Minimizing Peak Temperature for Pipelined Hard Real-time Systems

Long Cheng†, Kai Huang∗†, Gang Chen†, Biao Hu†, and Alois Knoll†

†Chair of Robotics and Embedded Systems, Technical University Munich, Germany
∗School of Mobile Information Engineering, Sun Yat-Sen University
Email: †{chengl, huangk, cheng, hub, knoll}@in.tum.de ∗huangk36@mail.sysu.edu.cn

***Abstract*—This paper addresses the problem of minimizing the peak temperature for pipelined multi-core systems under hard end-to-end deadline constraints by adversely using the Pay-Burst-Only-Once principle. The Periodic Thermal Management is adopted to control the temperature and every core is periodically switched between two power modes. With the peak temperature representation, we first formulate the problem of finding the thermal optimal periodic schemes which satisfies deadline constraints and then present a fast heuristic algorithm to solve it. Adopting real life processor platforms and applications, our simulation demonstrates that our approach reduces the peak temperature by up to 15°C on the 4-stage ARM platform compared to sub-deadline partition approach. Moreover, our algorithm is shown to be scalable w.r.t. the number of pipelined stages and its effectiveness is validated by the brutally searching approach.**

I. Introduction

With the increasing demand of computational performance, multi-core architectures are now widely adopted for products. To date, processors having 64 or more cores are available in the market. The architecture with such a high degree of parallelism poses designers a challenge: how to extract parallelism from applications and exploit them efficiently.

Pipelined computing, which can reduce the latency of a stream application, is a promising paradigm for real-time systems. The pipelined computing connects a set of processing units in series and executes the sub-tasks of the application on the pipelined processing system. In this way, the sub-tasks can be executed simultaneously, that is, parallel processing is performed. Therefore, pipelined computing can efficiently exploit the hardware performance advantage of multi-core processors and decrease the latency.

For hard real-time pipelined systems, bounding the latency is crucial for the correctness of the system. However, as power density is increasing exponentially with Moore's Law, the peak temperature on modern processors is rapidly elevated, which seriously threats the reliability and performance of the system. Since reducing the temperature usually requires decreasing power consumption, which means lower performance and larger latency, the trade-off between performance and temperature constraints should be made carefully. Therefore, it's an important and challenging task that designing a scheduling policy to optimize the peak temperature under the end-to-end deadline constraint for a pipelined real-time system on a multi-core processor .

This paper focuses on this issue and addresses the optimization problem by reversely using the Pay-Burst-Only-Once (PBOO) principle. Our work is inspired by the work of Chen et al. [1], which minimizes the total power consumption for pipelined stage systems. However, their work cannot be directly transplanted to temperature optimization, due to the reasons: (1) although temperature is a strong function of power, power management techniques that are effective for energy saving may not be suitable for temperature managing [2], which has already been theoretically proved by [3]. (2) The quadratic programming problem formulation in [1] cannot be reused, since calculating temperature is based on convolution while computing the total power consumption is based on integral. Therefore, the problem of temperature minimizing demands new analysis and approach, which are the main contributions of this paper:

- Based on the well-known HotSpot model, a peak temperature representation for a multi-core processor under Periodic Thermal Management PTM is given, where the heat flow between cores and the leakage current dependency on temperature (LDT) are considered.
- By reversely using the Pay-Burst-Only-Once principle, the optimization problem is transformed into a set of sub-problems. We formulate the sub-problem and solve it by a fast heuristic algorithm whose computing time grows (approximated) logarithmically with the number of stages.
- Based on two real life platforms: a homogeneous ARM multi-processor and the Intel Single-chip Cloud Computer (SCC), we evaluate the effectiveness and efficiency of our approach by comparing it with two brutally searching approaches, one with PBOO and one without PBOO.

The rest of this paper is organized as follows: Section II gives a brief introduction of related works. Our system models are introduced in Section III and Section IV shows a motivation example and presents the problem statement. We analyze the peak temperature in Section V and discuss our approach in Section VI. Section VII details the case studies and Section VIII concludes.

II. Related works

In this section, we briefly overview previous works on thermal-aware system scheduling policies and categorize them according if pipelined computing is considered.

Pipelined Computing Chen and et al. [1] utilized PBOO principle and presented an approach to optimize the power consumption of a pipelined system under the deadline constraint. An quadratic programming formulation of the problem is proposed and two methods are studied to solve the problem. As we stated above, the power optimal approach may not be thermal optimal. Therefore, a new approach is needed for the problem of peak temperature minimizing. There have already been several thermal management approaches for pipelined systems. However, previous works on this topic either don't consider the hard deadline constraints [4, 5] or just reduces the deadline misses percentage into a lower range [6]. Our approach considers the deadline constraint and gives hard real-time guarantee.

Multi-core Processors Aiming to minimize the chip peak temperature while satisfying the hard real-time constraints of an MPSOC, Thidapat and et al. [7] addressed the problem of assigning and scheduling tasks on the MPSOC. Due to the thermal analysis difficulties, this approach examines only steady-state temperatures without considering the transient behavior. In this paper, we provide a peak temperature formulation which considers the transient temperature. Yong and et al. [8] presented a feedback thermal control framework named Real-Time Multicore Thermal Control which dynamically enforces both the desired temperature and the CPU utilization bounds for multicore real-time systems, through DVFS. Buyoung [9] addressed the problem of avoiding thermal hotspot on a multi-core chip by employing a runtime thermal aware scheduler (TAS) using job-migration and power-gating techniques. In [10], Pradeep extended the concept of Thermal-Resiliency to multi-core architecture and then adopted a control-theoretic framework to ensure hard-real-time deadlines in a dynamic thermal environment while maintaining the thermal constraints. Above works all assumed simple task models, i.e., either periodic tasks or sporadic task model. In this paper, the task streams are modeled by a more general concept, arrival curve, therefore we can preserve more information such as the non-determinism of the event arrivals in the model.

III. SYSTEM MODEL

Notation: All matrices and vectors are denoted by bold characters.

Hardware Model In this paper, a multi-core processor which can handle partitioned applications is considered. The sub-tasks of a partitioned application can be mapped and executed on different cores which communicate with each other via FIFOs. Each core has two power dissipation modes, namely 'active' and 'sleep' mode. In 'active' mode, the core works with higher power consumption and tackles input events in a fixed frequency. We also consider the mode-switching overhead. To switch the core i from 'active' mode to 'sleep' mode and back, t^i_{swoff} and t^i_{swon} time units are required, respectively. During mode-switching, the power consumption equals that in 'active' mode. Moreover, no coming event can be handled in mode-switching or 'sleep' mode. Due to time overhead in mode-switching, the time lengths for which a core is switched to 'active' and 'sleep' mode must be larger than t_{swon} and t_{swoff}, respectively: $t_{off} > t_{swoff}$, $t_{on} > t_{swon}$. For brevity, we define $\mathbf{t_{swoff}} = (t^1_{swoff}, t^2_{swoff}, \cdots, t^n_{swoff})$ and $\mathbf{t_{swon}} = (t^1_{swon}, t^2_{swon}, \cdots, t^n_{swon})$ for a n-core processor.

Task Model We study the streaming applications which can be split into several sub-tasks. To model general task arrivals, the concept of arrival curve [11, 12, 13] $\bar{\alpha}(\Delta) = [\bar{\alpha}^u(\Delta), \bar{\alpha}^l(\Delta)]$ is introduced in this paper. The upper arrival curve $\bar{\alpha}^u(\Delta)$ and the lower arrival curve $\bar{\alpha}^l(\Delta)$ are the upper and lower bound of $R(t)$:

$$\bar{\alpha}^l(\Delta) \le R(t) - R(s) \le \bar{\alpha}^u(\Delta), \forall t - s = \Delta \tag{1}$$

where $R(t)$ is the cumulative workload and represents the number of tasks arriving in time interval $[0,t)$. On the same way, the service curve $\beta(\Delta)$ is employed to model available resources in any time interval Δ. $\beta(\Delta)$ provides the upper and lower bound of another cumulative function $C(t)$, which is the amount of total time slots that the processor provides to handle the tasks in time interval $[0,t)$.

It's worth noting that arrive curve is event-based and specifies the number of input events in any time interval Δ while service curve describes the amount of available execution time. Therefore service curve $\beta(\Delta)$ should be transformed to a event-based service curve $\bar{\beta}(\Delta)$, which can be accomplished by $\bar{\beta}^u(\Delta) = \beta^u(\Delta)/c$ and $\bar{\beta}^l(\Delta) = \beta^l(\Delta)/c$ [14], where c is the worst-case execution time (WCET). With these definitions, we

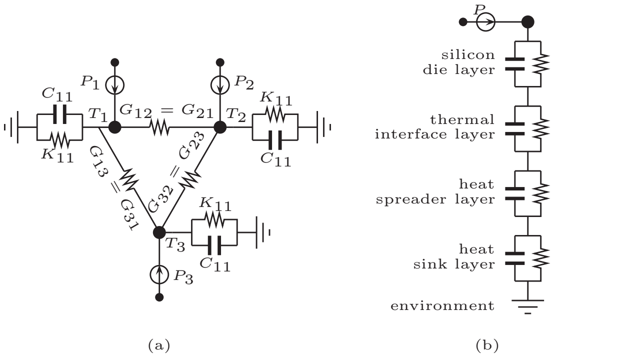

Fig. 1. (a) The equivalent RC circuit of the silicon layer for a floorplan with three processing components and (b) The vertical layout model.

say a system with service curve $\bar{\beta}(\Delta)$ satisfies the worst-case deadline D for a streaming application modeled by $\alpha(\Delta)$ if the following condition holds:

$$\bar{\beta}^l(\Delta) \ge \alpha^u(\Delta - D), \forall \Delta \ge 0 \tag{2}$$

Thermal Model The well established thermal model HotSpot is employed to model the multi-core processor [15]. The vertical layout of the processor is modeled by four layers, which are the heat sink, heat spreader, thermal interface and silicon die layers [16]. Each layer is divided into a number of blocks according to the processing components on the die. Moreover, the blocks are ordered in a way that all the processing components occupy the beginning part of the order list. To predict the temperature evolution, we take the advantage of the well-known electro-thermal analogy [7, 10, 15, 16, 17, 18, 19, 20], i.e., the RC thermal network. Every thermal block is mapped onto a node of the thermal circuit. An example of the model can be found in Fig. 1. Finally, the temperature vector $\mathbf{T}(t)$ can be determined by a set of first-order differential equations [16]:

$$\mathbf{C} \cdot \frac{\mathrm{d}T(t)}{\mathrm{d}t} = (\mathbf{P}(t) + \mathbf{K} \cdot \mathbf{T}^{amb}) - (\mathbf{G} + \mathbf{K}) \cdot \mathbf{T}(t) \tag{3}$$

where $\mathbf{C}$ is the thermal capacitance matrix, $\mathbf{P}$ is the power dissipation vector, $\mathbf{K}$ is the thermal ground conductance matrix, $\mathbf{G}$ is the thermal conductance matrix and $\mathbf{T}^{amb}$ is the ambient temperature vector which is defined as $\mathbf{T}^{amb} = T^{amb} \cdot [1, \cdots, 1]'$, where T^{amb} is the ambient temperature.

We assume the power vector $\mathbf{P}$ is the sum of the power $\mathbf{P}^d$ due to dynamic current and the power $\mathbf{P}^l$ due to leakage current [18, 21]. In 'active' and 'sleep' state $\mathbf{P}^d$ is assumed to be constant, i.e., $\mathbf{P}^a$ and $\mathbf{P}^i$, respectively. The dependency relationship between the leakage power and the temperature can be closely approximated by a linear function of the processor temperature [16, 17, 18, 22, 23]: $\mathbf{P}^l = \mathbf{W} \cdot \mathbf{T}(t) + \mathbf{V}$, where $\mathbf{W}$ is a diagonal matrix with constant coefficients and $\mathbf{V}$ is a vector with constant coefficients. Therefore, $\mathbf{P}$ can be represented as $\mathbf{P}(t) = \mathbf{W} \cdot \mathbf{T}(t) + \mathbf{V}^a$ if in active mode or $\mathbf{P}(t) = \mathbf{W} \cdot \mathbf{T}(t) + \mathbf{V}^i$ otherwise, where $\mathbf{V}^a = \mathbf{V} + \mathbf{P}^a$ and $\mathbf{V}^i = \mathbf{V} + \mathbf{P}^i$. Rewriting (3) with the power formulation, we can obtain the state space representation of the thermal model:

$$\frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} = \mathbf{A} \cdot \mathbf{T}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{4}$$

where $\mathbf{u}(t)$ is the input vector, $\mathbf{A} = -\mathbf{C}^{-1} \cdot (\mathbf{G} + \mathbf{K} - \mathbf{W})$ and $\mathbf{B} = \mathbf{C}^{-1}$. If component j is in active mode, $u_j(t) = V^a_j + K_{jj} \cdot T^{amb}$, otherwise $u_j(t) = V^i_j + K_{jj} \cdot T^{amb}$. Since $\mathbf{A}$ and $\mathbf{B}$ are constant, the thermal model is a first order linear time invariant system (LTI) and the closed-form representation of the temperature is:

$$\mathbf{T}(t) = e^{\mathbf{A}\cdot t}\cdot \mathbf{T}^0 + \int_{-\infty}^{\infty}\mathbf{H}(\xi)\cdot \mathbf{u}(t-\xi)d\xi \tag{5}$$

where $\mathbf{H}(t) = e^{\mathbf{A}\cdot t}\cdot \mathbf{B}$ is the matrix describing the impulse response between any two nodes. The self-impulse response $H_{ii}(t)$ is a non-negative decreasing function [16]. Regarding $H_{ij}(t)$ where $i \neq j$, we adopt the conjecture proposed in [16] that $H_{ij}(t)$ is a non-negative unimodal function. Let $\mathbf{T}^{init}(t) = e^{\mathbf{A}\cdot t}\cdot \mathbf{T}^0$, $H_{ij}(t) = u_i(t) = 0$ for $t \leq 0$, and $T_{ij}^{conv}(t) = \int_0^t H_{ij}(\xi)\cdot u_j(t-\xi)d\xi$, the temperature of node i yields:

$$T_i(t) = T_i^{init}(t) + \sum_{j=1}^{N} T_{ij}^{conv}(t). \tag{6}$$

To deduce the difficulties of calculating the peak temperature, we examine the model closely and make two observations that scale down the temporal and spatial exploring spaces.

1) For any i, j and a sufficient large t, we have $H_{ij}(t) = 0$ and $T_i^{init}(t) = 0$. The reason is our system is BIBO stable [24, 25], which is assured if and only if $\int_{-\infty}^{\infty}|H(t)|dt < \infty$ [25]. Considering $H_{ij}(t)$ is a non-negative function, one can prove that $H_{ij}(t)$ will approaches zero as t approaches infinity. Since $\mathbf{T}^{init}(t) = \mathbf{H}(t)\cdot \mathbf{C}\cdot \mathbf{T}^0$, $T_{ij}^{init}(t)$ also has this property.
2) The peak temperature can only occur on the processing component nodes. The intuition is that according to Second law of thermodynamics, in our thermal model, heat can only flow from a hotter node to a colder one. Since heat are generated from the processing components, the temperature on them will be higher.

IV. Problem Statement

In this paper, we deploy Periodic Thermal Management [26] to manage the temperature of the chip by periodically switching every stage between two power consumption states with an individual pair of (t_{on}, t_{off}). Therefore, two vectors, $\mathbf{t_{off}} = (t_{off}^1, t_{off}^2, \cdots, t_{off}^n)$ and $\mathbf{t_{on}} = (t_{on}^1, t_{on}^2, \cdots, t_{on}^n)$, should be determined offline to specify the PTMs deployed on the system. For the details of PTM, we refer to [26]. Now, we present a motivation example to illustrate the advantages of applying Pay Burst Only Once (PBOO) for thermal optimization. For comparison, the PTM schemes are derived from two approaches: the PBOO based approach (PBOO) and the one which partitions the end-to-end deadline into substage deadlines for each stage, namely SDP (Sub-Deadline Partition).

Motivation example In the example, an event stream with arrival curve $\alpha = 0.15\Delta + 2$ and deadline $D = 35ms$ passes through a two-stage pipelined system. The WCETs are set as $c_1 = c_2 = 1ms$, respectively. In this case, We set $\mathbf{t_{off}} = (5, 13)ms$ and then compare the $\mathbf{t_{on}}$s generated by the two methods. Fig. 2 graphically illustrates the derivation process corresponding to the two methods.

Let's first examine the strategy of SDP. We divide the deadline D into two sub-deadlines, $D_1 = 10ms$ and $D_2 = 25ms$, in this case. For simplicity, we adopt the bounded-delay function (BDF) [26] to calculate $\mathbf{t_{on}}$. A BDF is defined as $bdf(\Delta) = \max[0, \rho(\Delta - b)]$. For the first stage, given deadline $D_1 = 10ms$, the service demand should be $\beta_1 = \alpha(\Delta - 10)$. Since $t_{off}^1 = 5ms$, we obtain the slope of the BDF in the first stage as $\rho_1 = (2-0)/(10-5) = 0.4$ and then $\frac{t_{on}^1}{t_{on}^1 + t_{off}^1} = c_1 * \rho_1 = 0.4$, which indicates $t_{on}^1 = 3.3ms$. To derive the service demand for the second stage, the output arrive curve $\hat{\alpha}$ from the first stage is needed. From [11], $\hat{\alpha} = \alpha \oslash \beta_1^{srv} = 0.15\Delta + 2.7$, where β_1^{srv} is the service curve of first stage and is a TDMA curve specified by t_{off}^1 and t_{on}^1. On the same way, we have the slope of the BDF in second stage as $\rho_2 = 0.225$ and finally $t_{on}^2 = 3.7ms$.

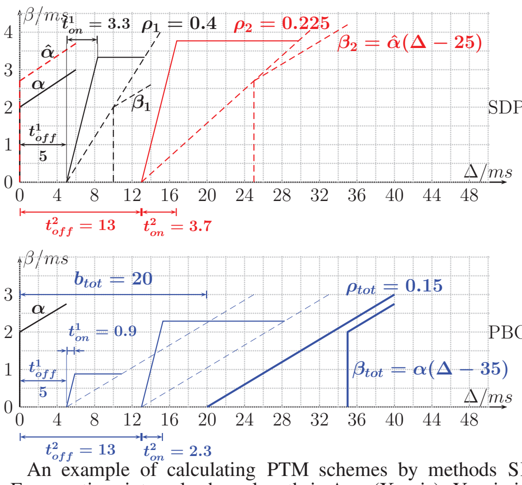

Fig. 2. An example of calculating PTM schemes by methods SDP and PBOO. For any time interval whose length is Δms (X axis), Y axis indicates the least number of time slots that the processor should provide for handling the workload to meet the deadline constraint.

Now, the PBOO method is utilized to get $\mathbf{t_{on}}$. Unlike the procedure in approach SDP, the total service demand is first obtained as $\beta_{tot} = \alpha(\Delta - 35)$ and then a BDF $bdf_{tot} = \max[0, \rho(\Delta - b)]$ should be determined such that the deadline constraint can be met as long as $\beta_1^{srv} \otimes \beta_2^{srv} \geq bdf_{tot}$. From (13), we can set b_{tot} as its minimum $b_{tot} = t_{off}^1 + t_{off}^2 + c_1 + c_2 = 20ms$, then the slope in bdf_{tot} can be determined to calculate $\mathbf{t_{on}}$: $\rho_{tot} = 0.15$, which is much smaller than ρ_1 and ρ_2, therefore, results in smaller $t_{on}^1 = 0.9ms$ and $t_{on}^2 = 2.3ms$.

The pessimism in the SDP method comes from paying an additional burst and delay when $\hat{\alpha}$ is calculated for the second stage, as pay-burst-only-once principle points out [11]. Moreover, as the stage number increases, this effect is accumulated and then causes more pessimistic results. On the other hand, PBOO directly calculates the total service demand and then retrieves $\mathbf{t_{on}}$ for every stage, which pays the burst only once and gets better results. Since lower partition of t_{on} means lower temperature of the processor, we can see that employing PBOO will achieve lower peak temperature than using SDP. Therefore, by reversely using pay-burst-only-once, we can avoid paying the burst repeatedly therefore better optimize the peak temperature for pipelined systems, especially for scenarios of many stages.

Problem Statement Now we define our problem as: *Given a n-stage pipelined platform specified by the above hardware and thermal models, an event stream with arrival curve* α *and end-to-end deadline D, and the* WCET*s* $\mathbf{c} = (c_1, c_2, \cdots, c_n)$*, our goal is to find the* PTM *schemes characterized by* $\mathbf{t_{off}}$ *and* $\mathbf{t_{on}}$ *such that the peak temperature is minimized while the deadline constraint is satisfied.*

V. Peak Temperature Analysis

In this section, we formulate the peak temperature of a multi-core processor which adopts PTM. Based on the first observation of the thermal model, some notations are defined. Let (1) t_{ij}^{conv} denote the certain time point after which $H_{ij}(t)$ can be considered as zero, (2) t_k^{init} denote the similar time point for $T_k^{init}(t)$, and (3) m and N indicate the numbers of total processing components and blocks in thermal model, respectively. Now some basic lemmas are shown for further analysis.

Lem. 1: For any $i \le N$ and $j \le m$, when $t \ge t_{ij}^{conv}$, $T_{ij}^{conv}(t)$ is a periodic function in the domain of t.

Lem. 2: When $t \ge t_i^{end}$, $T_i(t)$ is a periodic function, where $t_i^{end} = \max[t_i^{init}, t_{i1}^{conv}, \cdots, t_{im}^{conv}]$.

Lem. 3: For node i, the peak temperature $T_i^\star$ equals the maximal local peak temperature which is reached after t_i^{end}.

The proofs of above lemmas are omitted due to space limit.

Lem. 4: The peak temperature of node i is:

$$T_i^\star = \max\{\sum_{j=1}^{N}[T_{ij}^{conv}(t) | t_i^{end} \le t \le t_i^{end} + t_p^{lcm}]\}. \tag{7}$$

Proof: From Lem. 3, we can know that $T_i^\star$ can be obtained by finding the maximum of $T_i(t)$ for $t \ge t_i^{end}$. According to Lem. 2, when $t \ge t_i^{end}$, $T_i^\star$ is a periodic function and its period is t_p^{lcm}. Therefore the the maximum of $T_i(t)$ is the local maximum in every period and can be formulated as (7). ■

Based on above lemmas and definition, we present our first important result in the following theory.

Thm. 1: For a multi-core processor with hardware and thermal models described above, when the PTM schemes characterized by $\mathbf{t_{off}}$ and $\mathbf{t_{on}}$ are applied, the peak temperature of the processor can be formulated as:

$$T^\star = \max\{T_1^\star, T_2^\star, \cdots, T_m^\star\} \tag{8}$$

Proof: Based on the second observation of thermal model, the peak temperature of the processor must be the peak temperature of the processing component nodes, which are the first m nodes in the model. Therefore, we have $T^\star = \max\{T_1^\star, T_2^\star, \cdots, T_m^\star\}$. ■

VI. Algorithms

In this section, we first transform our optimization problem into a set of sub-problems, which can be formulated as below, then provide a fast heuristic to solve the sub-problem.

A. Real-time analysis and formulation

Before giving the formulation, we first present the timing property analysis to ensure that all tasks are finished before their deadlines. For a n-stage pipelined processor that employs PTM schemes characterized by $\mathbf{t_{off}}$ and $\mathbf{t_{on}}$, we define $K_i = \frac{t_{on}^i}{t_{on}^i + t_{off}^i}$ and $\mathbf{K} = \{K_1, \cdots, K_n\}$. Then, suppose an application with arrival curve α is processed by the n pipelined stages, the deadline D can be satisfied if the following condition holds [1][1]:

$$\min_{i=1}^{n}(\frac{K_i}{c_i})[\Delta - \sum_{i=1}^{n}(t_{off}^i + c_i)] \ge \alpha^u(\Delta - D) \tag{9}$$

[1]See Thm.1 in [1]

The right hand side of the inequality can be upper-bounded by a set of minimum bounded-delay functions $bdf_{min}(\Delta) = \max[0, \rho(\Delta - b)]$. It's worth noting that b can vary in a feasible region $[b_{min}, b_{max}]$, which is obtained from [26]. For a given b, the corresponding ρ is calculated from:

$$\rho(b) = \inf\{\rho : \rho(\Delta - b) \ge \alpha^u(\Delta - D), \forall \Delta \ge 0\} \tag{10}$$

For every individual $bdf_{min}(\Delta) \ge \alpha^u(\Delta - D)$, the deadline D is satisfied when the following inequity holds:

$$\min_{i=1}^{n}(\frac{K_i}{c_i})[\Delta - \sum_{i=1}^{n}(t_{off}^i + c_i)] \ge bdf_{min}(\Delta, \rho, b) \tag{11}$$

which can be transformed to:

$$\min_{i=1}^{n}(\frac{K_i}{c_i}) \ge \rho \quad \& \quad \sum_{i=1}^{n}(t_{off}^i + c_i) \le b \tag{12}$$

Then, from the peak temperature formulation, the sub-problem can be formulated for every individual pair of b and ρ.

$$\begin{aligned}
\text{minimize} \quad & T^\star(\mathbf{K}, \mathbf{t_{off}}) = \max\{T_1^\star, T_2^\star, \cdots, T_m^\star\} \\
\text{subject to} \quad & \min_{i=1}^{n}(\frac{K_i}{c_i}) \ge \rho \\
& \sum_{i=1}^{n}(t_{off}^i + c_i) \le b \\
& T_i^\star = \max\{\sum_{j=1}^{N}[T_{ij}^{conv}(t) | t_i^{end} \le t \le t_i^{end} + t_p^{lcm}]\} \\
& T_{ij}^{conv}(t) = \int_0^t H_{ij}(\xi) \cdot u_j(t - \xi) d\xi \\
& u_j(t) = \begin{cases} V_j^a + K_{jj} \cdot T^{amb} & \text{if active or in mode-switching} \\ V_j^i + K_{jj} \cdot T^{amb} & \text{otherwise} \end{cases} \\
& 0 \le K_i \le 1, i = 1, 2, \cdots, n \\
& t_{off}^i \ge t_{swoff}^i, i = 1, 2, \cdots, n \\
& t_{on}^i \ge t_{swon}^i, i = 1, 2, \cdots, n
\end{aligned} \tag{13}$$

With this formulation, Algo. 1 provides the pseudo-code of our approach.

Algorithm 1 Peak Temperature Optimization

```
Input: Thermal model, $\alpha$, $D$, $n$, $\mathbf{c}$, $\mathbf{t_{swon}}$, $\mathbf{t_{swoff}}$, $\varepsilon$
Output: $\mathbf{K}$, $\mathbf{t_{off}}$
1: calculate $[b_{min}, b_{max}]$.
2: $T_{min}^\star \leftarrow \infty$, $\mathbf{K} \leftarrow \mathbf{0}$, $\mathbf{t_{off}} \leftarrow \mathbf{0}$
3: for $b = b_{min}$ to $b_{max}$ with step $\varepsilon$ do
4:     get $\rho$ from (10)
5:     solve sub-problem (13) and get $T^\star$, $\mathbf{K}^\diamond$, $\mathbf{t_{off}^\diamond}$
6:     if $T^\star < T_{min}^\star$ then
7:         $T_{min}^\star \leftarrow T^\star$, $\mathbf{K} \leftarrow \mathbf{K}^\diamond$, $\mathbf{t_{off}} \leftarrow \mathbf{t_{off}^\diamond}$
8:     end if
9: end for
```

B. Solving sub-problem

Now we present a fast algorithm which is inspired by the gradient descent method to solve problem (13).

Lem. 5: For problem (13), K_i can be obtained safely by the following equation:

$$K_i = c_i \rho \tag{14}$$

Proof: From the definition of K_i, one can derive $t^i_{on} = \frac{K_i}{1-K_i} t^i_{off}$. Then, we have $\frac{dt^i_{on}}{dK_i} > 0$, which means a larger K_i results in a higher partition of t^i_{on}, that is, a higher peak temperature. Therefore K_i should equal its lower bound $c_i\rho$ such that the peak temperature won't be elevated unnecessarily. ■

Now, $\mathbf{t_{off}}$ remains to be determined. Intuitively, one can brutally search the whole exploring space to find the optimal solution. However, as the stage number n increases, the exploring space expends (approximated) exponentially and this approach will finally be infeasible. For example, if $n = 8$ and every t_{off} has 50 candidates in the exploring space, there will be $50^8 = 3.90625 \times 10^{13}$ combinations in total. The brutally searching algorithm needs more than 120 years to finish if the computer can check 10000 combinations per second. Therefore, a more clever algorithm is needed to solve sub-problem (13). Inspired by the gradient descent algorithm, we present a fast algorithm to find the optimal $\mathbf{t_{off}}$.

Algorithm 2 Sub-optimization with given b and ρ

```
Input: Thermal model, $n$, $b$, $\rho$, $\mathbf{c}$, $\mathbf{t_{swon}}$, $\mathbf{t_{swoff}}$, $\xi$
Output: $\mathbf{K}$, $\mathbf{t_{off}}$
1: $\mathbf{e_1} = (1,0,\cdots,0)$, $\mathbf{e_2} = (0,1,\cdots,0)$, $\mathbf{e_n} = (0,0,\cdots,1)$
2: $\mathbf{K} \leftarrow \rho \cdot \mathbf{c}$, $\mathbf{t_{off}} \leftarrow \mathbf{t_{swoff}}$, go$\leftarrow 1$, $T^\star_{last} \leftarrow T^\star(\mathbf{K}, \mathbf{t_{off}})$
3: while go do
4:     if $\sum_{i=1}^n (t^i_{off} + c_i) + \xi \le b$ then
5:         for $i = 1$ to $n$ with step 1 do
6:             $g(i) = T^\star(\mathbf{K}, \mathbf{t_{off}} + \xi\mathbf{e_i}) - T^\star_{last}$
7:         end for
8:         if $\min g(i) \ge 0$ then
9:             go = 0
10:        else
11:            find $i$ where $g(i) == \min g(i)$
12:            $\mathbf{t_{off}} = \mathbf{t_{off}} + \xi\mathbf{e_i}$, $T^\star_{last} \leftarrow T^\star_{last} + g(i)$
13:        end if
14:    else
15:        for every pair of $i, j$ that $1 \le i, j \le n$ do
16:            $G(i,j) = T^\star(\mathbf{K}, \mathbf{t_{off}} + \xi\mathbf{e_i} - \xi\mathbf{e_j}) - T^\star_{last}$
17:        end for
18:        if $\min G(i,j) \ge 0$ then
19:            go = 0
20:        else
21:            find $i$ and $j$ where $G(i,j) == \min G(i,j)$
22:            $\mathbf{t_{off}} = \mathbf{t_{off}} + \xi\mathbf{e_i} - \xi\mathbf{e_j}$, $T^\star_{last} \leftarrow T^\star_{last} + G(i,j)$
23:        end if
24:    end if
25: end while
```

Algo. 2 outlines the pseudo-code of the algorithm. It takes the thermal model, stage number n, mode-switching overhead vectors, b, ρ, $\mathbf{c}$, and a fixed stepsize ξ as input. The iteration starts at the initial point $\mathbf{t_{swoff}}$ (line 2). In every iteration, it first checks whether feeding one ξ to $\mathbf{t_{off}}$ will exceed the limit of b (line 4). If not, ξ will be added to one of the t_{off}s (line 12). If the limit of b is hit, we will subtract ξ from anther t_{off} at the same time(line 22). In every iteration, all the possible directions are checked and the direction leading to the current steepest descent will be selected to update $\mathbf{t_{off}}$ (lines 5-7, 11-12, 15-17, 21-22). The algorithm executes the iteration until evolving towards all possible directions with step ξ won't result in a lower peak temperature (lines 8-9, 18-19). It's worth noting that based on a set of systemic simulations, we conjecture that the peak temperature in Thm.(8) is a convex function of t^i_{off}, which means $T^\star(\mathbf{t_{off}})$ has only one minimum in the domain of $\mathbf{t_{off}}$. Therefore, the result found by Algo. 2 is the global minimum in the exploring space.

VII. Case Studies

We evaluate the effectiveness and feasibility of our proposed approach in this section. Three approaches are compared: (1) our Gradient Descent based PBOO algorithm (GD), (2) Brutally Searching based PBOO algorithm (BS), and (3) the brutally searching Sub-Deadline Partitions algorithm introduced in section IV (SDP). SDP brutally examines all the possible sub-deadline partitions and returns the one yielding the lowest peak temperature.

A. Setup

We implement the approaches on two simulation platforms: (1) a homogeneous multi-processor ARM platform with eight cores (ARM), (2) the Single-Chip Cloud Computer (SCC), a processor created by Intel that has 48 distinct physical cores [27]. The power and thermal parameters of the two platforms come from [16, 28] and parameter calibration. The thermal matrices $\mathbf{G}$, $\mathbf{C}$ and $\mathbf{K}$ are obtained from the HotSpot toolbox. All the simulations are performed on a computer with an Intel i7-4770 processor and 16GB memory. Regarding determining the layout of activated cores, we select and activate the n cores whose locations are close to core #1.

Our simulation runs the peak temperature optimization for three partitioned applications: (1) the H.263 decoder application modeled by four tasks [29], (2) the MP3 decoder application which can be split into five tasks [29], and (3) the MADplayer application that consists of five tasks [30]. Their worst-case execution times are determined and scaled from [29] and [30], respectively. All the mode-switching overheads are $\mathbf{t_{swoff}} = \mathbf{t_{swon}} = (1,\cdots,1)ms$. The activation periods of the three applications are set as $50ms$, $60ms$ and $50ms$, respectively. The relative deadline D is determined by deadline factor δ: $D = \delta \times p$, where p is the activation period.

B. Results

The three applications are executed with deadline factor $\delta = 0.9$ and the peak temperatures on different platforms are examined for three- and four- stage scenarios. Fig. 3 provides results on ARM platform while Fig. 4 shows those from SCC platform. From the figures we can see that: (1) In all the cases, the peak temperatures obtained from BS and GD are identical in value, which proves the effectiveness of our fast heuristic method. (2) For the two platforms, the temperature difference between our PBOO based algorithms and algorithm SDP gets bigger when stage number increases. This is due to that SDP pays burst for more times when stage number increases and therefore returns higher peak temperatures. (3) Compared to ARM, the peak temperature and the gap between the approaches are much lower on SCC platform, which is owed to (a) the difference in the thermal parameter, such as chip thickness, heat sink size. (b) SCC has 48 cores, only turning on three or four cores won't warm the whole chip sufficiently, therefore the heat can be conducted to the environment faster. To further confirm the effectiveness and feasibility of our approach, we simulate a randomly generated application on the two platforms and then increase the stage number n from 2 up to 8 on ARM and to 12 on SCC (simulating up to 48 stages needs a huge figure to display the results and simulating up to 12 stages is enough to determine the scalability). The WCETs of the sub-tasks are randomly generated between $[6,8]ms$ and the application is activated every $100ms$ with $\delta = 1.2$. The results are shown in Fig. 5. Due to that SDP and BS may suffer from exploring space explosion as stage number increases, we terminate their simulation when n reached 7 and 11, respectively. For clarity, only the time expense on SCC platform is shown in the figure. Observe

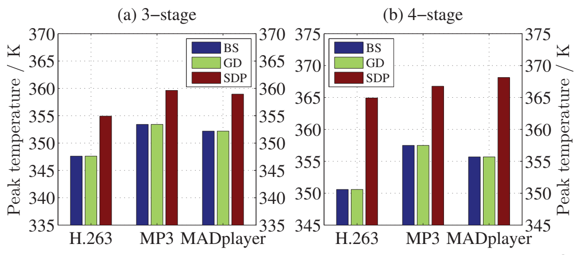

Fig. 3. Peak Temperature produced by the tested approaches with $\delta = 0.9$ when the applications executed on platform ARM with different stage number.

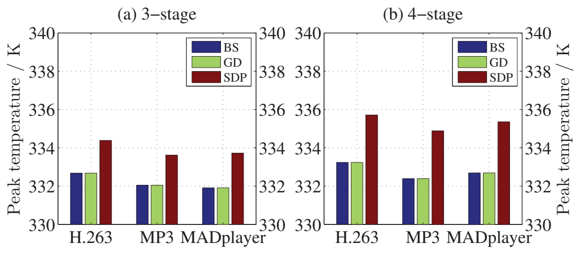

Fig. 4. Peak Temperature produced by the tested approaches with $\delta = 0.9$ when the applications executed on platform SCC with different stage number.

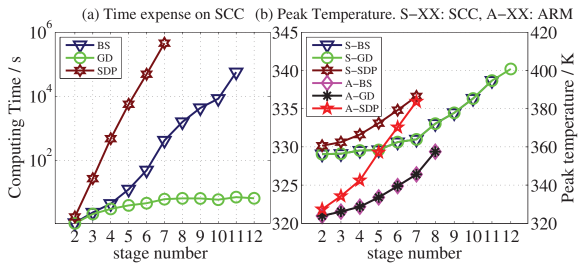

Fig. 5. (a) Computing time on SCC platform and (b) Peak Temperature of the tested approaches on SCC and ARM platforms, the right Y axis indicates the results on ARM platform.

that the time required by GD generally is the lowest and the curve is nearly flat as n increases, which indicates GD is feasible for pipelined systems with many stages. Fig. 5a also shows that the computing time consumed by SDP is always the highest and grows exponentially. This is because SDP examines all the possible deadline partitions, the amount of which increases exponentially as the stage number increases. Moreover, computing the service demand for every following stage requires numerical min-plus convolution, which incurs significant computation and memory overhead. Similarly, we find that the time overhead of BS grows exponentially as stage number increases. Therefore, we can say that SDP and BS are not scalable with the stage number regarding the requirement for computing resource. As suggested by Fig. 5b, we can clearly see that the peak temperature generated by GD always equals that from BS, which further strength the effectiveness of GD. We notice that the peak temperature gap between the approaches is bigger in platform ARM than SCC, as we explained above. Fig. 5b also demonstrates that the temperature difference between SDP and GD widens as stage number increases, which is expected because SDP pays burst more times and therefore generates PTM schemes in which $\mathbf{t_{on}}$ occupies bigger partition.

VIII. Conclusion

We have proposed a new approach to minimize the peak temperature of a pipelined hard real-time system by reversely utilizing the Pay-Burst-Only-Once principle. The problem is transformed into a set of sub-problems and a fast heuristic algorithm is proposed to solve the problem. We conduct simulation of our approach on two actual platforms for real life applications and the results show that our approach reduces the peak temperature more efficiently than the approach without PBOO, especially for many-stage scenarios. It is also shown that the time expense our fast algorithm grows (approximated) logarithmically as the stage number increases, indicating the algorithm is scalable with the number of stages.

IX. Acknowledgements

This work has been partly funded by China Scholarship Council, German BMBF projects ECU (grant number: 13N11936), and China SYSU the Fundamental Research Funds for the Central Universities: 15lgjc32.

References

[1] Gang Chen and et al. Applying pay-burst-only-once principle for periodic power management in hard real-time pipelined multiprocessor systems. *TODAES*, 20(2):26, 2015.
[2] Sushu Zhang and Karam S Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *TCAD*, pages 281–288. IEEE, 2007.
[3] Nikhil Bansal and Kirk Pruhs. Speed scaling to manage temperature. In *STACS*, pages 460–471. Springer, 2005.
[4] Andrea Alimonda and et al. Temperature and leakage aware power control for embedded streaming applications. In *DSD*, pages 107–114. IEEE, 2008.
[5] Marco Cox and et al. Thermal-aware mapping of streaming applications on 3d multi-processor systems. In *ESTIMedia*, pages 11–20. IEEE, 2013.
[6] Fabrizio Mulas and et al. Thermal balancing policy for multiprocessor stream computing platforms. *TCAD*, 28(12):1870–1882, 2009.
[7] Thidapat Chantem and et al. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. *VLSI*, 19(10):1884–1897, 2011.
[8] Yong Fu and et al. Feedback thermal control of real-time systems on multicore processors. In *EMSOFT*, pages 113–122. ACM, 2012.
[9] Buyoung Yun and et al. Thermal-aware scheduling of critical applications using job migration and power-gating on multi-core chips. In *TrustCom*, pages 1083–1090. IEEE, 2011.
[10] Pradeep M Hettiarachchi and et al. Achieving thermal-resiliency for multicore hard-real-time systems. In *ECRTS*, pages 37–46. IEEE, 2013.
[11] Le Boudec and et al. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer, 2001.
[12] L. Thiele and et al. Real-time Calculus for Scheduling Hard Real-Time Systems. *ISCAS*, 4:101–104, 2000.
[13] Lothar Thiele and et al. Real-time interfaces for composing real-time systems. In *EMSOFT*, pages 34–43, 2006.
[14] Kai Huang and et al. Periodic power management schemes for real-time event streams. In *CDC/CCC*, pages 6224–6231. IEEE, 2009.
[15] Wei Huang and et al. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *VLSI*, 14(5):501–513, 2006.
[16] Lothar Thiele and et al. Predictability for timing and temperature in multiprocessor system-on-chip platforms. *TECS*, 12(1s):48, 2013.
[17] Vinay Hanumaiah and et al. Temperature-aware dvfs for hard real-time applications on multicore processors. *Computers*, 61(10):1484–1494, 2012.
[18] Pradeep M Hettiarachchi and et al. A design and analysis framework for thermal-resilient hard real-time systems. *TECS*, 13(5s):146, 2014.
[19] Srinivasan Murali and et al. Temperature-aware processor frequency assignment for mpsocs using convex optimization. In *CODES+ISSS*, pages 111–116, 2007.
[20] Santiago Pagani and et al. Tsp: thermal safe power: efficient power budgeting for many-core systems in dark silicon. In *CODES+ISSS*, page 10. ACM, 2014.
[21] Ravishankar Rao and et al. Fast and accurate prediction of the steady-state throughput of multicore processors under thermal constraints. *TCAD*, 28(10):1559–1572, 2009.
[22] Yongpan Liu and et al. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *DATE*, pages 1526–1531, 2007.
[23] Lars Schor and et al. Worst-case temperature guarantees for real-time applications on multi-core systems. In *RTAS*, pages 87–96. IEEE, 2012.
[24] Dante C Youla. Two observations regarding first-quadrant causal bibo-stable digital filters. *Proceedings of the IEEE*, 78(4):598–603, 1990.
[25] Dayan Adionel Guimaraes. *Digital Transmission: A Simulation-Aided Introduction with VisSim/Comm*. Springer Science & Business Media, 2010.
[26] Long Cheng and et al. Periodic thermal management for hard real-time systems. In *SIES*, pages 1–10. IEEE, 2015.
[27] John Howard and et al. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In *ISSCC*, pages 108–109. IEEE, 2010.
[28] MohammadSadegh Sadri and et al. Single-chip cloud computer thermal model. In *THERMINIC*, pages 1–6. IEEE, 2011.
[29] Hyunok Oh and Soonhoi Ha. Hardware-software cosynthesis of multi-mode multi-task embedded systems with real-time constraints. In *CODES*, pages 133–138. ACM, 2002.
[30] Hoeseok Yang and Soonhoi Ha. Pipelined data parallel task mapping/scheduling technique for mpsoc. In *DATE*, pages 69–74, 2009.