Dear Editor:

We would like to submit the enclosed manuscript entitled “Dynamic Thermal Management Based on Model Predictive Control with Task Migration and DVFS ”, which we wish to be considered for publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
Some preliminary results of this paper is going to appear in *2014 IEEE Asia Pacific Conference on Circuits and Systems* [1], which is attached in this submission. This submission has the following significant changes compared with the conference paper:

1) We have expanded the 4-page conference paper into this over 8-page article with more discussions on the backgrounds and related works, new figures for better illustration of ideas, detailed explanation of the new algorithm, more examples for better demonstration, new comparison with existing method, and newly collected experiment results on many new CPUs with different number of cores.
2) We have added introduction on dynamic thermal management in Section II-B for completeness of this article.
3) We have added more detailed discussions and more steps of model predictive control in Section III-A.
4) Presentation of the new algorithm has been revised for clearer presentation. Two problem cases encountered after weighted bipartite matching has been classified and discussed in details in Section III-C.
5) The whole algorithm flow is given as pseudo-code in Algorithm 1.
6) A detailed example has been added in Section III-C to illustrate the processing steps of the second problem case with a new figure Fig. 3 added.
7) We have updated the experimental results with additional comparison against an existing MPC based DTM method. Also, more experimental results are reported on many new CPUs with different number of cores. CPU performance comparison results and algorithm runtime are reported in this article.

Thank you very much for your kind attention.

Sincerely,
Hai Wang

Dynamic Thermal Management Based on Model Predictive Control with Task Migration and DVFS

Hai Wang*, Jian Ma*, Sheldon X.-D. Tan†, Chi Zhang*, and He Tang*

***Abstract*—Dynamic thermal management methods are important to control the temperature of microprocessor at runtime and improve the reliability performance of the chip. In this article, a model predictive control (MPC) based dynamic thermal management method is proposed with hybrid thermal regulation techniques combining task migration and dynamic voltage frequency scaling (DVFS) methods. The new method is designed to reduce processor performance degradation, lower temperature variation among different cores, while tracking the specified safe temperature ceiling. Experimental results on several CPUs with different number of cores have shown the proposed method can successfully track the given temperature ceiling, and is able to enhance CPU performance by up to 4%, lower temperature variation by up to 50% compared to MPC assisted DVFS method.**

I. Introduction

High temperature caused by high power density of the chip has become one of the most important constraints for the evolution of microprocessor: the high temperature not only lowers the reliability of the chip, and also limits the performance of the processor [2].

Dynamic thermal management (DTM) techniques are introduced to mitigate the thermal induced reliability issues as well as enhance the performance of the processor under safe temperatures [3]. Effective DTM methods include Task Migration and Dynamic Voltage and Frequency Scaling (DVFS) techniques. Task migration method [4], [5], [6], [7], [8], [9] migrates heavy load (high power consumption) tasks from high temperature cores with light load (low power consumption) tasks from low temperature cores, and as a result, minimizes the temperature variation among cores. It is able to keep the high performance of the processor, reduce spatial thermal gradient, and also lower the peak temperature, but it cannot guarantee the core temperatures to be within the safe temperature range. Instead, DVFS method [10], [11], [12] keeps the core temperature in safe range through reducing the power consumption by means of lowering the voltage and frequency of core/cores. It is able to guarantee the temperature safety, but with the penalty of significant performance drop. There are also hybrid methods proposed which usually combines DVFS and task migration methods for more effective control, including experience based method [13], hybrid method which optimizes performance [14], and hybrid method which optimizes energy consumption [15].

Model predictive control (MPC) [16] has been introduced in control community to track a target output in the future by computing the desired input trajectory. In [17], [18], [19], MPC has been introduced for microprocessor DTM problems and used with DVFS method which takes the frequencies of cores as the control input. However, due to the limitation of DVFS introduced before, such algorithms have huge impact on the performance of the processor. Power consumption of a core is not only related to the frequency, but also associated with the application running on the core: with the same frequency, some applications consume much less power than others. As a result, in MPC assisted DVFS, there are cores which run at maximum frequency but still have low power consumptions, and probably have temperatures way below the safe line, while at the same time, there are also cores whose frequencies have been greatly reduced in order to match the temperature targets on them.

In this paper, we propose a new DTM method: hybrid dynamic thermal management method with model predictive control. The simplified flow of the new method is shown in Fig. 1. The new method utilizes model predictive control to compute the desired power input distribution (desired power map as shown in the figure) for a user specified temperature target (desired thermal map), then performs task migration and DVFS to adjust current power distribution (current power map) to match the desired one (desired power map). With the guidance of the weighted bipartite matching method, voltages and frequencies of only limited number of cores are scaled to ensure the CPU is running at its maximum speed without violating the thermal constraint. The hybrid method takes advantages from model predictive control, task migration and DVFS techniques: it is able to maximize the performance of the processor, minimize the temperature variations over cores, and also track target temperatures. The rest of this article is organized as follows: Section II introduces preliminary knowledges related to the new method, including basics of thermal modeling and dynamic thermal management; in Section III, the new DTM method based on MPC, DVFS, and Task migration is presented in details; experimental results are given in Section IV, with comparison with existing MPC based algorithm; finally, conclusions are drawn in Section V.

Hai Wang, Jian Ma, Chi Zhang, and He Tang are with State Key Laboratory of Electronic Thin Films and Integrated Devices, University of Electronic Science and Technology of China, Chengdu, 610054 China, and School of Microelectronics and Solid-State Electronics, University of Electronic Science and Technology of China, Chengdu, 610054 China. Sheldon X.-D. Tan is with Department of Electrical Engineering, University of California, Riverside, CA 92521 USA. Please address comments to Hai Wang (e-mail: wanghai@uestc.edu.cn).

This research is supported in part by National Science Foundation of China grant under No. 61404024, in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, in part by the Open Foundation of State Key Laboratory of Electronic Thin Films and Integrated Devices (KFJJ201409), and in part by an initial startup grant from UESTC.

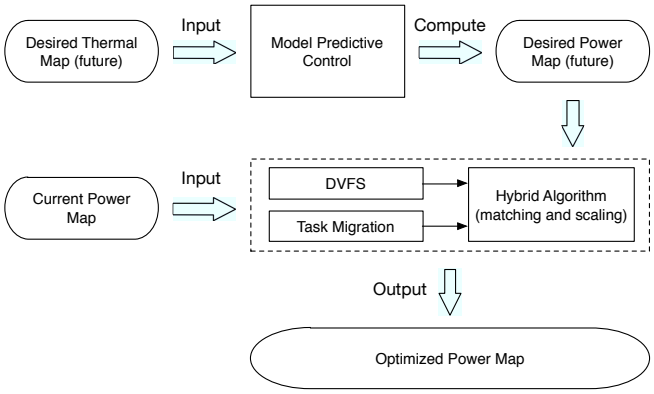


Fig. 1. Simplified flow of the new DTM method.

II. Basics of Thermal Modeling and Dynamic Thermal Management

In this section, the basics of thermal modeling techniques are presented first, followed with the introduction of the dynamic thermal management method.

A. Thermal modeling

A thermal model of a microprocessor with n_c cores can be expressed as an ordinary differential equation [20], and then written into discretized form by applying Euler's method with fixed time step as

$$\begin{aligned} t_m(k+1) &= A_m t_m(k) + B_m p(k), \\ y(k) &= C_m t_m(k), \end{aligned} \tag{1}$$

where $t_m(k) \in \mathbb{R}^n$ is the thermal vector representing temperatures of n blocks of the processor, which includes n_c cores (with $n_c < n$) and also boundary condition nodes; $A_m \in \mathbb{R}^{n\times n}$ and $B_m \in \mathbb{R}^{n\times n_c}$ contains thermal dynamic information determined by the thermal conductance, thermal capacitance, and the topology of the processor; $p(k) \in \mathbb{R}^{n_c}$ is the power vector of n_c cores at time k, and is the input of the model; $y(k)$ is the thermal vector of n_c cores, and is the output of the model; $C_m \in \mathbb{R}^{n_c\times n}$ is the output selection matrix, which selects the n_c core temperatures from $t_m(k)$.

B. Dynamic thermal management

It is well known that high temperature and large temperature distribution variation induces many reliability issues such as material melt down, thermal induced stress, Electro Migration (EM), etc. Dynamic thermal management methods are introduced to optimize the on-chip temperature such that the thermal-induced reliability performance can be improved. Dynamic thermal management mainly focus on two aspects of temperature behavior: first, the peak on-chip temperatures, i.e., hot spots, second, the temperature distribution, i.e., the spacial thermal gradient.

Among many dynamic thermal management methods, two most widely used and effective ones are introduced here: task migration based method and DVFS based method. As briefly introduced in Section I, task migration operates on multi/many-core systems. It monitors the temperatures of different cores (usually through temperature sensors or soft thermal sensors [21]), and re-distribute the loads of cores by moving the heavy load tasks from the high temperature cores to low temperature cores. For example, in [3], hotspot imbalance is defined for every core which reveals the thermal gradient on the core. Cores with most critical hotspot imbalance are considered first for migration, and task which is most able to reduce the heating of the core's critical hotspot will be chosen to migrate to this core. However, on migration decision making (i.e., which tasks should migrate, and where should they migrate to), most existing methods are experience based as shown above. Although they can be effective, there is no theoretical support to guarantee that the tasks after migration lead to safe temperatures for all cores.

DVFS can be performed on both single-core and multi/many-core systems. It reduces the dynamic and static power, through lowering the supply voltage levels. Since supply voltage levels are lowered, operation frequencies should be lowered at the same time in order to satisfy the timing constraints. Please note that DVFS harms the performance of the system since operation frequencies of cores are lowered. Although many DVFS algorithms are heuristic based, there are theoretical based DVFS algorithms proposed which uses convex optimization [12] or model predictive control (MPC) method [17], [18] to calculate the DVFS levels which lead to safe temperatures. Nevertheless, these methods usually over sacrifice system performance because there may exist low temperature cores with light tasks even without DVFS, and light tasks on such cores could have been switched with heavy tasks on high temperature cores to avoid unnecessary DVFS. As a result, the system with such DVFS method is not always running at the maximum speed as it could.

There are also other dynamic thermal management methods, for example power gating and clock gating. Power gating [22] can be viewed as the extreme case of DVFS, which sets both voltage and frequency to be zero and as a result the gated block consumes neither dynamic power nor static power. On the other hand, clock gating [13] blocks the clock signal through setting the enable condition of registers to false, such that reduces the dynamic power of the gated block. Although clock gating has lower system overhead and is easier to implement compared to power gating, it cannot reduce static power, which is irrelevant to the clock.

III. Hybrid DTM method with MPC

In this section, we will demonstrate the whole flow of the new method. First, we show that MPC method can be used to compute the desired power map given the desired temperature constraints. Next, we show how to adjust the power input to match the desired power map computed by MPC, through introducing a hybrid method combining task migration and DVFS.

A. Compute desired power map using MPC

For the completeness of this article, model predictive control method is briefly introduced focusing on thermal application. Interested readers are referred to [16] for detailed MPC discussion.

In order to maximize the performance of the processor without jeopardizing the reliability of the chip, we want every core of the chip to be running with the load *as heavy as possible* resulting in a temperature *just at the ceiling* of the safe temperature range. In traditional DVFS and task migration based works, temperatures of the cores are used to guide the voltage/frequency scaling and task migration, where the performances of the hot cores are lowered to make it thermally safe. However, these methods have one major drawback: due to the lack of theoretical support on the power side, DTM methods based on pure thermal information may lead to temporary overheated cores and/or over cooled cores. Information of powers which lead to the target temperature is highly demanded as a guidence for DTM. In this work, MPC method is used to compute the desired power information.

The basic idea of MPC method is to calculate the appropriate input powers to control the output core temperatures, as a result, the connection between the input power difference and output core temperatures needs to be first formulated. In order to do that, we define the difference of the state and input variables as

$$\begin{aligned} \Delta t_m(k) &= t_m(k) - t_m(k-1), \\ \Delta p(k) &= p(k) - p(k-1). \end{aligned} \tag{2}$$

Taking the difference of adjacent two steps of (1), there is

$$\begin{aligned} \Delta t_m(k+1) &= A_m \Delta t_m(k) + B_m \Delta p(k), \\ y(k+1) - y(k) &= C_m A_m \Delta t_m(k) + C_m B_m \Delta p(k). \end{aligned} \tag{3}$$

Introducing a new variable

$$t(k) = \begin{bmatrix} \Delta t_m(k) \\ y(k) \end{bmatrix},$$

we can rewrite (3) into the following augmented model

$$\begin{aligned} t(k+1) &= At(k) + B\Delta p(k), \\ y(k) &= Ct(k), \end{aligned} \tag{4}$$

where

$$A = \begin{bmatrix} A_m & 0_m \\ C_m A_m & I \end{bmatrix}, \qquad B = \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix},$$
$$C = \begin{bmatrix} 0_m & I \end{bmatrix}, \qquad t(k) = \begin{bmatrix} \Delta t_m(k) \\ y(k) \end{bmatrix},$$

with 0_m as a matrix with all zero elements with suitable size.

So far, we have obtained the connection between the power input difference (control source) and the output core temperatures in (4). Next, the power input difference needs to be determined given the desired ceiling safe temperatures of cores. Assume the ceiling safe temperatures of cores over several time steps into the future are given, and written in a vector form as

$$G = [g^T, g^T, ..., g^T]^T \in \mathbb{R}^{mN_p \times 1}.$$

In this vector, $g \in \mathbb{R}^{m \times 1}$ contains the ceiling safe temperatures of each core,[1] N_p stands for a time frame from current to the N_p steps into the future, and is called the prediction horizon.

[1]Here we assume the ceiling safe temperature does not change over time, which usually fits the real world situation. Please note that this is not a limitation of the new method.

In order to keep the core temperatures tracking the ceiling safe temperature in the prediction horizon, at a time k, the future control trajectory (which is actually unknown and needs to be computed in the end) is introduced as

$$\Delta P = [\Delta p(k), \Delta p(k+1), \ldots, \Delta p(k+N_c-1)]^T,$$

where N_c is called the control horizon. The prediction of core temperatures

$$Y = [y(k+1|k)^T, y(k+2|k)^T, \ldots, y(k+N_p|k)^T]^T,$$

where $y(k+j|k)$ is the predicted core temperatures at time $k+j$ using information of current time k, can be calculated assuming ΔP is known, using

$$Y = Vt(k) + \Phi \Delta P, \tag{5}$$

where V and Φ are shown in (6) on top of the next page.

In order to minimize the difference between the core temperatures and the desired ceiling safe temperatures, we can formulate the cost function F as

$$F = (G-Y)^T(G-Y) + \Delta P^T R \Delta P, \tag{7}$$

where $R = rI_{N_c \times N_c}$ is tuning matrix with r as the tuning parameter. Please also note that Y is a function of the unknown variable ΔP. The first term of (7) reveals the difference between the output core temperatures and the desired temperatures. The second term indicates the size of the input power change, i.e., we prefer small change in input power. The value of r determines the weight between the two terms, and can be fine tuned through experiment according to practical situations, including the number of cores, different running applications (different power traces), etc.

Next, optimization is performed to minimize (7) by taking the first derivative of (7) with respect to ΔP and making it equal to zero. The solution of ΔP is

$$\Delta P = (\Phi^T \Phi + R)^{-1} \Phi^T (G - Vx(k)). \tag{8}$$

At each MPC time k, we only use the first computed control signal $\Delta p(k)$ from (8) and update the power distribution as

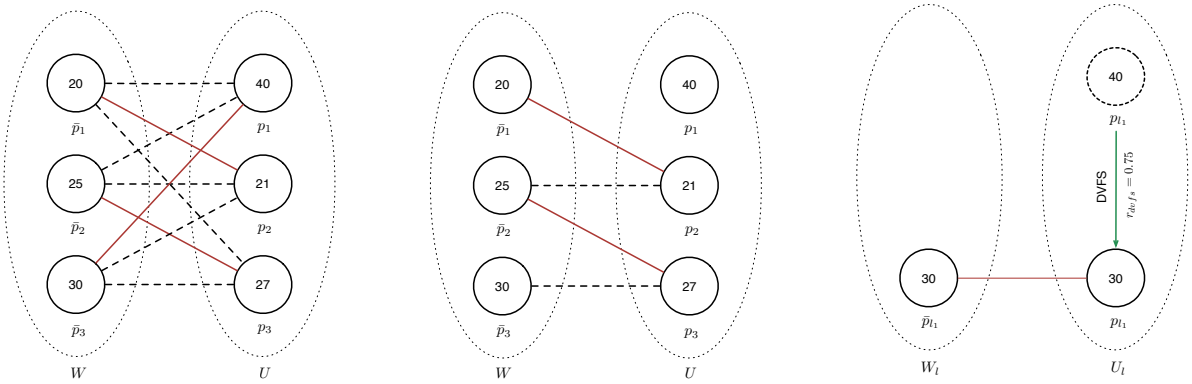
$$\bar{p}(k) \leftarrow p(k) + \Delta p(k), \tag{9}$$

where $\bar{p}(k)$ is the updated power distribution. The resulting temperature $y(k)$ would track the desired temperature ceiling with the updated power input. In other words, the updated power input is the highest power can be reached without violating the temperature requirements.

B. Task migration with weighted bipartite matching method

Since we are able to obtain the desired power map using MPC, the DTM problems boils down to matching powers of current tasks with the desired power map. We first consider task migration method only. Task migration method redistribute tasks among processor cores in order to reduce the temperature variation among cores. It usually involves swapping heavy load tasks from high temperature cores with light load tasks from low temperature cores. However, the purpose of using task migration method in this work is a little

$$V = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix}, \Phi = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \cdots & CA^{N_p-N_c}B \end{bmatrix} \tag{6}$$



(a) Pure task migration case.

(b) Hybrid case showing the weighted bipartite matching with the threshold of 5.

(c) Hybrid case showing DVFS process after (b) and then re-perform bipartite matching.

Fig. 2. Example of pure task migration (a) and hybrid method (b) (c) with weighted bipartite matching algorithm. The number in the vertex stands for the power value of the core. The solid red edges are the output edges connecting the matched vertex pairs, the dashed edges mean the corresponding two vertices are not matched. Weights of edges are not shown in the figure for simplicity.

different: we want to simply redistributes tasks (powers) as the "update" action in (9).

First, the current power distribution of all m cores is represented as $p(k) = [p_1, p_2, \cdots, p_m]$, the desired power distribution after task migration, i.e. the updated power distribution from MPC, is denoted as $\bar{p}(k) = [\bar{p}_1, \bar{p}_2, \cdots, \bar{p}_m]$. To perform the "update" action using task migration, we only need to determine how to re-arrange the elements in $p(k)$ to match the corresponding elements in $\bar{p}$. This is an assignment problem, and can be formulated into a weighted bipartite matching problem. A weighted complete bipartite graph $\mathcal{G} = (U, W, E)$ is first built, with vertex sets $U = \{p_1, p_2, \cdots, p_m\}$, $W = \{\bar{p}_1, \bar{p}_2, \cdots, \bar{p}_m\}$, and edge set E containing connections between U and W with weights. As a complete bipartite graph, every vertex in U is connected with *all* vertices in W. The weight between p_i and $\bar{p}_j$ is defined as the

$$w_{ij} = |p_i - \bar{p}_j|. \tag{10}$$

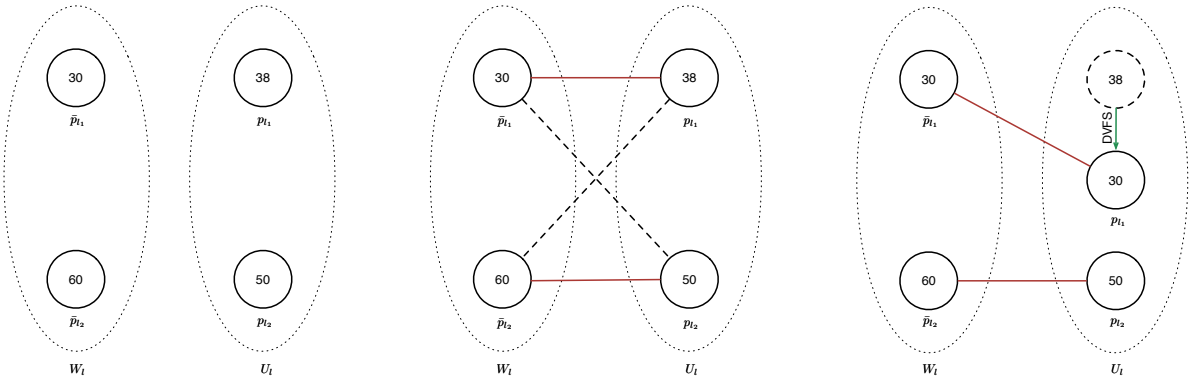
Weighted bipartite matching based on Hungarian algorithm [23], [24] is performed on the graph, and the output of the algorithm is another bipartite graph with the same vertex sets U and V but every vertex in U (or W) is connected with only one vertex in W (or U) such that the cost (total weight of the output edges) is minimized.

An example of pure task migration method is provided in Fig. 2 (a). All the vertices are paired as expected, with the total cost of 13. It is also obvious that the pair $(p_3, \bar{p}_1)$, which contributes 10 in the total cost, is a worse matched pair than the other two matched pairs. This badly matched pair will lead to temperature violation after the corresponding task migration, since the actually power (40) is much higher than the desired power (30). This problem will be solved in the following subsection by introducing a hybrid method with DVFS and task migration.

C. Hybrid method with DVFS and task migration

Sometimes, pure task migration with MPC is able to enhance the performance of the processor. However, two problem cases are associated with pure task migration: first, if there are much more heavy load tasks than light load ones, the resulting temperature will always be higher than the ceiling temperature, such as caused by the $(p_3, \bar{p}_1)$ pair in Fig. 2 (a); second, even if there are somewhat equal number of heavy and light load tasks, sometimes it is still difficult for the resulting temperature to track the ceiling temperature, because many times a good match (nearly zero cost) in bipartite matching algorithm does not exist. The two problems are solved next by integrating DVFS and introducing a hybrid method.

In this subsection, DVFS method is introduced to assist the bipartite matching algorithm in order to find the nearly perfect match, even when there are mostly heavy load tasks or the perfect match does not exist. In pure task migration case, all the matched pairs are found directly, which include some bad matched pairs due to the limitation of pure task migration method stated before. Instead, we first find the good pairs *only*, by removing the edges with weights larger than a threshold w_{th} from $\mathcal{G}$, and run bipartite matching algorithm on the modified graph $\tilde{\mathcal{G}} = (U, W, \tilde{E})$. After the



(a) Collect the leftovers in U_l and W_l. Current threshold is 5.

(b) Do bipartite matching with threshold relaxed to infinity.

(c) Perform DVFS on the pairs with $p_{l_i} > \bar{p}_{l_j}$.

Fig. 3. Example of leftover case two. The threshold is raised from 5 to infinity, bipartite matching is performed on the graph with new threshold, finally DVFS is performed where necessary.

first match, assume there are q leftover vertices from U (same number from W). The leftovers from U and W are collected in $U_l = \{p_{l_1}, p_{l_2}, \cdots, p_{l_q}\}$ and $W_l = \{\bar{p}_{l_1}, \bar{p}_{l_2}, \cdots, \bar{p}_{l_q}\}$, respectively. Then, we need to identify which problem case (out of the two cases introduced at the beginning of this subsection) we have encountered and the solution is different for each case.

Case one: assume average power from U_l and W_l are $avg(p_l)$ and $avg(\bar{p}_l)$, if there is $avg(p_l) > avg(\bar{p}_l)$, then we have encountered the first problem case (many heavy load tasks). This problem is solved by performing DVFS on the corresponding cores of U_l with the power scaling ratio of

$$r_{dvfs} = avg(\bar{p}_l)/avg(p_l), \tag{11}$$

and perform bipartite matching again on the scaled leftovers. It is possible that there are still leftovers after this round of DVFS and matching. The leftovers should be processed iteratively, i.e., they should be collected and the problem case needs to be re-determined and treated until there is no remaining leftovers.

Case two: it is also possible that there is $avg(p_l) < avg(\bar{p}_l)$ which means the first round leftovers are caused by the second case problem (nearly equal number of heavy and light load tasks but still cannot be matched). To solve this problem, the threshold w_{th} is relaxed to infinity in order to pair all vertices. We further classify the poorly matched pairs into two types. For pair $\{p_{l_i}, \bar{p}_{l_j}\}$, if there is $p_{l_i} > \bar{p}_{l_j}$, DVFS is performed on the current power p_{l_i} with scaling ratio

$$r_{dvfs} = \bar{p}_{l_j}/p_{l_i}. \tag{12}$$

If $p_{l_i} > \bar{p}_{l_j}$, because current power is even lower than the desired power, then we can leave the current power unchanged and still get the resulting temperature below threshold.

The whole flow of the hybrid algorithm is summarized in Algorithm 1.

An example of the new hybrid method is shown in Fig. 2 (b) and (c). The modified weighted bipartite graph with the weight threshold 5 is shown in Fig. 2 (b). The edges in Fig. 2 (a) which have weights larger than 5 are removed in this new figure. The resulting matched pairs are only $(p_1, \bar{p}_2)$, and $(p_2, \bar{p}_3)$, with total cost of 3. p_3 and $\bar{p}_1$, which cannot be matched well with any other vertices, are leftovers of the first round. They are collected into the leftover group U_l and W_l and re-numbered as p_{l_1} and $\bar{p}_{l_1}$ as shown in Fig. 2 (c). Because there is $avg(p_l) > avg(\bar{p}_l)$, we have encountered the first leftover case, then DVFS is performed on $\bar{p}_{l_1}$ with the scaling ratio $r_{dvfs} = 0.75$ calculated using (11). Finally, p_{l_1}

Algorithm 1 Efficient dynamic thermal management algorithm based on MPC with DVFS and task migration

```
Calculate the desired power distribution p̄(k) using MPC equations (8) and (9).
while U ∉ φ do
    Build bipartite graph G̃ = (U, W, Ẽ) using p(k), p̄(k), and threshold w_th.
    Do bipartite matching on G̃, output matched pairs, and collect un-matched pairs in U_l and W_l.
    if avg(p_l) > avg(p̄_l) then
        Perform DVFS and scale elements in U_l with ratio in (11).
        Build bipartite graph with W_l, the scaled U_l, and w_th, and perform bipartite matching on it.
        Output the matched pairs and the scaling ratio.
    else
        Relax w_th to infinity (or to a higher value)
        Build bipartite graph with U_l, W_l, and new w_th, and perform bipartite matching on it.
        for ∀(i, j) do
            if p_{l_i} > p̄_{l_j} then
                Perform DVFS to scale p_{l_i} with ratio in (12).
            end if
        end for
        Output the matched pairs and the scaling ratios.
    end if
    Update U, W using the leftovers.
end while
```

C11	C12	C13
C21	C22	C23
C31	C32	C33

Fig. 4. The configuration of the 3×3 microprocessor.

and the scaled $\bar{p}_{l_1}$ are matched with the second round weighted bipartite matching. In this example, there is no more leftovers and this ends the algorithm.

We provide another example in Fig. 3 to show the other leftover case. Assume the leftovers from the previous iteration are collected in Fig. 3 (a), and the current threshold is $w_{th} = 5$. Since $avg(p_l) < avg(\bar{p}_l)$, it is obvious that we have encountered the second leftover case. Then, the threshold w_{th} is relaxed to infinity, and the subsequent bipartite matching result is shown in Fig. 3 (b). Last, matched pairs from the previous bipartite matching are examined, and DVFS is performed on the pairs $\{p_{l_i}, \bar{p}_{l_j}\}$ with $p_{l_i} > \bar{p}_{l_j}$. In this example, DVFS is performed only on the pair $\{p_{l_1}, \bar{p}_{l_1}\}$, with the scaling ratio 0.79. The other pair $\{p_{l_2}, \bar{p}_{l_2}\}$ can be left untouched because $p_{l_2} < \bar{p}_{l2}$ means it will not cause any temperature emergency problems.

IV. Experimental Results

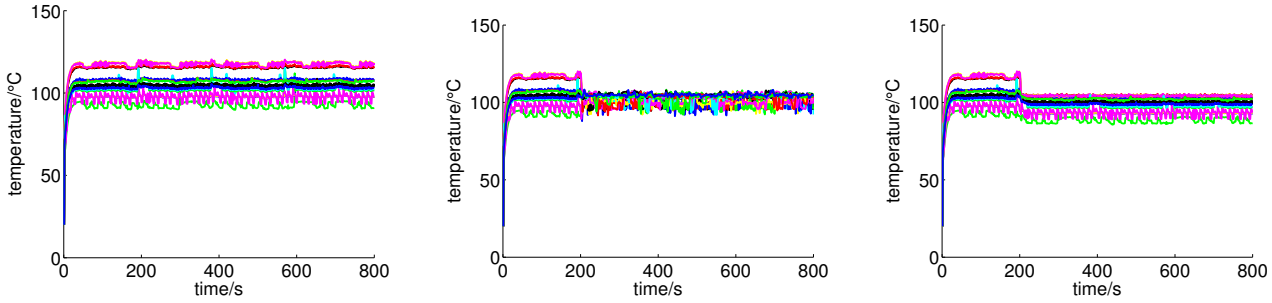
The experiments are performed on a Linux server with two 2.90GHz 8-core 16-thread CPUs and 64GB memory. The new method is implemented using MATLAB R2010a, and HotSpot [25] is used to build the thermal model based on microprocessors with 8 different core configurations, from 9 cores distributed in a 3×3 fashion to 100 cores distributed in a 10×10 fashion. The configuration of the 3×3 microprocessor is shown in Fig. 4 as an example, all other microprocessors have similar structures. The size of all chips is $9mm \times 9mm \times 0.15mm$. Wattch [26] is used to generate the power and instruction information with SPEC benchmarks [27]. We use 9 power traces on 9 different SPEC benchmarks as the power traces of the 3×3 microprocessor. For the power traces of other cases, we recycle those 9 power traces to get 16 power traces, 25 power traces and so on. The instruction traces correspond to power traces. The DVFS and task migration activating time step is set to be different case: 1, 10, 20, 40 second, and the ambient temperature is $20°C$. For comparison, we have implemented another MPC based DTM method proposed in [17]. Different from our method which utilizes both DVFS and task migration techniques, DTM in [17] employs DVFS method only.

First, we let CPUs with different configurations run at maximum speed without any DTM methods involved. This "free run" transient temperature traces of the CPU with 4×4 core configuration are shown in Fig. 5 (a). It can be seem that the cores have temperatures range from $90°C$ to $120°C$, which puts the chip in great danger. Moreover, temperature variance among cores of the 4×4 microprocessor is shown in Fig. 6 (a), which reveals the large temperature differences among cores.

Next, we test our new method by setting the safe temperature ceiling as $105°C$ for all cores, and activate our hybrid DTM with MPC at the time of 200 second. The corresponding transient temperature traces of the 4×4 CPU are given in Fig. 5 (b). It's in the case that the time step is $1s$. It can be seem that temperatures from all cores track the given ceiling of $105°C$ as soon as the new DTM method takes effect at 200 second. It is also noticed that there are temperature spikes which temporarily violate the "temperature ceiling". These spikes are caused by the fast changing temperature responses inside the DTM action window (between two DTM actions), thus they cannot be avoided since they are irrelevant to the DTM method (which means they exist in any DTM methods with the same width of action window). In order to guarantee the safety of the chip, we can provide the DTM method with the "temperature ceiling" slightly lower than the real safe temperature line based on experiments. In order to study the temperature variation among cores with the new DTM method, the corresponding temperature variance is shown in Fig. 6 (b). It is obvious that the temperature variance among cores has been greatly reduced using the new method.

Then, we test DTM in [17]. Similarly, we activate this DTM at the time of 200 second, and compare the results with those of our method. We set the same safe temperature ceiling as $105°C$ for all cores. The corresponding transient temperature traces of the 4×4 CPU are given in Fig. 5 (c), and the corresponding temperature variance is demonstrated in Fig. 6 (c). From Fig. 5 (c), it can be seem that this DTM method is actually very effective, which keeps the temperatures of all cores below the temperature ceiling (except for the spikes). However, comparing Fig. 6 (b) and (c), it is obvious that DTM in [17] leads to larger temperature variation among cores than our new method. This is because [17] only uses DVFS, and there may exist cores which are originally below temperature ceiling without voltage-frequency scaling. In this case, [17] will do nothing on these cores, and resulting in some cores' temperatures to be below the temperature ceiling. This means the CPU is not running with its best performance.

We have already shown the performances of the two DTM methods on 4×4 CPU in details, and we also compared the new method with DTM method in [17] on other CPUs with different number of cores and different time step of the method activating. Since the size of the core scales as the core number raises (we assume all chips have the same size) which leads to unrealistic power density (and extremely high temperatures), we scale the power traces coordinately to ensure that all CPUs have similar power density. And for CPUs with different core number, the threshold w_{th} in the task migration process and r in MPC are manually tuned to ensure the temperature tracking quality of the new algorithm. All these parameters are shown in Table I. The variance comparison results on all CPUs are summarized in Table II, where var_o is the average variance in "free run" state, var_d is average variance of transient temperature traces with DTM in [17], and var_n is the average variance of transient temperature traces

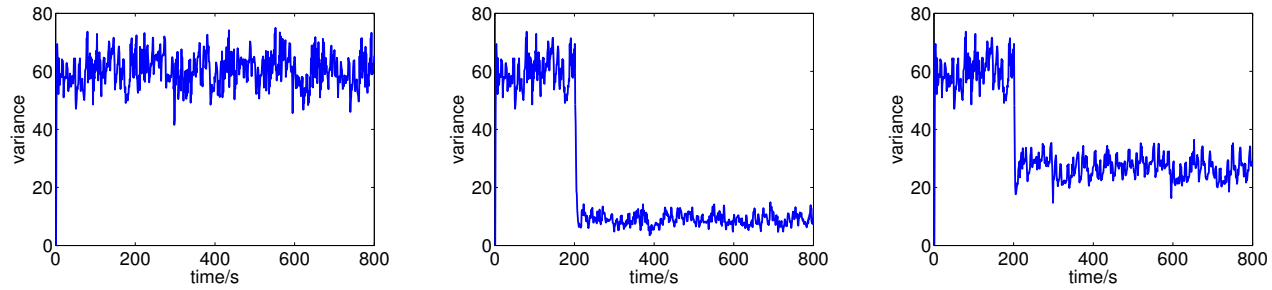


(a) Temperature traces in "free run" without any dynamic thermal management methods.

(b) Temperature traces with the new DTM method activated at the 200 second.

(c) Temperature traces with DTM method in [17] activated at the 200 second.

Fig. 5. Transient temperature traces of the 4×4 CPU with different DTM methods. Lines with different colors represents temperature traces of different cores.



(a) The temperature variance of the "free run".

(b) The temperature variance with the new DTM method activated at the 200 second.

(c) The temperature variance with DTM method in [17] activated at the 200 second.

Fig. 6. Transient variances among 16 cores in the 4×4 CPU with different DTM methods.

with our new method. Form this table, it is obvious that the temperature variance among cores has been greatly reduced using both DTM methods, and our new method leads to even significantly smaller temperature variance than DTM method in [17] for all CPUs: the variance can be reduced over 50% (as in 7×7, 8×8, 9×9, and 10×10 CPUs) compared to that of DTM method in [17]. The smaller temperature variance among cores leads to better reliability performance of the chip. But as the increase of time step, the variance become large. We should not set the time setp too large.

TABLE I
PARAMETERS OF THE NEW DTM METHOD FOR CPUS WITH DIFFERENT CORE CONFIGURATION.

Configuration	Scale	w_{th}	r
3×3	2.1	1.0	20
4×4	1.5	1.0	20
5×5	0.85	0.4	50
6×6	0.58	0.1	50
7×7	0.48	0.1	50
8×8	0.32	0.05	200
9×9	0.25	0.01	500
10×10	0.21	0.01	500

Besides reliability considerations, CPU computing performance is equally important. So the CPU performance of the two methods is also compared. CPU performance can be measured as its throughput, i.e., amount of tasks completed in unit time. In this work, we estimate CPU performance by measuring the average number of instructions per second of the core as an indicator of the throughput, i.e., higher IPS indicates better performance as long as the CPU is safe in temperature. The average IPS of the core using the new DTM method and the DTM method in [17] are collected in Table III. The safe temperature ceilings of both DTM methods are set to be $105°C$. In the table, $MIPS_o$ represents the average number of instructions per second of the core without any DTM method, i.e., the "free run" state, $MIPS_d$ stands for the average number of instructions per second of the core using DTM method in [17], and $MIPS_n$ denotes the average number of instructions per second of the core with our new DTM method. The improvement in performance of our new method against the DTM method in [17] is calculated as $(MIPS_n - MIPS_d)/MIPS_d$. It can be seem that our new method surpasses DTM method in [17] in CPU performance for all CPUs, and the improvement can be up to around 3%.

Finally, we measure the computational time of our new method. We measured the runtime needed by the new method for 1 seconds of DTM. Since our method is mainly composed of MPC and bipartite matching, we have splited the total

TABLE II
THE TEMPERATURE VARIANCE AMONG CORES IN CPUS WITH DIFFERENT CORE CONFIGURATION.

Configuration	var_o	time step 1s		time step 10s		time step 20s		time step 40s	
		var_d	var_n	var_d	var_n	var_d	var_n	var_d	var_n
3×3	86.1	50.1	30.0	49.2	34.0	49.4	35.8	49.9	43.2
4×4	60.6	27.0	9.2	25.5	11.1	25.5	11.5	25.3	13.2
5×5	79.0	29.8	15.5	31.2	12.5	31.1	12.8	30.9	16.1
6×6	66.0	27.7	11.4	26.8	11.5	25.5	12.9	25.6	12.8
7×7	60.4	25.7	6.4	23.1	6.7	22.9	6.8	23.2	7.8
8×8	69.7	30.4	9.2	28.4	9.4	28.2	9.5	28.2	10.8
9×9	75.1	35.0	9.5	32.7	10.1	32.5	11.5	32.6	14.1
10×10	71.2	26.7	6.4	25.1	7.6	24.9	8.2	24.9	10.1

TABLE III
THE AVERAGE NUMBER OF INSTRUCTIONS PER SECOND OF ONE CORE ON CPUS WITH DIFFERENT DISTRIBUTION.

Configuration	$MIPS_o$	time step 1s			time step 10s			time step 20s			time step 40s		
		$MIPS_d$	$MIPS_n$	Improvement	$MIPS_d$	$MIPS_n$	Improvement	$MIPS_d$	$MIPS_n$	Improvement	$MIPS_d$	$MIPS_n$	Improvement
3×3	699.6	684.0	690.5	0.95%	683.5	689.3	0.84%	683.5	689.4	0.86%	683.8	687.4	0.53%
4×4	651.5	630.6	643.8	2.09%	628.9	642.1	2.10%	628.8	642.2	2.14%	628.4	641.8	2.11%
5×5	530.2	508.0	520.4	2.44%	508.9	519.7	2.10%	508.6	519.2	2.06%	508.2	518.8	2.08%
6×6	477.1	460.4	471.4	2.39%	459.6	470.9	2.46%	458.1	470.5	2.71%	457.5	470.4	2.82%
7×7	442.6	426.4	439.1	2.97%	423.8	438.0	3.35%	423.3	438.1	3.49%	422.7	437.5	3.51%
8×8	390.9	377.3	387.1	2.60%	375.9	386.6	2.85%	375.6	386.4	2.87%	375.3	386.2	2.90%
9×9	359.0	348.5	357.3	2.54%	347.3	356.7	2.69%	347.2	356.7	2.73%	347.0	356.3	2.67%
10×10	339.9	326.8	335.2	2.54%	325.6	334.5	2.74%	325.3	334.4	2.80%	325.0	333.7	2.66%

TABLE IV
THE AVERAGE RUNTIME OF THE NEW METHOD FOR ONE SECOND ON CPUS WITH DIFFERENT CORE CONFIGURATIONS.

Configuration	time step 1s		time step 10s		time step 20s		time step 40s	
	$t_p(10^{-3})$	$t_m(10^{-1})$	$t_p(10^{-4})$	$t_m(10^{-2})$	$t_p(10^{-4})$	$t_m(10^{-2})$	$t_p(10^{-4})$	$t_m(10^{-3})$
3×3	0.20	0.02	0.26	0.02	0.15	0.01	0.08	0.05
4×4	0.29	0.04	0.31	0.03	0.19	0.01	0.10	0.10
5×5	0.38	0.08	0.53	0.07	0.35	0.03	0.22	0.16
6×6	0.50	0.19	0.63	0.17	0.44	0.09	0.32	0.43
7×7	0.61	0.41	0.78	0.41	0.59	0.19	0.41	0.83
8×8	0.96	1.01	1.26	0.80	0.67	0.40	0.45	2.18
9×9	2.15	2.15	2.72	1.77	1.35	0.94	0.74	4.41
10×10	2.41	3.37	2.59	2.81	1.39	1.33	0.92	7.14

DTM algorithm runtime into MPC time (t_p) and matching time t_m, and recorded them in Table IV. It can be seem that the computing time of MPC and matching increases with the number of cores. Runtime of MPC is generally smaller than matching and increases slowly with the number of cores. But the time of matching increases quickly, and for the 9×9 CPU, matching process causes more than $0.2s$ computing time for $1s$ of DTM in the case of time step $1s$, which is obviously unacceptable. If the time step is large, the time is well. These data show that the new method is suitable for multi-core CPUs with number of cores not very large, and handling the future many-core CPUs exceeds the ability of the new method at current stage. Designing efficient DTM algorithm for such CPUs is our future work.

V. CONCLUSION

In this article, an MPC based hybrid DTM method combining task migration and DVFS techniques is proposed. The new method utilizes MPC to calculate the suitable power consumption to track a specified safe temperature ceiling. An algorithm, with both task migration and DVFS, is developed to adjust current core power consumptions according to the computed ones from MPC. Experimental results on several CPUs with different number of cores have shown the proposed method can successfully track the given temperature ceiling, and is able to enhance CPU performance by up to 4%, lower temperature variation by up to 50% compared to MPC assisted DVFS method.

REFERENCES

[1] J. Ma, H. Wang, S. Tan, C. Zhang, and H. Tang, "Hybrid dynamic thermal management method with model predictive control," in *IEEE Asia Pacific Conference on Circuits and Systems*, November 2014.

[2] D. Brooks, R. Dick, R. Joseph, and L. Shang, "Power, thermal, and reliability modeling in nanometer-scale microprocessors," *IEEE Micro*, vol. 27, no. 3, pp. 49–62, May-June 2007.

[3] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. Int. Symp. on Computer Architecture (ISCA)*, June 2006, pp. 78–88.

[4] M. Powell, M. Gomaa, and T. Vijaykumar, "Heat-and-Run: Leveraging SMT and CMP to manage power density through the operating system," in *Proc. of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, October 2004, pp. 260–270.

[5] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," in *Proc. Design Automation Conf. (DAC)*, June 2010, pp. 579–584.

[6] T. Chantem, S. Hu, and R. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 10, pp. 1884–1897, October 2011.

[7] G. Liu, M. Fan, and G. Quan, "Neigbor-Aware Dynamic Thermal Management for Multi-core Platform," in *Proc. European Design and Test Conf. (DATE)*, March 2012, pp. 187–192.

[8] R. Ayoub and T. Rosing, "Predict and act: Dynamic thermal management for multi-core processor," in *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, August 2009, pp. 99–104.

[9] T. Ebi, M. Al Faruque, and J. Henkel, "TAPE: thermal-aware agent-based power economy for multi/many-core architectures," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, 2009, pp. 302–309.

[10] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. on Computer Architecture (ISCA)*, 2003, pp. 2–13.

[11] R. Jayaseelan and T. Mitra, "A hybrid local-global approach for multi-core thermal management," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, 2009, pp. 314–320.

[12] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. De Micheli, and R. Gupta, "Processor speed control with thermal constraints," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 56, no. 9, pp. 1994–2007, September 2009.

[13] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. IEEE Int. Symp. on High-Performance Computer Architecture (HPCA)*, Jan. 2001, pp. 171–182.

[14] V. Hanumaiah, S. Vrudhula, and K. Chatha, "Performance optimal online DVFS and task migration techniques for thermally constrained multi-core processors," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1677–1690, November 2011.

[15] V. Hanumaiah and S. Vrudhula, "Energy-efficient operation of multicore processors by DVFS, task migration, and active cooling," *IEEE Trans. on Computers*, vol. 63, no. 2, pp. 349–360, February 2014.

[16] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, 2009.

[17] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Multicore thermal management with model predictive control," in *Proc. 19th European Conference on Cicuit Theory and Design*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 90–95.

[18] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *Proc. Int. Symp. on Computer Architecture (ISCA)*, 2009, pp. 314–324.

[19] 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 Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 170–183, January 2013.

[20] H. Wang, S. Tan, D. Li, A. Gupta, and Y. Yuan, "Composable thermal modeling and simulation for architecture-level thermal designs of multi-core microprocessors," *ACM Trans. on Design Automation of Electronics Systems*, vol. 18, no. 2, pp. 28:1–28:27, March 2013.

[21] H. Wang, S. Tan, G. Liao, R. Quintanilla, and A. Gupta, "Full-chip runtime error-tolerant thermal estimation and prediction for practical thermal management," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, Nov. 2011, pp. 716–723.

[22] D.-S. Chiou, S.-H. Chen, S.-C. Chang, and C. Yeh, "Timing driven power gating," in *Proc. Design Automation Conf. (DAC)*, 2006, pp. 121–124.

[23] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[24] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[25] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, May 2006.

[26] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. on Computer Architecture (ISCA)*, 2000, pp. 83–94.

[27] J. L. Henning, "SPEC CPU 2000: Measuring CPU performance in the new millennium," *IEEE computer*, vol. 1, no. 7, pp. 28–35, July 2000.