads to the shortest runtime. DBP usually finishes second. It is slower than GDP mainly due to the lacking of global information as a distributed method. For the two TSP methods, TSP with DBP mapping performs better than the original TSP with first unused mapping, because active core clusters are avoided. For benchmark "canneal", all methods perform similarly. This is because "canneal" is a memory-intensive application with low CPU utilization, power budget is not its major performance constraint.

The transient temperature results of the "swaptions" benchmark are shown in Fig. 7. We can see that all methods lead to safe temperatures except for a short and small temperature overshot at the beginning with DBP, caused by the distributed transient temperature estimation inaccuracy. With higher power budgets, DBP and GDP bring the active cores' temperatures closer to the threshold than TSP, which explains their superiority in system performance.

Finally, we demonstrate the computing overhead of DBP, with the runtime results shown in Table I. As a distributed method, DBP shows a great scalability advantage as its runtime is both small and irrelevant to the problem size (core number and active core number). Although performs best in power budgeting accuracy, GDP has the worst performance in overhead as expected. TSP performs relatively well in overhead with a small but still larger runtime than DBP. We remark that TSP does not need to re-compute the power budget unless the active core distribution changes. However, its overhead grows with the problem size. It also does not have the active core locating ability and cannot compute power budget according to the current thermal state as shown previously.

TABLE I: The average computing times of different power budgeting methods for one power budgeting time step.

Core #	Active #	DBP (ms)	GDP (ms)	TSP (ms)
49	5	0.03	0.14	0.02
	25	0.03	0.59	0.05
	44	0.03	1.36	0.08
100	10	0.03	0.26	0.02
	50	0.03	1.72	0.09
	90	0.03	4.63	0.17
144	14	0.03	0.42	0.03
	72	0.03	3.67	0.14
	130	0.03	10.82	0.28

IV. Conclusion

In this brief, we have presented a distributed power budgeting method called DBP for many-core systems in dark silicon. DBP locates each active core to form a good active core distribution and computes the power budget for each active core, both in a distributed way. Experiments show that DBP outperforms the state-of-the-art power budgeting methods TSP and GDP on many-core systems in dark silicon, by achieving an accurate power budgeting and a low computing overhead with good scalability at the same time.

References

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

[2] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The EDA challenges in the dark silicon era," in *Proc. Design Automation Conf. (DAC)*, June 2014.

[3] S. Pagani, H. Khdr, J.-J. Chen, M. Shafique, M. Li, and J. Henkel, "Thermal safe power (TSP): Efficient power budgeting for heterogeneous manycore systems in dark silicon," *IEEE Trans. on Computers*, vol. 66, no. 1, pp. 147–162, January 2017.

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

[5] 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. Design Automation Conf. (DAC)*, May 2013.

[6] H. Khdr, S. Pagani, M. Shafique, and J. Henkel, "Thermal constrained resource management for mixed ILP-TLP workloads in dark silicon chips," in *Proc. Design Automation Conf. (DAC)*, June 2015.

[7] A. Kanduri, M.-H. Haghbayan, A. M. Rahmani, M. Shafique, A. Jantsch, and P. Liljeberg, "adBoost: Thermal aware performance boosting through dark silicon patterning," *IEEE Trans. on Computers*, vol. 67, no. 8, pp. 1062–1077, August 2018.

[8] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini, "Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller," *IEEE Trans. on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 170–183, January 2013.

[9] R. Diversi, A. Bartolini, and L. Benini, "Thermal model identification of computing nodes in high-performance computing systems," *IEEE Trans. on Industrial Electronics*, vol. 67, no. 9, pp. 7778–7788, September 2020.

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

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