A Fast Energy-Efficient Light Core Distribution Estimation Method for Multi-Core Systems in Dark Silicon

Jiachun Wan*, Hai Wang*, Sheldon X.-D. Tan†, Chi Zhang*, Yuan Yuan‡, Keheng Huang§, and Zhenghong Zhang§ *School of Microelectronics & Solid-State Electronics,
University of Electronic Science & Technology of China, Chengdu, 610054 China †Department of Electrical Engineering, University of California, Riverside, CA 92521 USA ‡School of Automation Engineering, University of Electronic Science & Technology of China, Chengdu, 610054 China §Southwest China Research Institute of Electronic Equipment, Chengdu, 610036 China

I. Introduction

In the coming many-core era, due to the tight power budget, power efficiency is cricital for multi-core chip design. In the previous work by Dong Hyuk Woo, evaluation of energy efficiency on the basis of performance and power (PPW) models is developed, which shows the tendency of PPW with number of cores. However, due to the dependency of leakage current on temperature and the prevalence of DVFS and dark silicon in multicore processor, the speedup potential given by existing extending methods of Amdahl's law no longer works out. In this paper, We implement dark silicon and thermal model into the evaluation of PPW to provide an alternative suitable with the dark silicon era.

II. Prior work

In this section, we briefly review some important researches in extended Amdahl's law, especially in energy-efficient area.

Extended Amdahl's law in energy-efficient area is widely studied in recent years. Woo and Lee extended Amdahl's law for energy-efficient parallel computation [?], who analyzed the energy efficiency of sysmmetric superscalar processor, symmetric processor with smaller and power-efficient cores, and asymmetric many-core processor. By taking the overhead of data preparation into consideration, the performance-energy efficiency can be reevaluated []. The performance change by tuning the operating frequency of cores can also be implemented to achieve better energy efficiency [?], [?]; the dynamic voltage/frequency scaling (DVFS) and heterogeneous microarchitectures (HMs) are also compared in terms of energy efficiency [?].

The works mentioned above share a common problem: they have difficulty in considering the temperature dependent leakage power for energy efficiency estimation, the reason is that most of the work set the power consumption of idle cores to be a constant. Only few research set the power consumption of idle cores to be a variable. For instance, in [?], the fraction of power a processor consumes in idle state is dependent upon the number of active cores for many-core platform. However, the leakage power it provides is too coarse to be used for energy efficiency estimation.

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

III. Background

In this section, we first present the Amdahl's law and performance-per-watt's traditional expression in multi-core processor. Then the static power modeling and thermal modeling is introduced, with which we can revise the expression of performance-per-watt in multi-core processor.

A. Amdahl's law

Amdahl's law put an upper bound for the speedup that a multi-core processor can achieve by parallelization as:

$$Perf = \frac{1}{(1-f)+f/n} \tag{1}$$

where n is the number of cores, and f is the fraction of a program's execution time that is parallelizable $(0 \leq k \leq 1)$. Noted that, (1) is purely theroretical for it doesn't consider any constraints such as power budget.

With the failure of Dennard scaling, Dynamic Voltage and Frequency Scaling (DVFS) has been widely applied in modern processors, which is considered to be an effective technique to achieve trade-off between performance and power consumption. Once a high temperature is detected, DVFS reduces the voltage and frequency of the corresponding core to lower its temperarue. Also DVFS is an effective method to reduce the operating frequency of cores, it's not always true that lower frequency leads to higher energy efficiency, due to the fact that low frequency increases the task execution time.

The DVFS strategy in this paper is to set the DVFS stage of each core as high as possible without violating the thermal constraint.

Considering the performance deterioration introduced by DVFS, we set the ratio of operating frequency with the full frequency as the current core's performance:

$$D = \frac{F_o}{F_f}, \tag{2}$$

Amdahl's law with DVFS can therefore be expressed as:

$$Perf = \frac{1}{\frac{1-f}{D_a} + \frac{f}{\sum_{i=1}^{n} D_i}}, \tag{3}$$

where D_a stands for the performance of the active core during sequential computation process, D_i stands for the performance of each of the active cores during parallel computation process.

The average power consumption of the many-core processor with DVFS is as follows:

$$W = \frac{P_1 \times \frac{1-f}{D_a} + P_n \times \frac{f}{\sum_{i=1}^{n} D_i}}{\frac{1-f}{D_a} + \frac{f}{\sum_{i=1}^{n} D_i}}, \tag{4}$$

in which, P_1 is the power consumption during the sequential computation process, P_n is the power consumption during the parallel computation process.

$Perf/W$ of a multi-core processor is expressed as:

$$\begin{aligned} \frac{Perf}{W} &= \frac{1}{\frac{1-f}{D_a} + \frac{f}{\sum_{i=1}^{n} D_i}} \times \frac{\frac{1-f}{D_a} + \frac{f}{\sum_{i=1}^{n} D_i}}{P_1 \times \frac{1-f}{D_a} + P_n \times \frac{f}{\sum_{i=1}^{n} D_i}} \\ &= \frac{1}{P_1 \times \frac{1-f}{D_a} + P_n \times \frac{f}{\sum_{i=1}^{n} D_i}} \end{aligned} \tag{5}$$

Likewise, $Perf/J$ of a multi-core processor is expressed as:

$$\frac{Perf}{J} = \frac{1}{\frac{1-f}{D_a} + \frac{f}{\sum_{i=1}^{n} D_i}} \times \frac{1}{P_1 \times \frac{1-f}{D_a} + P_n \times \frac{f}{\sum_{i=1}^{n} D_i}} \tag{6}$$

B. *Static power modeling*

It is widely acknowledged that the total power of a chip is composed of dynamic and static power. The dynamic power is dependent on the activities of the chip, therefore it's easily estimated by methods implemented with performance counter. Yet the static power p_s of the chip is caused by leakage current I_{leak} as:

$$p_s = V_{dd} I_{leak} \tag{7}$$

Due to the non-linear relationship between I_{leak} and temperature, static power is also sensitive to temperature, which makes it hard to obtain.

I_{leak} is composed of many components, including subthreshold current, gate current, reverse-biased junction leakage current, et cetera. Among which, subthreshold current and gate current are the main parts of leakage current, therefore I_{leak} can be approximated as:

$$I_{leak} = I_{sub} + I_{gate} \tag{8}$$

Noted that I_{gate} is cause by tunneling between the gate terminal and the other three terminals, does not depend on temperature and can be considered as a technology-dependent constant. Yet I_{sub} is considered to be highly related to temperature, and can be modeled in the commonly accepeted MOSFET transitor model BSIM 4 as:

$$I_{sub} = K v_T^2 e^{\frac{V_{GS}-V_{th}}{\eta v_T}} (1 - e^{\frac{-V_{DS}}{v_T}}) \approx K v_T^2 e^{\frac{V_{GS}-V_{th}}{\eta v_T}} \tag{9}$$

Apparently, the leakage current has a complex relationship with temperature. In this work, we use (7), (8), and (9) to model the static power considering such relationship. The parameters of leakage current can be obtained by curve fitting using HSPICE simulation data. In order to see the accuracy of the model used, Fig. 1 shows an HSPICE simulation result of leakage using TSMC $65\,\mathrm{nm}$ process model and its curve fitting result using approximate leakage model. From the figure, we can see that the static power model has high accuracy for all common temperatures of IC chips.

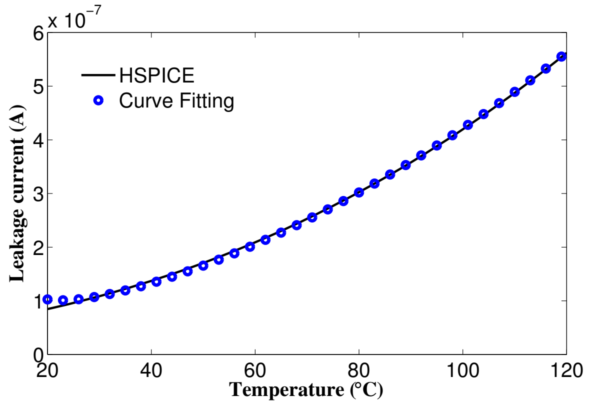


Fig. 1: Comparison of leakage of a TSMC $65\,\mathrm{nm}$ process MOSFET from HSPICE simulation with its curve fitting result using (9). An example of temperature region division is also shown in the figure, which will be discussed later.

We can conclude that the static power distribution depends mainly on the temperature distribution for a certain chip with constant physical parameters. Since temperature also depends on power, in order to view the whole picture, thermal model of IC chip is used to describe temperature's dependency on power as shown next.

C. *Thermal modeling*

In this work, the multi-core dark silicon system is packaged in a common structure in Fig. x. To estimate the power consumption of a IC chip, we first divide the chip and its package into multiple blocks called thermal nodes, with partition granularity determined by the accuracy requirements. For the dark silicon multi-core system (a 16-core chip's floorplan example is shown in Fig. x), 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 these thermal nodes are determinded, which model the thermal transport and power response behaviors. Please note that multi-core system floorplan different from the one shown in Fig. x are fullly compatible with this work, and each core's internal structure can also be modeled if necessary. With above mentioned information, the thermal mdoel for a chip with n total thermal nodes can be generated:

$$\begin{aligned} GT(t) + C\frac{dT(t)}{dt} &= BP_T \\ T_c(t) &= LT_t \end{aligned} \tag{10}$$

where $T(t) \in \mathbb{R}^n$ is the temperature vector (distinguished from T_p, which is a scalar representing temperature at only one place), representing temperatures at n places of the chip and package; $G \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{n\times n}$ contain equivalent thermal resistance and capacitance information respectively; $B \in \mathbb{R}^{n\times l}$ stores the information of how powers are injected into the thermal nodes; $P(T,t) \in \mathbb{R}^l$ is the power vector, which contains power consumptions of l components on chip, including both dynamic power vector P_d and static power vector P_s, i.e., $P(T,t) = P_s(T,t) + P_d(t)$, reminding that static power $P_s(T,t)$ is actually a function of temperature T; $T_c(t) \in \mathbb{R}^m$ is the output temperature vector, containing only temperatures of thermal nodes that the user is interested in, for example, thermal nodes on the chip only (excluding package thermal nodes); $L \in \mathbb{R}^{m\times n}$ is the corresponding output selection matrix which selects the m chip temperatures from $T(t)$.

By applying thermal model, the effect of temperature on cores can be introduced. Through appropriate search methods, the number and distribution of light core with the maximum PPW for number of light cores from 1 to n can be specified.

IV. New Method

This work aims to find the optimal number and distribution of light cores for a fixed set of cores to achieve the maximum performance-per-watt, with consideration of thermal constraints. In order to estimate the power consumption of a multi-core system for different number of light cores and various light core distributions, we first propose an iteration based full-chip power estimation method with brute force search strategy, which is accurate but very time-consuming. Furthermore, a non-iteration based method is proposed, which can implement local linearization to avoid time-consuming iterations. Additionally, a greedy based search strategy can be integrated into the non-iteration based power estimation method to achieve further acceleration.

A. Finding maximum PPW with brute force search strategy

In this paper, there are three different states for cores, dark state, active state and idle state. While in dark state, the core is completely shut down. In active state, the core functions normally, the power consumption of the core include P_s and P_d. In idle state, the power consumption of the core only comes from P_s. The latter two states can be called light state as a whole.

As discussed above, the maximum PPW for a multi-core chip corresponds to the minimum $P_1 \times (1-f) + P_n \times f/n$, let $P_a = P_1 \times (1-f) + P_n \times f/n$ to simplify notation. Noted 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. The task of finding the maximum PPW can be formulated as the following optimization problem

$$\begin{aligned} &\text{minimize } P_a \\ &\text{subject to} = \{\ T_c \preceq T_{th}. \end{aligned} \tag{11}$$

In order to solve such optimization problem, we have to go through all the possible light core numbers. For a multi-core system with n cores, to find the optimum light core number, that means to try light core numbers ranging from 1 to n. For each light core number, we will have to try all possible light core distributions to find the optimum light core distribution. For each light core distribution, we have to decide which core is active and which cores are idle during sequential computation process that can lead to maximum PPW. The whole process is shown in Fig. x. Therefore, a solution of this optimization problem includes light core number, light core distribution and the state of light cores during sequential computation process.

The optimization problem above is a combinational problem, and its optimal solution can be found by brute force search of all possible combinations of the cores in the multi-core system. However, this brute force search is very computationally expensive. Especially in this case, the number of light core is not set, which means this brute force search strategy can only be applied to multi-core systems with a small core number.

B. Iteration based leakage-aware power estimation

For each of the light cores combination in the optimization problem, we can estimate the power consumption of the multi-core system in series computation process and parallel computation process, with which we could estimate the performance-per-watt of this combination. The steady state power is set by temperature, which is calculated using model (x) by neglecting the differential term $C\frac{dT(t)}{dt}$, leading to

$$T_c = L^T G^{-1} B P \tag{12}$$

It seems straightforward to implement the previous thermal model x for steady state power estimation. By simply providing $P_s(T)$ and P_d, we are able to calculate the temperature of each core T_c. However, such power estimation is valid for dynamic power only scenario and cannot be used when leakage current is considered. This is because $P_s(T)$ is a function of current temperature T_c, leading to the fact that once T_c is computed based on $P_s(T)$, $P_s(T)$ also needs to be updated based on T_c, which will only stop when steady state is reached.

Because of the dependency of strategy power on temperature, it's not straightforward to compute the strategy power and temperature of steady state based on current power. Iteration method can be implemented to solve such problem, the computation flow is shown in Fig.x.

The initial value of $P_s^0(T)$ is a guess we provide based on the process technology. The temperature distribution T^0 can be calculated with such guess. $P_s^1(T)$, the strategy power of next time step is updated with T^0. Next, the temperature distribution T^1 can be derived from $P_s^1(T)$, which concludes one iteration loop. Such iteration goes on until the convergence test is satisfied as $\|P_s^i(T)\text{-}P^(i-1)_s(T)\| < \epsilon$. Finally, the strategy power and temperature of steady state is outputted.

Combined with the brute force search strategy, the iteration based power estimation method can produce an accurate

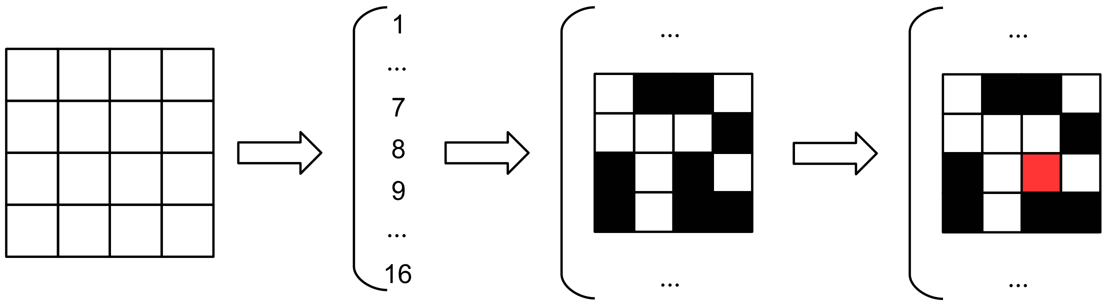


Fig. 2: some figure

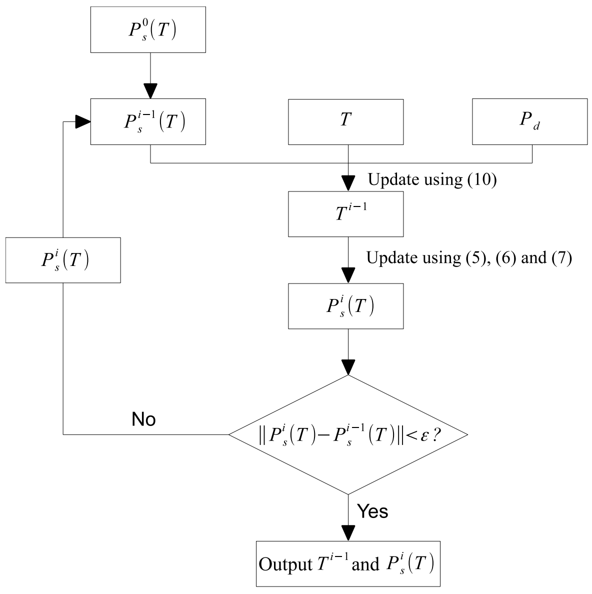

$P_s^0(T)$
$P_s^{i-1}(T)$
T
P_d
Update using (10)
T^{i-1}
Update using (5), (6) and (7)
$P_s^i(T)$
$P_s^i(T)$
No
$\|P_s^i(T) - P_s^{i-1}(T)\| < \varepsilon$?
Yes
Output T^{i-1} and $P_s^i(T)$

Fig. 3: Flow diagram of iteration based leakage-aware power estimation for one time step

outcome providing the ϵ is chosen to be small enough, which could be considered as the golden accuracy baseline. However, the simulation time for such iteration based method is way too long, especially for multi-core systems with a large core number.

C. Local linearized thermal model

The major difficulty of calculating leakage-aware power estimation comes from the nonlinear thermal model shown in (x), which is caused by the nonlinear relation between subthreshold current and temperature.To reduce the long computing time caused by iteration method, the leakage current I_{leak} can be linearized to eliminate the non-linearity between p_s and temperature, thus accelerating the computation. The basic idea is to approximate the original nonlinear leakage model using multiple new linear leakage models. By implementing such linear leakage models, we can reformulate the original nonlinear thermal model into multiple linear thermal models, such that traditional non-iteration based thermal estimation method can be applied.

In order to generate a linear leakage model, Taylor expansion is performed on the original I_{leak} model at a expansion point T_0. Thus the linearized relation of I_{leak} and temperature is obtained as:

$$\begin{aligned} I_{sub} =& K(\frac{k}{q})^2 e^{(}\frac{q(V_{GS}-Vth)}{\eta k T_{p0}} \\ & \times (T_{p0}^2 + (2T_{p0} - \frac{q(V_{GS}-V_{th})}{\eta k})(T_p - T_{p0})) \\ & + o[(T_p - T_{p0})^2], \end{aligned} \tag{13}$$

where $o[(T_p - T_{p0})^2]$ is the remainder. By ignoring the remainder, the linearized I_{sub}, denoted as I_{lin} can be expressed as:

$$\begin{aligned} I_{lin} =& K(\frac{k}{q})^2 e^{(}\frac{q(V_{GS}-Vth)}{\eta k T_{p0}} \\ & \times (T_{p0}^2 + (2T_{p0} - \frac{q(V_{GS}-V_{th})}{\eta k})(T_p - T_{p0})). \end{aligned} \tag{14}$$

Provided the actual temperature value T_p is close to the reference temperature point T_{p0}, the approximation accuracy of I_{lin} can be guaranteed. From previous research, it has been shown that due to the characteristic of today's semiconductor process, such local linear approximation of leakage current has high accuracy around the expansion point.

With the linearized relation of subthreshold current and temperature, the relation between strategy power and temperature in linear form can also be achieved as:

$$\begin{aligned} p_s &= V_{dd} I_{leak} \\ &= V_{dd} \times (I_{lin} + I_{gate}) \\ &= V_{dd} \times (I_{lin}(T_p) + I_{const}), \end{aligned} \tag{15}$$

where $I_{lin}(T_p)$ represents the terms associated with T_p in (x), I_{const} contains constant terms that are not associated with T_p and the gate leakage I_{gate}.

A new linear thermal model can be built based on this strategy power model. In order to do that, we need to integrate (x) into (x). Please note that (x) is in scalar form for only one certain thermal node while (5) is in vector/matrix form including information of all thermal nodes. Therefore, by collecting and accumulating scalars $I_{lin}(T_p)$ and I_{const} at multiple positions of the chip into vectors, we can rewrite (x) in vector/matrix form. The current variables can also be changed to power by multiplying voltage V_{dd}. The linearized strategy power equation in matrix form is

$$P_s = P_0 + A_s T \tag{16}$$

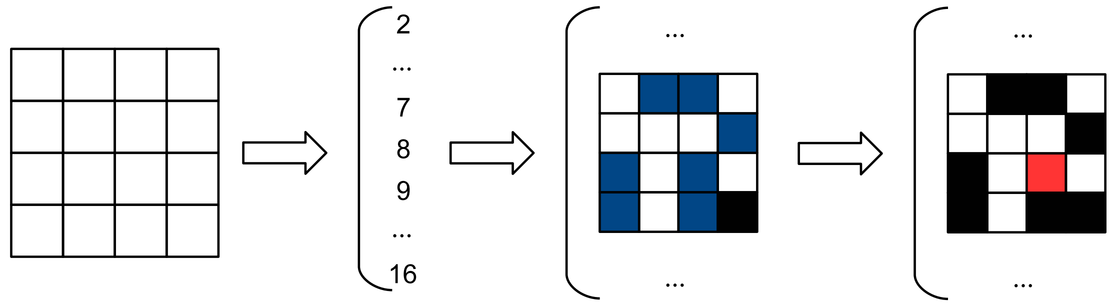


Fig. 4: some figure

where $P_0 \in \mathbb{R}^l$ is a known vector, with each element formed by terms not associated with T_p in (x) at each position of the chip. $A_s \in \mathbb{R}^{l\times n}$ is a known rectangular diagonal matrix (the left $l \times l$ block matrix is diagonal representing thermal nodes on the chip, and the right $l \times (n-l)$ block matrix is all zeros representing the thermal nodes of packages), with each diagonal element formed by the coefficient associated with T_p in (x) at each position of the chip.

Integrating (14) into (8), and let $G_l = G - BA_s$, we have

$$G_l T(t) + C\frac{dT(t)}{dt} = B(P_d + P_0) \tag{17}$$

With the obtained linear thermal model considering strategy power and eliminated the nonlinear relationship of strategy power and temperature, we can simulate the locally linearized leakage-aware thermal model just as in (8) by viewing "G_l" as the new "G" matrix and "$P_d + P_0$" as the new "P_T" vector.

D. Non-iteration based power estimation

Since we have generated the linearized thermal model as shown above, the Taylor expansion temperature points need to be determined, for they affect the linearized thermal model's accuracy greatly, also P_0 and A_s in (14) are formulated by the expansion point information. Therefore, we will discuss how to determine the Taylor expansion points for thermal nodes in the system.

As a property of Taylor expansion approximation, it is accurate only when the actual temperature T_p is close to the expansion point T_{p0}. As a result, in order to ensure the approximation accuracy, we want each expansion point T_{p0} to be close to the actual temperature T_p. To find a totally accurate outcome, $T_p = T_{p0}$ is necessary. However, such strategy requires T_{p0} to be updated for each iteration, which would lead to much longer computing time than we expected. In order to balance the accuracy and computing time, the remainder $o[(T_p - T_{p0})^2]$ introduced by the deviation of T_{p0} from T_p can be tolerated if T_{p0} is not far from T_p.

Considering that the temperature difference between the idle cores and active cores can be very large, it's not possible to find a single Taylor expansion temperature point that produces remainder $o[(T_p - T_{p0})^2]$ within tolerance for idle cores and active cores simultaneously.

In this work, we can notice that the temperature of cores can be classified into two parts, one is for active-state cores around $70\,°\mathrm{C}$, one is for idle-state cores around $40\,°\mathrm{C}$. Therefore, in order to balance the accuracy and computing cost, we set two Taylor expansion temperature points T_{p1} and T_{p2}, one for active cores, one for idle cores. As can be shown in the experimental results, the approximation accuracy of this strategy is satisfactory in this work.

E. Greedy search strategy based acceleration of power estimation

In previous sections, to find the optimal light core distribution for a multi-core system which leads to highest performance-per-watt, a combinational method with high complexity is implemented. However, it's especially impractical when the number of cores is too large. 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. Therefore, a greedy based method which can find a sub-optimal light core distribution with much less time consumption is applied.

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 a n-core system with n_a light cores, the basic idea of finding such sub-optimal light core distribution is described as follows: we first find the optimal solution for only one light core. Next, we fix the first light core position determined by the first step, and find the optimal solution of two cores, with the second light core position determined. Please note that although we say optimal in the second step, such solution is only the optimal solution with the first light core fixed at the position determined by the first step, but not the true optimal solution for general two light cores. Similarly, in the $(i + 1)$-th step, we look for the optimal solution for $i + 1$ light cores with the positions of i light cores found in all previous steps remain fixed. This process is shown in Fig. x. By proceeding such strategy for n steps, the optimum light core distribution that leads to maximum PPW for each light core number can be specified.

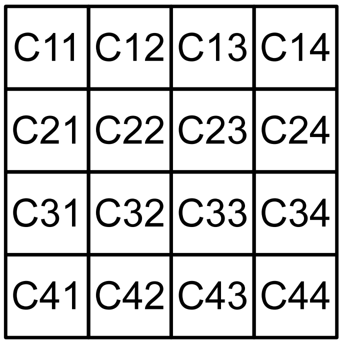


Fig. 5: Configuration of the 16-core chip.

V. Experimental Results

In this section, we evaluate both accuracy and efficiency of the proposed light core distribution estimation method.

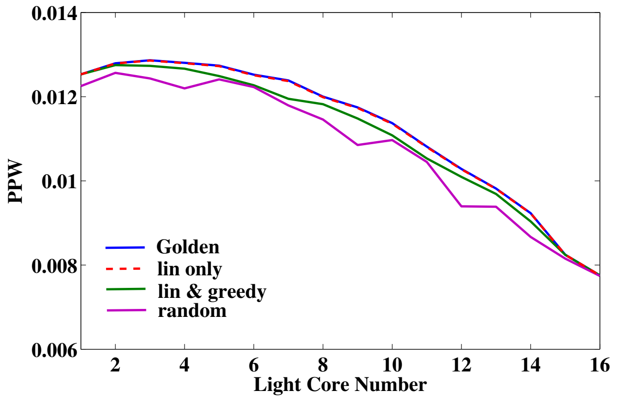
A. Experiment setup

All data are collected on a PC with Intel I5 2400 CPU and 2 GB memory. In the experiment, multi-core systems with the number of cores ranging from 9 to 64 are used, and each system is tested with different parallel computation ratios. The thermal models of these systems are extracted from HotSpot [x] with default package and chip parameters. For all test cases, we set the ambient temperature as $20\,°\mathrm{C}$, and the thermal constraint as $95\,°\mathrm{C}$. Noted, for multi-core systems with different core numbers, the power density and power supply remain constant.

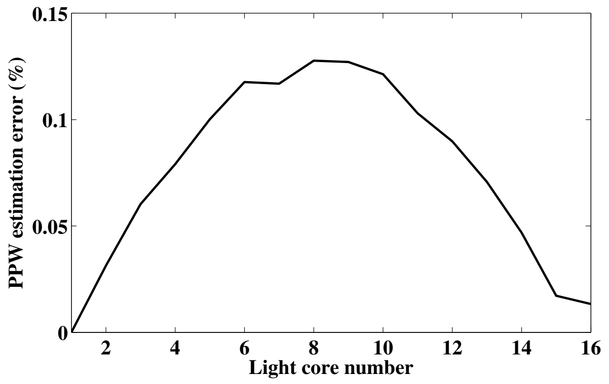
Through HSPICE simulation, the impact of temperature on device leakage can be characterized. With the collected data, we can obtain the parameters of model through curve fitting as shown in Fig.x.

For accuracy and speed comparison, we first perform the iteration based power estimation with brute force search strategy, which is accurate but very time-consuming, therefore we consider it as the golden accuracy baseline (called ”golden” for short). Then, because our complete method includes two acceleration techniques (Taylor expansion based linear thermal models, and greedy based search strategy), we use two cases, in the first case, we implement the Taylor expansion based linear thermal models only, i.e., the non-iteration based power estimation method with brute force search strategy (called ”lin-only” for short). In the second case, we further implement the greedy search strategy, i.e., the non-iteration based power estimation with greedy search strategy (called ”lin & greedy” for short). each with one more technique than the previous one, to better analyze our method.

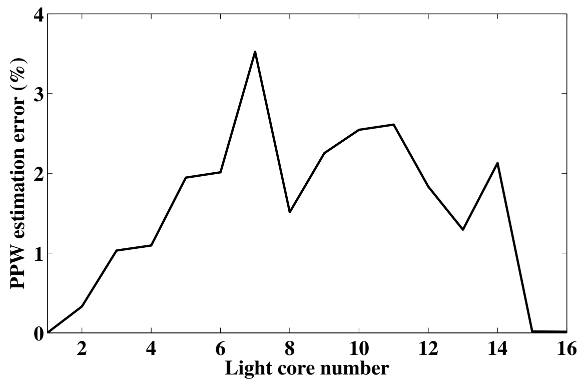
The schematic of the parallel operation is described here. The main core (the core in red) runs at all times, both in serial computation and parallel computation; while the other light cores only run during parallel segment of the code, they will be in idle state when executing the serial segment of the code. The remaining cores will always be off. The energy consumption is calculated according to this schematic.



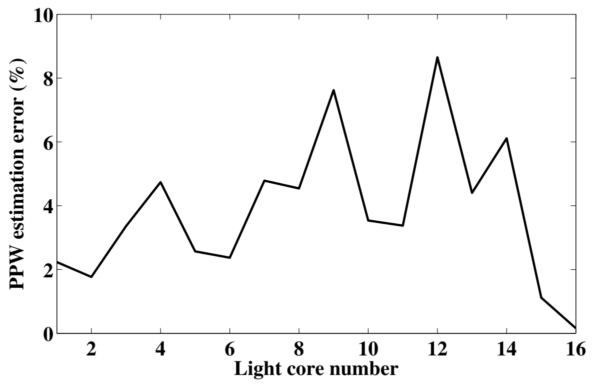
(a) The estimated PPW curves by different methods for accuracy comparison when $f = 0.9$.



(b) PPW estimation error curve of using “lin only” method.

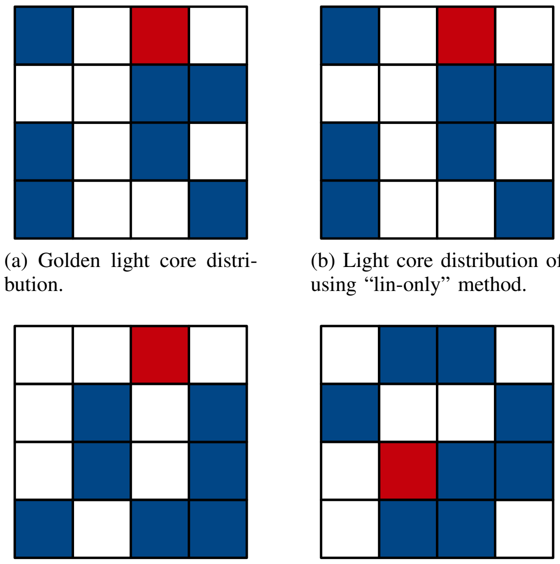


(c) PPW estimation error curve of using “lin & greedy” method.



(d) PPW estimation error curve of using “random” method.

Fig. 6: Accuracy comparison and estimation error curves of the proposed method on the 16-core chip.


(a) Golden light core distribution.

(b) Light core distribution of using “lin-only” method.

(c) Light core distribution of using “lin & greedy” method.

(d) Light core distribution of using “random” method.

Fig. 7: Light core distribution estimation results of the 16-core system when light core number is 8 and $f = 0.9$.

B. Estimation accuracy of the proposed method

We first test the accuracy of the proposed light cores distribution estimation method. Here we use the 16-core system as an example for demonstration and discussion. Results on systems with other core numbers are also collected and will be discussed later. To further demonstrate the effectiveness of the proposed light core distribution method, randomly generated light core distributions with light core number ranging from 1 to 16 are implemented (called ”random” for short).

The estimated PPW results are shown in Fig. x to analyze the accuracy of different methods. To clearly show the results, we only plot the curves representing $f = 0.9$, f is the fraction of the code that can be run in parallel. From Fig. x, we can see that the curve representing ”lin-only” almost overlaps with that of the golden result. Besides, the curve representing ”lin & greedy” are close to the golden curve, showing small estimation error. While the curve representing ”random” deviates greatly from the golden curve. Such observation means that implementing Taylor expansion based thermal models for PPW estimation is accurate, which can be verified by curve ”lin-only”. And further performing greedy based search strategy introduces small extra error as expected, but it can significantly reduce the estimation time as can be shown later. The PPW estimation error of ”lin-only” is very small, which can be shown in Fig. x. While the PPW estimation error of ”lin & greedy” is larger, meaning the greedy based search strategy can boosting the estimation speed greatly by introducing reasonable error. The PPW estimation result of ”random” deviates from that of the golden too much, which can be shown in Fig. x.

In addition to showing the PPW estimation results of different methods, we also plot the light core distribution maps of all methods when light core number is 8 in Fig. x. The cores in blue and red are active during parallel computation process, while the cores in while are off. During sequential computation process, the core in red is active, and the cores in blue are in idle state. It is clear that the light core distributions of golden and ”lin-only” are identical to each other. While the light core distribution of ”lin & greedy” differs from the golden one, and the reason is that the greedy based search strategy can only find the optimal solution based on the solution of previous step, which makes the solution sub-optimal. As the step number increases, the difference between the optimal solution and sub-optimal solution may increase. But as can be verified in our experiments, the PPW estimation of sub-optimal light core distribution only slightly degenerates than that of the optimal solution.

C. Speed and accuracy data of the proposed method

We have graphically seen from Section x that the new method has good accuracy. Now in this part, we show the speed and accuracy comparison results of the new method against the iterative method. Multi-core systems with core numbers from 9 to 64 are implemented.

Table 1 records the speed and accuracy data of the proposed light core distribution estimation method comparing with iterative method. For estimation accuracy, the ”lin-only” introduces small errors, because it is based on the linear approximation of static power at two Taylor expansion points. Errors given by ”lin & greedy” are slightly larger than those of the ”lin-only” one, the reason is that greedy based search strategy can only find the sub-optimal solution.

Then let us look at the speed comparison in Table 1. First, ”lin-only” are several times faster than ”ite” because it implements the linear thermal model to avoid the time-consuming iteration process. Then, ”lin & greedy” method can further greatly reduce estimation time. Due to the greedy based search strategy, we can magnificently speed up the estimation at the cost of a little accuracy.

For multi-core systems with core number exceeding 16, we cannot get the estimation results by implementing methods with brute force search strategy, for it consumes too much time and computation resource. In table 2, we give the speed and PPW estimation data of the proposed light core distribution estimation method.

D. Analysis

We first analyze the properties of $Perf$ of a multi-core system in dark silicon. Here we use the 49-core system as an example for demonstration and discussion. In this case, the $Perf$ corresponds to the light core distributions in which the PPW is maximum. Noted, the Fred Pollack’s performance efficiency rule is implemented, which states that, the performance of a processor that consumes N times more transistors can only improve $\sqrt{T}$ times. We set the performance of a core in the 16-core system to be 1.

As can be shown in Fig. x, the maximum $Perf$ of the 49-core system is 7.28, where $f = 1$ and $n = 31$. However, as the number of light cores increase after the critical value of 31, the $Perf$ would be worse, this is due to the performance deterioration resulting from DVFS. After certain light core

TABLE I: Accuracy and speed comparison of different light core distribution estimation methods.

core #	method	f	PPW error (%) max	PPW error (%) avg	time (s)	speedup vs ite
9	ite	1	NA	NA	1.24	NA
		0.9				
		0				
	lin-only	1	0	0	0.35	3.54
		0.9	0.01	0		
		0	0.20	0.05		
	lin & greedy	1	0.79	0.24	0.10	12.4
		0.9	0.71	0.22		
		0	0.20	0.06		
16	ite	1	NA	NA	596.32	NA
		0.9				
		0				
	lin-only	1	0	0	124.38	4.79
		0.9	0.13	0.08		
		0	0.97	0.61		
	lin & greedy	1	3.83	1.62	0.32	1863.50
		0.9	3.52	1.51		
		0	1.11	0.67		

TABLE II: Speed and PPW estimation data of the proposed ”lin & greedy” method.

core #	f	Optimal light core number	Max PPW	time (s)
25	1	7	0.0160	1.23
	0.9	2	0.0157	
	0	1	0.0151	
36	1	7	0.0191	8.63
	0.9	4	0.0184	
	0	1	0.0183	
49	1	12	0.0223	30.25
	0.9	2	0.0210	
	0	1	0.0207	
64	1	15	0.0254	85.84
	0.9	1	0.0237	
	0	1	0.0237	

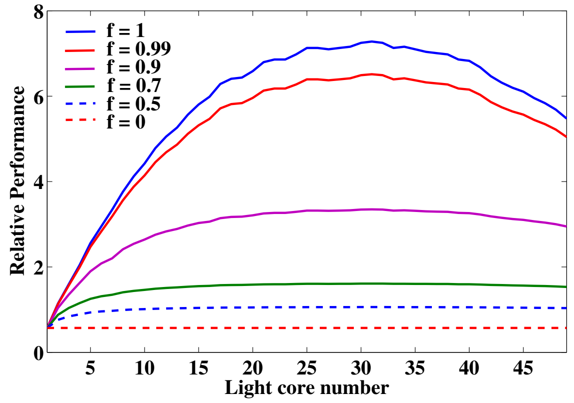


Fig. 8: The relative performance of the 49-core system in dark silicon

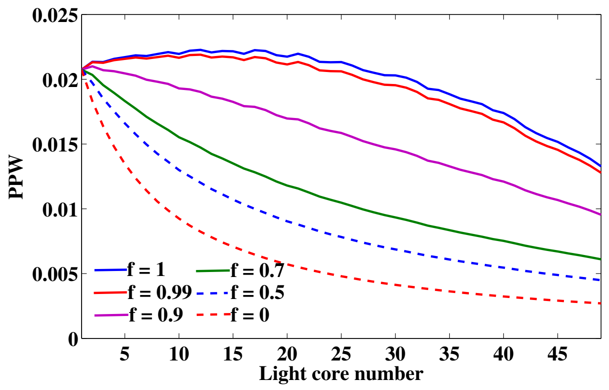


Fig. 9: The estimation of the PPW of the 49-core system in dark silicon

number, the $Perf$ introduced by new light core would be less than the $Perf$ deterioration it causes to former light cores.

Next, we shall discuss the properties of PPW of a multi-core system in dark silicon. Here, we also use the 49-core system as an example for demonstration and discussion.

Fig. x shows the PPW of a 49-core system in dark silicon. We can see that for the cases where f is relatively high, which means a large part of the code can be run in parallel, a optimal light core number n which is higher than 1 for maximum PPW exists. After this critical value of n, the PPW will decrease. The reason is that the relative performance of low fraction of parallel computation saturates much more easily, and increasing light core number after performance saturates will only drag down the energy efficiency.

It is also clear that the PPW of higher fraction of parallel computation decrease faster than that of lower fraction of parallel computation, the reason is that for sequential computation, increasing light core number only introduces an idle core, the performance degenerates only a little, while for parallel computation, the overall performance shall degenerate much more after certain light core number.

We can also see that higher f means overall higher PPW, this is because higher fraction of parallel computation can provide more computing tasks to run on cores in parallel.