Received July 31, 2019, accepted August 11, 2019, date of publication August 14, 2019, date of current version August 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935339

Adaptive Periodic Thermal Management for Pipelined Hard Real-Time Systems

LONG CHENG[1], KAI HUANG[1], (Member, IEEE), GANG CHEN[1], ZHENSHAN BING[2], AND ALOIS C. KNOLL[2], (Senior Member, IEEE)
[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]Department of Informatics, Technical University of Munich, 85748 Munich, Germany

Corresponding author: Zhenshan Bing (bing@in.tum.de)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872393.

ABSTRACT This paper proposes a novel dynamic thermal management approach named Adaptive Periodic Thermal Management (APTM) for multi-core pipelined processors with hard real-time constraints. During online execution, the approach optimizes the peak temperature of the processor by dynamically calculating and deploying new adaptive periodic thermal management schemes, which periodically switch the stages to sleep state until next adapting. At each such instant of adapting, new APTM schemes which optimize the peak temperature under real-time constraints are calculated according to the extended pay-burst-only-once principle. To reduce the overhead of online computation, our approach utilizes thermal properties obtained from offline experiments in searching the ATPM scheme minimizing the temperature. We evaluate our APTM with simulation on the Intel SCC chip as well as real experiments on an Intel I7 processor. Compared to existing approaches, our approach reduces the peak temperature by up to 8 K for the Intel I7 processor, and 7.5 K for the Intel SCC using four to sixteen processor cores.

INDEX TERMS Pay-burst-only-once, real-time systems, temperature optimization, thermal management.

I. INTRODUCTION

Recently, multi-core architectures are widely adopted in modern processors due to their advantages in performance, power consumption, and implementation complexity over highly superscalar uniprocessor architectures. For example, Intel launched the 64-core Intel Xeon Phi processor 7210 in 2016. While shifting processors towards more parallelism, multi-core architectures raise a serious question: how to effectively extract and utilize the high degree of parallelism for applications. The pipelined architecture, which, in principle, can increase the throughput by performing multiple operations simultaneously, is believed to be one of the promising computing paradigms of multi-core systems [1]. A pipelined architecture consists of a set of processing components connected in series and thus achieves parallel computing by handling sub-tasks simultaneously on the components.

For real-time systems, the worst-case latency must be upper bounded for the correctness of the system. For this purpose, real-time system designers should consider an important factor, i.e., the temperature of the processor, which plays a key role in determining the allowable running speed of the processor [2]. Due to the increasing power density, the temperature of modern processors is consequently raised. Without proper thermal management policies, the high temperature may severely hamper the performance and reliability of the system. To reduce the temperature, a thermal management usually lowers the power consumption of processors, which means lower performance and longer latency. The trade-off between temperature and performance must be made carefully for real-time systems. Therefore, it's nontrivial to design a scheduling policy for real-time systems to effectively reduce the temperature and provide hard real-time guarantees.

This paper addresses such thermal optimization problem for pipelined hard real-time systems. Most previous related work either cannot provide hard real-time guarantees [3]–[6] or demands deterministic modeling of tasks [7]–[12], e.g., periodic task models. Long et al. proposed a thermal policy in [13] for hard real-time systems handling non-deterministic event arrivals. This approach is an offline one and considers the worst-case event arrivals and execution time of the applications to provide hard real-time guarantees. Thus, it could lead to pessimistic results due to the runtime

The associate editor coordinating the review of this article and approving it for publication was Cihun-Siyong Gong.

variability of event arrivals and execution times. Gang et al. proposed an adaptive online Dynamic Power Management approach called Balanced Workload Scheme (BWS) in [1] for hard real-time pipelined systems. However, BWS cannot be directly shifted to temperature optimization problem since it is designed for energy saving. Despite that temperature strongly depends on power, power management techniques that are effective for energy saving may not be suitable for temperature managing [14], which has already been theoretically proved by [15]. Moreover, the effectiveness of this approach strongly depends on the online adapting frequency. Therefore, to effectively optimize the temperature of pipelined real-time systems, new analysis and technique are required.

This paper studies how to minimize the peak temperature of pipelined multi-core systems under real-time constraints by adaptively applying dynamic power management during online execution. The concepts of arrival curve and service curve [16]–[18] are adopted as the workload and service model so that our approach can handle general event arrivals. The dynamic counter technique [19] is adopted to give a precise prediction of future event arrivals based on arrival history. The states of the FIFO (First In First Output) buffers between cores, which reflect actual event executions, are also collected and utilized in our approach. Combining them, we present the demanded service for unfinished and future events and then use it to calculate Adaptive Periodic Thermal Management (APTM) [20] schemes. To avoid the computation overhead of solving the thermal optimization problem online, we propose a heuristic scheme to compute APTM schemes according to the processor's unique thermal properties that are obtained from offline experiments. The contributions of this paper can be summarized as:

- We present a sufficient condition of guaranteeing deadline constraints of unfinished and future events for pipelined systems under APTM schemes. The condition can be utilized to derive APTM schemes that satisfy real-time constraints at instants of adapting.
- A set of online lightweight algorithms are presented to compute APTM schemes efficiently according to the unique thermal properties of the stages. The obtained APTM schemes can effectively reduce the peak temperature under real-time constraints for the pipelined system with negligible online overheads. Experimental results show that the average online overhead of these algorithms is only 275 us on an Intel I7-4770k processor.
- The effectiveness and efficiency of our proposed approach for reducing temperature are evaluated by simulations and experiments on real computer platforms. Two existing approaches are also implemented for comparison. For the simulation, Intel SCC is chosen and four up to sixteen processing cores are used. For real experiments, we implemented APTM on top of Linux with C++ using POSIX interface, using a Dell desktop equipped with an Intel I7-4770k processor. Experimental results show that the temperature can be reduced up to 8 K on the Intel I7 processor, and 7.5 K for the SCC processor running five cores. Compared to the adaptive approach BWS, our approach can better optimize the temperature with only a slightly increase in the adapting overhead.

The rest of this paper is organized as follows: The related work is briefly introduced in section II. Section III describes our system models and presents the problem statement. A motivation example is presented in Section IV. Section V presents backgrounds of our approach. Then, the real-time constraints are analyzed in Section VI. The heuristic scheme is discussed in Section VII and section VIII. Section IX presents the evaluation of APTM. Section X concludes this paper.

II. RELATED WORK

In this section, we briefly introduce related work on thermal management for hard real-time pipelined systems.

There have been several approaches on thermal optimization of pipelined systems. Cheng et al. presented an offline approach to minimize the peak temperature of pipelined systems under real-time constraints in [13]. The approach computes a PTM scheme for each stage to determine the switch on/off pattern during design phase. It searches the optimal solution that minimizes the peak temperature, as aforementioned, in offline manner, considering the worst-case execution time and event arrivals. Thus the results could be pessimistic due to the runtime variability of event arrivals and execution time. Cox et al. proposed a fast thermal-aware approach for streaming applications based on a 3D MPSoC model under the throughput constraints in [10]. The mapping of the multiple applications is determined at design time so that the peak temperature is minimized under throughput constraints. This approach assumes the periodic task model. Moreover, real-time constraints are not considered in this work. There are also several approaches on this topic. However, they either don't consider hard deadline constraints [3] or just reduce the deadline miss percentage into a low range [4]. Designed for hard real-time systems, our approach can guarantee that the worst-case delay is no larger than the deadlines of the tasks.

Chen et al. studied energy optimization by combining DVFS (Dynamic Voltage and Frequency Scaling) and DPM (Dynamic Power Management) technologies for real-time systems in [21]. They also explored how to apply DPM in an adaptive manner to optimize leakage power consumption for pipelined multi-core systems under deadline constraints in [1]. As aforementioned, an effective power management may not be suitable for thermal management. In addition, the proposed approach has one major drawback. The dynamic power management is based on the rate-latency pattern, that is, at each adapting instant, the approach only computes a sleep interval for one stage. The stage sleep for the corresponding time length, and then stay 'active' until next adapting instant. This simplifies the real-time analysis during online execution. However, this method demands a high adapting frequency, in other words, short adapting periods. Otherwise, the results worsen very quickly since the stages could be active unnecessarily between two adapting instants. In our approach, the stages are switched to sleep state periodically between two adapting instants so that the result is much less influenced by the adapting period. Therefore, our approach can still offer acceptable results with large adapting periods.

There has been significant work on thermal management of multi-core architectures. Wang *et al.* [11] addressed the problem of minimizing the peak temperature of a real-time application executed on multi-core platforms. Three computationally efficient algorithms are presented for deploying applications to individual devices. This approach assumes simple task models, i.e., periodic tasks model. The authors of [5] addressed the DVS scheduling problem for multi-core systems under both temperature and energy constraints. Since the problem is NP-hard, two algorithms, an exact one and an approximated one, are proposed to give results in different levels of accuracy. A stochastic thermal control approach is proposed in [6] to reduce the chip-wide temperature gradient of a multi-core processor handling multiple stochastic real-time task streams. The technique of job migration among active and passive cores of the stream is adopted to reduce the chip-wide temperature gradient. While making great contributions to the field, the above two approaches cannot provide hard real-time guarantees. Authors of [22] introduced an online DVFS management scheme for multi-core processors running hard real-time tasks. The technique in [19] is adopted to predict future event arrivals based on the arrival history. Although aiming at minimizing the temperature, the online DVFS approach doesn't consider any temperature feedback or thermal property of the cores, which can help to reduce the temperature further. In contrast, our approach utilizes the sampled temperature and the unique thermal properties of each core in determining scheduling decisions to obtain a lower peak temperature.

III. SYSTEM MODEL AND PROBLEM STATEMENT

Notation: In this paper, matrices and vectors are represented by bold characters. Moreover, time units millisecond and microsecond are denoted by ms and us, respectively.

A. HARDWARE MODEL

We consider a multi-core system with coarse-grained pipelined architecture which processes partitioned applications. The multi-core system could be heterogeneous or homogeneous. The sub-tasks of an application are mapped to and executed on different cores that communicate with each other via FIFO buffers. Without loss of generality, let n be the number of stages of the pipeline. We term the i^{th} stage as $\mathbb{S}_i$ in the following of this paper.

It is assumed that each core has three power consumption states, 'active', 'idle' and 'sleep'. To handle tasks, a core must stay at 'active' state. When having no task to handle, the core automatically switches from 'active' to 'idle' state for less power consumption. The core can also be switched off to stay at 'sleep' state to reduce the power more effectively. Note that the time overheads introduced by switching from 'active' or 'idle' states to 'sleep' state and switching back are considered and denoted as t^{swoff} and t^{swon}, respectively. Moreover, we assume the power consumption during mode-switching equals that in 'active' mode. It is worth noting that to compensate for the time overhead of switching, the sleep length t^{off} should be larger than t^{swoff}.

$$t^{off} > t^{swoff} \tag{1}$$

In runtime, our approach needs the temperature $\mathbf{T}(t)$ of the stages to make scheduling decisions. For processors with physical thermal sensors, the temperature can be obtained by reading the sensors. For processors without hardware thermal sensors on each core, we assume that soft thermal sensors [23] are employed to estimate the temperature.

In summary, this paper assumes that the hardware meets the following requirements.

- Each core has three power consumption states, 'active', 'idle' and 'sleep'.
- The processor can offer temperature readings, no matter from hardware sensors or predicted by programs.

B. TASK MODEL

In this paper, we consider applications which can be partitioned to sub-tasks. The end-to-end deadline of the application is denoted as D and the worst-case execution times (WCET) of the sub-tasks are represented by a vector $\mathbf{c} = [c_1, c_2, \ldots, c_n]$, where c_i is the WCET of the sub-task on stage $\mathbb{S}_i$.

In order to model general task arrivals with non-determinism, the arrival curve [16]–[18] is adopted in this work. The arrival curve $\alpha(\Delta)$ of a task is composed of two parts: the upper arrival curve $\alpha^u(\Delta)$ and the lower arrival curve $\alpha^l(\Delta)$.

$$\alpha^l(\Delta) \le R(t) - R(s) \le \alpha^u(\Delta), \quad \forall t - s = \Delta \tag{2}$$

where $R(t)$ is the cumulative workload function and denotes the number of tasks arriving in time interval $[0, t)$. Therefore, $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$ represent the maximal and minimal number of task-arrivals in any time interval with length Δ. For brevity, we specify such application by a tuple $\Phi = [D, \mathbf{c}, \alpha^u(\Delta), \alpha^l(\Delta)]$ in the following sections. Similarly, the concept of service curve $\bar{\beta}_i(\Delta) = [\bar{\beta}_i^u(\Delta), \bar{\beta}_i^l(\Delta)]$ is introduced to model the service provided by stage $\mathbb{S}_i$.

$$\bar{\beta}_i^l(\Delta) \le C_i(t) - C_i(s) \le \bar{\beta}_i^u(\Delta), \quad \forall t - s = \Delta \tag{3}$$

where cumulative function $C_i(t)$ denotes the amount of total time slots that stage $\mathbb{S}_i$ provides to handle tasks in time interval $[0, t)$.

Note that arrival curve $\alpha(\Delta)$ is event-based and denotes the number of tasks arriving in any time interval with length Δ whereas service curve $\bar{\beta}_i(\Delta)$ specifies the amount of time slots that provided by stage $\mathbb{S}_i$ in any time interval with length Δ. To perform operations between the arrival curve and the service curve, we transform time-based $\bar{\beta}_i(\Delta)$ to an event-based one:

$$\beta_i(\Delta) = \lfloor \bar{\beta}_i(\Delta)/c_i \rfloor. \tag{4}$$

where $\beta_i(\Delta)$ denotes the number of events that can be handled by stage $\mathbb{S}_i$ in any time interval with length Δ. Then, according to the existing results in [16] and [17], a system with aggregate service curve $\beta(\Delta)$ can satisfy the deadline constraint for an application Φ, if the following condition holds.

$$\beta^l(\Delta) \geq \alpha^u(\Delta - D)$$

C. ADAPTIVE PERIODIC THERMAL MANAGEMENT

In this paper, we study how to minimize the peak temperature for coarse-grained pipelined multi-core processors. An online approach named Adaptive Periodic Thermal Management is proposed to adaptively switch the cores to 'sleep' state in the run time.

At each adapting instant, an APTM scheme is applied to each core in the pipeline and is updated at the next adapting instant. The APTM scheme applied to stage $\mathbb{S}_i$ is specified by a pair of parameters (t_i^{on}, t_i^{off}). The period is calculated as $t_i^{on} + t_i^{off}$. For brevity, $\mathbf{t^{on}}$ and $\mathbf{t^{off}}$ denote the vectors $[t_1^{on}, t_2^{on}, \ldots, t_n^{on}]$ and $[t_1^{off}, t_2^{off}, \ldots, t_n^{off}]$, respectively. Given an APTM scheme, whichever state it is currently at, stage $\mathbb{S}_i$ first switches to 'sleep' state and stays for t_i^{off} time units. Then, it switches to 'active' or 'idle' state and keeps for t_i^{on} time units. This procedure repeats until the next adapting instant. An example of APTM scheme is demonstrated in Fig. 1. Note that due to the switching overhead, the valid/invalid time interval in each period for handling workload should be revised as:

$$t^{vld} = t^{on} - t^{swon}, \tag{5}$$
$$t^{inv} = t^{off} + t^{swon}. \tag{6}$$

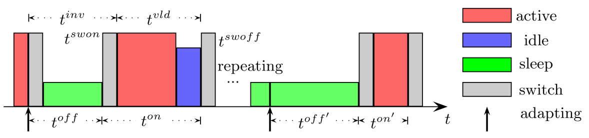


FIGURE 1. The adaptive periodic thermal management schemes after two adapting instants.

In addition, we define the valid partition as:

$$K = t^{vld}/(t^{on} + t^{off}). \tag{7}$$

D. PROBLEM STATEMENT

At each online adapting instant, our approach should offer an APTM (t_i^{on}, t_i^{off}) scheme to each stage $\mathbb{S}_i$ in the pipelined system. The values of t_i^{on} and t_i^{off} should be chosen prudently. First, it's clear that the peak temperature can be reduced by increasing the sleep interval or decreasing the active interval. However, this action should be done carefully, otherwise the deadline constraint of current or future events may be violated. Moreover, the sleep intervals for all stages are not independent with each other due to deadline constraints. The influence of such interaction between two stages on the temperature distribution should be handled carefully. Finally, increasing the sleep interval length under real-time constraints causes the active interval to increase simultaneously. Thus, setting a large t_i^{off} results in a large t_i^{on}, which may raise the peak temperature. Therefore, at each adapting instant, for each stage, our approach should: (1) calculate the real-time constraints to $\mathbf{t^{on}}$ and $\mathbf{t^{off}}$; (2) properly select and balance t_i^{on} and t_i^{off} for all stages under the real-time constraints so that the peak temperature can be minimized.

In conclusion, the problem can be formally defined as: *For a given n-stage pipelined platform modeled in Section III, a real-time application specified by* Φ*, at each adapting instant during runtime we should determine the APTM schemes applied to all stages so that the peak peak temperature of the processor is minimized and the worst-case end-to-end delay of any task is less than the deadline D.*

IV. MOTIVATION OF OUR WORK

In this section, the motivation of our approach is presented by comparing it with an offline approach in a concrete example.

In contrast to the approach Offline Pay-Burst-Only-Once (O-PBOO) in [13], which searches the optimal $(\mathbf{t^{off}}, \mathbf{t^{on}})$ in offline manner and then applies them to the system, APTM adaptively applies $\mathbf{t^{on}}$ and $\mathbf{t^{off}}$ to the stages according to the current workload and the states of processors. Compared to O-PBOO, APTM utilizes the following two facts to reduce the temperature.

Dynamic slack An offline approach analyzes the pipelined system by considering the worst-case pattern of event arrivals, i.e., the upper bound $\alpha^u(\Delta)$ and the lower bound $\alpha^l(\Delta)$, which represent the maximal and minimal number of event arrivals in any time interval with length Δ, respectively. However, the worst-case scenarios rarely happen. In reality, events arrive with a variable delay bounded by the arrival curve $\alpha(\Delta)$. For example, consider a task specified by period $p = 50$ ms and jitter $j = 50$ ms. The worst-case event arrival is a burst with two events released at the same time. However, in reality, for example, events may arrive at time [0, 40, 110, 150, 210, 230] ms. We term the difference between the worst-case and real event arrivals as the dynamic slack.

Execution slack When performing offline analysis, static approaches used the worst-case execution time (WCET) to bound the execution time of a task on the target system. However, due to the inherent variability of execution time, most events, in reality, finishes before their WCETs. The fact is termed as execution slack, which occurs as a result of the difference between the real execution time and WCET of an event.

By recording the history of event arrivals and monitoring the filling level of FIFOs between stages, APTM can effectively utilize the above slacks and achieve significant temperature reduction compared to offline approaches. Now, we show the advantage of APTM by a concrete example.

In the example, O-PBOO and APTM are applied to manage the temperature of a pipelined system with three stages. The task is specified by period $p = 50$ ms and jitter $j = 50$ ms. The worst-case execution times on all stages are $\mathbf{c} = [4, 4, 4]$ ms. We assume the deadline of the task equals its period. Events are released at time [0, 0, 50, 100, 150, 200, 250] ms. The real execution times of the events on three stages are set as the random numbers between 1.6 ms and 4 ms. The adapting periods of APTM is set as 50 ms.

We run the two approaches for the task trace and obtain the temperature evolutions. The temperature of the first stage is depicted in Fig. 2. For a clean figure, we consider the power dissipations in 'idle' state and 'active' are the same in the simulation. Since we only want to compare the relative performance of the two approaches, this simplification is acceptable. As displayed, APTM offers the lowest temperature after three adapts at time $t = 100$ ms. The higher temperature given by O-PBOO is caused by considering the worst-case task timing parameters in the design phase. In other words, O-PBOO must ensure the end-to-end deadlines are met with the assumption that all events require WCET to finish and the event arrival burst may happen at anytime during real execution. We also extend the task trace to 60 seconds and run both approaches again. The final peak temperatures of O-PBOO and APTM are 391 K and 375 K, respectively. This further strengthens our observation.

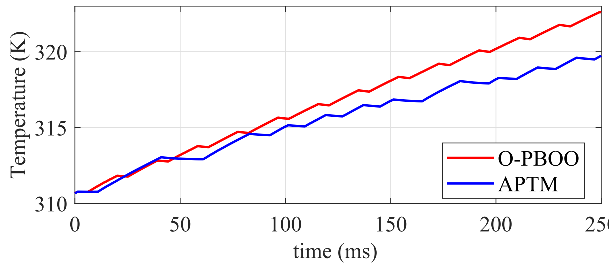


FIGURE 2. The temperature of the first core in the ARM 3-stage platform when the two methods are applied to manage it.

In summary, utilizing and managing the aforementioned slacks, APTM can significantly reduce the peak temperature, which is the main motivation of our work. In the next section, we discuss how the dynamic slack and execution slack are explicitly utilized and managed by our approach.

V. UTILIZING THE TWO SLACKS

In Section IV, we investigated the facts of dynamic slack and execution slack. In this section, we present how to effectively utilize them in APTM.

A. DEMANDED SERVICE OF UNFINISHED EVENTS

Our approach utilizes execution slack by monitoring the numbers of unfinished events in the FIFOs as well as the states of the ongoing job in each stage. With the obtained dynamic information, we get the demanded service of unfinished events.

Unfinished events could be stored in FIFOs or the core processing it at an adapting instant. Let $E_i(t)$ represent the union set of events stored in $FIFO_i$ and the core on stage $\mathbb{S}_i$ at time t. The number of events in $E_i(t)$ is denoted as $|E_i(t)|$. It is worth noting that, although the absolute deadlines for events in $E_i(t)$ do not change, their relative deadlines should be revised according to the relative distances between absolute deadlines and time t. Assume events in $E_i(t)$ are ordered from the earliest absolute deadline to the latest. Then the event stored in the core should be the front of the queue. Moreover, its unfinished part can be upper bounded by:

$$\delta_i(t) = \frac{c_i - t_i^{exe}(t)}{c_i}. \tag{8}$$

where c_i is the WCET of the job on stage $\mathbb{S}_i$ and $t_i^{exe}(t)$ is the time length for which the event has already been executed. We represent the demanded service curve of set $E_i(t)$ by notation $\alpha_i^d(t, \Delta)$, which denotes the event-based service demanded by $E_i(t)$ in time interval $[t, t + \Delta]$ to meet their deadlines, which can be given as:

$$\alpha_i^d(t, \Delta) = \begin{cases} j-1+\delta_i(t), & \text{if } D_{i,j} \le t+\Delta \le D_{i,j+1} \\ |E_i(t)|-1+\delta_i(t), & \text{if } t+\Delta > D_{i,|E_i(t)|}, \end{cases} \tag{9}$$

where $D_{i,j}$ is the absolute deadline of the j^{th} event in $E_i(t)$. Note that the value of $t_i^{exe}(t)$ can be obtained from the execution time monitoring ability of the system. Such functionality is already commonly available on many real-time platforms [24].

B. ARRIVAL CURVE OF FUTURE EVENTS

To effectively exploit the dynamic slack, the dynamic counter proposed in [19] is adopted to predict future event arrivals. According to [19], at time t, the number of event arrivals in time interval $[t, t + \Delta]$ can be bounded by $\alpha^f(t, \Delta)$, which is calculated based on the original arrival curve $\alpha(\Delta)$ and the event arrival history upon t. It's worth noting that dynamic counter can be easily implemented as part of the hardware with negligible overhead [19], [25].

VI. PROPOSED APPROACH

In this section, we present our adaptive periodic thermal management approach to reduce the peak temperature of pipelined multi-core processors. At one adapting instant for decisions, APTM should compute $\mathbf{t^{on}}$ and $\mathbf{t^{off}}$ that guarantee the timing constraints of unfinished events as well as future events. To model the system containing unfinished events, we first transform it into a multi-stream system in which the unfinished events in each stage are modeled as a stream (Section VI-A). With the transformed system, the end-to-end

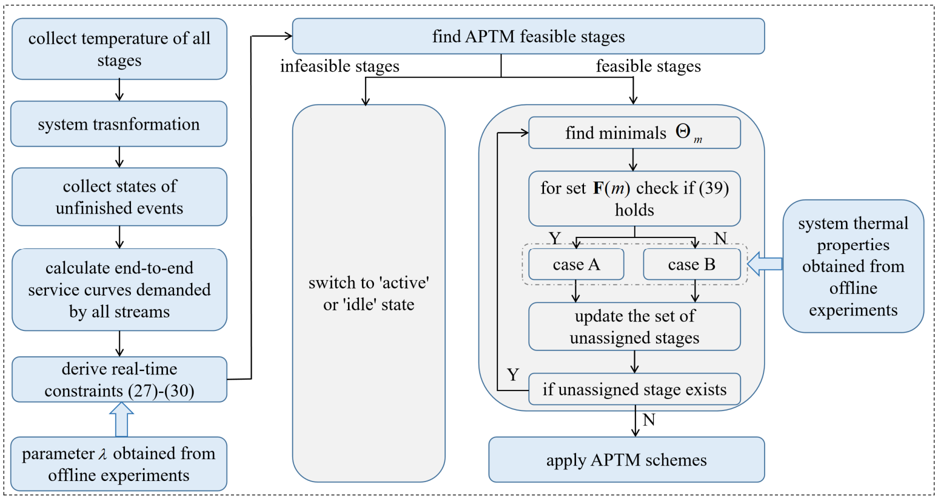

collect temperature of all stages
system trasnformation
collect states of unfinished events
calculate end-to-end service curves demanded by all streams
derive real-time constraints (27)-(30)
parameter λ obtained from offline experiments
find APTM feasible stages
infeasible stages
feasible stages
switch to 'active' or 'idle' state
find minimals Θ_m
for set F(m) check if (39) holds
Y
N
case A
case B
update the set of unassigned stages
Y
if unassigned stage exists
N
system thermal properties obtained from offline experiments
apply APTM schemes

FIGURE 3. The overall procedure of our approach at each adapting instant to assign APTM schemes to the stages.

service provided to each stream by the system under APTM can be calculated based on the extended pay-burst-only-once principle (Section VI-B). Then a set of constraints is applied to $\mathbf{t^{on}}$ and $\mathbf{t^{off}}$ to satisfy the deadlines of unfinished events as well as future events (Section VI-C). Finally, we present a heuristic scheme to solve the thermal optimization problem. Several lightweight algorithms are proposed to compute $\mathbf{t^{on}}$ and $\mathbf{t^{off}}$ which minimize the peak temperature of the pipelined system (Section VII). The overall procedure of our approach is depicted in Fig. 3, which is discussed in details in the following sections.

A. SYSTEM TRANSFORMATION

At each adapting instant, our approach must offer APTM schemes satisfying the end-to-end deadline constraints of new arriving events and the unfinished events stored in the system. For an offline approach, the Pay-Burst-Only-Once principle can be adopted in real-time analysis. However, as pointed in [26], the unfinished events prevent our approach from utilizing the Pay-Burst-Only-Once principle directly during online adapting. As already demonstrated in [13], [27], approaches without using Pay-Burst-Only-Once account for the burst in the original stream more than once and requires lots of computation, thus leading to pessimistic results. In this paper, we transform the system into an alternative formation with which the time properties of the system can be analyzed effectively.

For an n-stage pipelined multi-core system, the transformation is done by modeling the unfinished events $E_i(t)$ in stage $\mathbb{S}_i$ as an individual input stream S_i with demand curve $\alpha_i^d(t, \Delta)$ passing through the following stages, indexed by $J_i = \{i, i+1, \ldots, n\}$. Specifically, the input stream of the first stage comprises the new arriving events and unfinished set $E_1(t)$. An example of transformation is shown in Fig. 4. Moreover, let set $J_{i,k}(k > i) = \{k, k+1, \ldots, n\}$ represent all the stages that are traversed both by stream S_i and S_k. Finally,

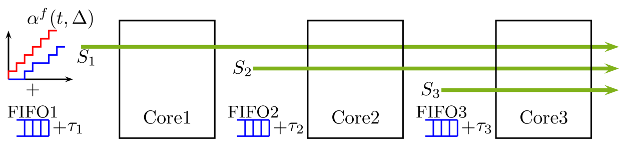


FIGURE 4. An example of the transformation of a 3-stage pipelined multi-core system.

notation $\digamma_i = \{i+1, \ldots, n\}$ denotes the indexes of streams that are interferenced by stream S_i. For instance, considering the example in Fig. 4, we have $J_1 = \{1, 2, 3\}$, $J_{1,2} = \{2, 3\}$ and $\digamma_1 = \{2, 3\}$.

B. REAL-TIME CONSTRAINTS

Now, we analyze how to guarantee the deadline requirements for unfinished as well as future events at each adapting instant. It's worth noting that the service-providing-ability of the pipelined system is directly determined by t^{vld}s and t^{inv}s, instead of t^{on}s and t^{off}s. Thus, we use t^{vld}s and t^{inv}s in following analysis and then retrieve back t^{off}s and t^{on}s according to (5) and (6).

For the aforementioned system transformation, the EPBOO principle in [26] and the theory in [1] can calculate the aggregate service curve provided for stream S_i for scenarios in which every stage provides a service curve in *rate-latency* format. However, under APTM, the service curve of each stage is no longer rate-latency. In addition, the time overhead of mode-switching is also not considered in their work. Therefore, the results in [1], [26] cannot be utilized in our work. In this section, we present a new theorem which lower bounds the end-to-end service curve for stream S_i in the transformed system under APTM.

Lem. 1: Given an n-stage pipelined multi-core system modeled in Section III. The system can be viewed as an n-stream system according to the aforementioned system transformation. Suppose APTM schemes specified by ($\mathbf{t^{on}}$, $\mathbf{t^{off}}$) are applied to the stages. Let β_i^{ete} represent the end-to-end service curve provided by the system to stream S_i. If the valid time length in each period t_i^{vld} is positive integer times of c_i, then we have

$$\beta_i^{ete}(\Delta) \geq \left[\kappa_i(\Delta - \sum_{j=i}^{n}(t_j^{inv}+c_j) - \sum_{j=i+1}^{n}\frac{W_j+\delta_j(t)}{\kappa_j})\right]^+, \quad (10)$$

where W_i denotes the number of events stored in *FIFO*$_i$, $\delta_i(t)$ is calculated from (8), and

$$[x]^+ = \max(0, \ x)$$
$$\kappa_i = \min_{j \in J_i}(K_i/c_i).$$

Proof: From the existing result presented in [26],[1] one can obtain (11) for the stream of interest S_i with any time instants satisfying $t_1 \leq t_2 \leq \cdots t_{n+1}$:

$$R_i^{n+1}(t_{n+1}) - R_i^1(t_1) \geq \sum_{j \in J_i}\beta_j^l(t_{j+1}-t_j) - \sum_{k \in \digamma_i}\sum_{j \in J_{i,k}}(R_k^{j+1}(t_{j+1}) - R_k^j(t_j)), \quad (11)$$

where $R_i^j(t)$ is the workload function of S_i at j^{th} stage and denotes the number of events in stream S_i that arrive at the j^{th} stage in time interval $[0, t)$. Specifically, $R_i^{n+1}(t)$ is the workload function of the output of stream S_i.

From the definition of $\digamma_i$ and $J_{i,k}$, one can easily determine that:

$$\sum_{j \in J_{i,k}}(R_k^{j+1}(t_{j+1}) - R_k^j(t_j)) = R_k^{n+1}(t_{n+1}). - R_k^k(t_k) \quad (12)$$

At each adapting instant, the right hand side of (12) can be safely bounded by the number of unfinished events in *FIFO*$_k$ and $\mathbb{S}_k$, i.e., $W_k(t)+\delta_k(t)$ as discussed in section V-A. Therefore,

$$\sum_{j \in J_{i,k}}(R_k^{j+1}(t_{j+1}) - R_k^j(t_j)) \leq W_k(t) + \delta_k(t). \quad (13)$$

When t_i^{vld} is positive integer times of c_i, the stair-case service curve $\beta_i^l(\Delta)$ of APTM can be lower bounded by a bounded-delay function [28]:

$$\beta_i^l(\Delta) \geq \max(0, \ \frac{K_i}{c_i}(\Delta - (t_i^{inv} + c_i))), \quad (14)$$

i.e.,

$$\beta_i^l(\Delta) \geq [\frac{K_i}{c_i}(\Delta - (t_i^{inv} + c_i))]^+ \quad (15)$$

Let $bdf_i(\Delta)$ denote bounded-delay function $[K_i/c_i(\Delta - (t_i^{inv} + c_i))]^+$. Then, combining the definition of $\digamma_i$, equations (11), (13) and (15), we finally have:

$$R_i^{n+1}(t_{n+1}) - R_i^1(t_1) \geq \sum_{j \in J_i} bdf_j(t_{j+1}-t_j) - \sum_{k=i+1}^{n}(W_k(t)+\delta_k(t)). \quad (16)$$

[1]See equation (14) in [26].

The bounded-delay function $bdf_i(\Delta)$ is actually in the rate-latency format. Therefore, following the derivation similar to that in [26], one can finally have inequality (10). □

Lem. 1 offers a lower bound of service curve *provided to* stream S_i if APTM schemes specified by ($\mathbf{t^{on}}$, $\mathbf{t^{off}}$) are applied. Next, we will discuss the service curve *demanded by* S_i to meet deadline constraints.

Lem. 2: Given the n-stream system obtained from system transformation, the end-to-end service curve demanded by stream S_i to meet the corresponding deadline constraints can be given as:

$$\beta_i^{dmd}(\Delta) = \begin{cases} \alpha^f(t, \Delta - D) + \alpha_1^d(t, \Delta), & \text{if } i = 1 \\ \alpha_i^d(t, \Delta), & \text{if } i \geq 2. \end{cases} \quad (17)$$

Proof: For the case that $i \geq 2$, it's already proved in section V-A. When $i = 1$, the input workload comprises the new arriving events as well as the unfinished events stored in the first *FIFO* and stage. As studied in section V-B, the new arriving tasks can be tightly and safely bounded by the arrival curve $\alpha^f(t, \Delta)$. Right shifting $\alpha^f(t, \Delta)$ by deadline D yields the demanded service curve of new arriving tasks. Then, adding it with $\alpha_1^d(t, \Delta)$ gives the demanded service curve for first stage. □

With Lem. 1 and Lem. 2, the sufficient condition for satisfying deadline constraints for all tasks under Adaptive Periodic Thermal Management is given as the following theorem.

Thm. 1: Given the same n-stage pipelined system in Lem. 1, suppose APTM schemes specified by ($\mathbf{t^{on}}$, $\mathbf{t^{off}}$) are applied to the system. Then, the worst-case end-to-end delay of any task is guaranteed to be no larger than its deadline D if the following conditions hold for any stream S_i.

$$\forall i \leq j \leq n, \quad t_j^{vld} = g_j \cdot c_j, \ g_j \in \mathbb{N} \quad (18)$$

$$\kappa_i(\Delta - \sum_{j=i}^{n}(t_j^{inv} + c_j) - \sum_{j=i+1}^{n}\frac{W_j + \delta_j(t)}{\kappa_j}) \geq \beta_i^{dmd}(\Delta) \quad (19)$$

Proof: It is clear that the deadline constraints can be met if the end-to-end service curve is β_i^{ete} no less than the demanded service curve for every stream S_i, i.e., $\beta_i^{ete}(\Delta) \geq \beta_i^{dmd}(\Delta)$. Let $\beta_i^{\diamond}$ denote the left side of (19). From Lem. 1, once conditions (18) and (19) are satisfied, we have $\beta_i^{ete}(\Delta) \geq \beta_i^{\diamond} \geq \beta_i^{dmd}(\Delta)$. Therefore, the end-to-end delay of any event can be guaranteed to be no larger than its deadline D. □

C. OPTIMIZATION PROBLEM AT ADAPTING INSTANTS

We have presented the real-time constraints for n-stage pipeline systems in Thm. 1. At every adapting instant, our approach should provide a set of APTM schemes which minimizes the peak temperature under these constraints. To improve the efficiency of the online searching algorithms, we discuss how to transform and simplify the constraints in Thm. 1 in this section.

Let's examine condition (19) closely. The left side of (19) is actually a bounded-delay-function $bdf(\Delta) = [\rho_i(\Delta - b_i)]^+$.

One can easily see that given b_i, the minimal slope ρ_i satisfying (19) can be determined by:

$$\rho_i = \min\{\rho|[\rho(\Delta - b_i)]^+ \geq \beta_i^{dmd}(\Delta)\}. \tag{20}$$

It's worth noting that ρ_i can be obtained efficiently by implementing a binary search. For a pair of b_i and ρ_i obtained from (20), it is intuitive that condition (19) is guaranteed if the following two constraints hold simultaneously [13]:

$$\kappa_i = \min_{j=i}^{n}(K_i/c_i) \geq \rho_i, \tag{21}$$

$$\sum_{j=i}^{n}(t_j^{inv} + c_j) + \sum_{j=i+1}^{n}\frac{W_i + \delta_j(t)}{\kappa_j} \leq b_i. \tag{22}$$

Now, the problem is how to determine b_i. In fact, b_i can be any real number between b_i^{min} and b_i^{max}, which are the lower and uuper bounds of b_i for curve $\beta_i^{dmd}(\Delta)$, respectively.

$$b_i^{max} = \max\{b| \frac{1}{\max_{j=i}^{n}(c_j)}(\Delta - b) \geq \beta_i^{dmd}(\Delta)\} \tag{23}$$

$$b_i^{min} = \sum_{j=i}^{n} c_j + \sum_{j=i+1}^{n}\{(\max_{k=j}^{n} c_j)(W_j + \delta_j(t))\} \tag{24}$$

Searching for the optimal b_i in its feasible region incurs significant computing overhead, which hampers the effectiveness and efficiency of our approach. Therefore, in this paper, we obtain a sub-optimal b_i by:

$$\tilde{b}_i = \lambda b_i^{max} + (1-\lambda)b_i^{min}, \tag{25}$$

where λ is a real number in range $[0,\ 1]$, The key parameter λ can be determined by offline simulation and set as the one yielding the lowest peak temperature, which is described in algorithm 5. Although this method doesn't offer the global optimal solution, it introduces negligible overhead during online adapting and thus is suitable for our approach. Then, constraint (22) is revised as $\sum_{j\in J_i} t_j^{inv} \leq b_i - (\sum_{j=i}^{n} c_j + \sum_{j=i+1}^{n}\frac{W_i+\delta_j(t)}{\rho_j})$. For brevity, let $\bar{b}_i$ denote $b_i - (\sum_{j=i}^{n} c_j + \sum_{j=i+1}^{n}\frac{W_i+\delta_j(t)}{\rho_j})$, and we can replace (22) by:

$$\sum_{j\in J_i} t_j^{inv} \leq \bar{b}_i. \tag{26}$$

Finally, the corresponding slope for $\tilde{b}_i$ calculated according to (20) is termed as $\tilde{\rho}_i$.

Combining the real-time constraints of all streams and the hardware constraints (1), we format our problem as follows.

$$\begin{aligned}
\text{minimize}\quad & T^\star(\mathbf{t^{on}}, \mathbf{t^{off}}) = \max_{t\geq t^a}\{\max \mathbf{T}(t)\} \\
\text{subject to}\quad & t_i^{vld} = g_i \cdot c_i, \quad g_i \in \mathbb{N},\ \forall i \in [1, \ldots, n] \quad (27)\\
& t_i^{inv} \geq t_i^{swon} + t_i^{swoff}, \quad \forall i \in [1, \ldots, n] \quad (28)\\
& K_i \geq \max_{j=1}^{i}(\tilde{\rho}_j)c_i, \quad \forall i \in [1, \ldots, n] \quad (29)\\
& \sum_{j\in J_i} t_j^{inv} \leq \bar{b}_i,\ \forall i \in [1, \ldots, n] \quad (30)
\end{aligned}$$

where t^a denotes the time instant of adapting.

The peak temperature $T^\star$ can be computed by the algorithm proposed in [13]. However, solving such an optimization problem requires significant computational effort due to two reasons. First, calculating $T^\star$ incurs the costly convolution operation as we consider heat influence between two blocks in the thermal model. Second, even with an efficient searching algorithm, exploring the n-dimension search space is still heavy in computing. Therefore, it is infeasible to search for the accurate solution of such problem with online adapting. To address this issue, one lightweight heuristic scheme is proposed in the next section to offer sub-optimal APTM schemes under hard real-time constraints.

VII. ONLINE PART

In this section, we present one heuristic scheme that determines APTM schemes online by following the guidance of processor thermal properties obtained from offline experiments. In this way, the online computation of exploring search space is significantly saved. The problem is solved in two steps. First, our approach decides which stages are feasible for APTM schemes. Then, the t^{inv}s and t^{vld}s of these stages are determined according to the thermal properties obtained in offline experiments.

A. FEASIBLE STAGES FOR APTM

At an adapting instant, it's possible that some stages cannot adopt APTM schemes, otherwise the real-time constraints will be violated. The reason is that hardware-constraint (28) may conflict with real-time constraint (30). For example, consider a 4-stage pipelined system. Assume constraint (30) for stream S_3 is already given as $t_3^{inv} + t_4^{inv} \leq 1.5$. Suppose the time-overheads of mode-switching, t^{swon} and t^{swoff}, are all 0.5 ms. Therefore, according to (28), we have: $t_3^{inv} + t_4^{inv} > 2$ ms, which conflicts with $t_3^{inv} + t_4^{inv} \leq 1.5$. Then, it is impossible to find APTM schemes for both stages $\mathbb{S}_3$ and $\mathbb{S}_4$ simultaneously. In this case, at least one or more stages should always be 'active' or 'idle' state to meet constraints (28) and constraint (30). We call such stages the infeasible stages. Other stages that can adopt APTM schemes are called feasible stages and will stay at 'active' or 'idle' state until next adapting instant.

We propose algorithm 1 to find feasible stages at an adapting instant. It's clear that adopting an APTM scheme outperforms always staying at 'active' or 'idle' state in lowering temperature. Therefore, our approach assigns a higher priority to the stage having higher temperature in adopting an APTM scheme. Basically, this algorithm follows two principles: (1) try to put as many stages to feasible stages as possible, (2) a stage having higher temperature is assigned a higher priority in adopting an APTM scheme. In this way, we can balance the temperature between stages and thus reduce the peak temperature.

Algorithm 1 takes the vector of current temperatures $\mathbf{T}(t)$ of all stages as input. It returns two sets of indexes: $\mathbf{F}$ stands for the indexes of feasible stages and $\mathbf{A}$ for infeasible stages. We first sort the temperatures of stages in the

Algorithm 1 Determine APTM-Feasible Stages

```
Input: T
Output: F, A
1: $\mathbf{F} \leftarrow \emptyset$, $\mathbf{A} \leftarrow \emptyset$
2: sort $\mathbf{T}$ in descending order, save the sorted indexes in $\mathbf{I}$
3: for $i$ in vector $\mathbf{I}$ do
4:     $t_i^{inv} \leftarrow t_i^{swon} + t_j^{swoff}$
5:     if $\sum_{j\in J_k} t_j^{inv} \le \bar{b}_k$ holds $\forall k \in \{1,\ldots,i\}$ then
6:         $\mathbf{F} \leftarrow \mathbf{F} \bigcup i$
7:     else
8:         $\mathbf{A} \leftarrow \mathbf{A} \bigcup i$
9:         $t_i^{inv} \leftarrow (\mathbb{S}_i$ is 'sleep' ? $t_i^{swon}$ : $0)$
10:    end if
11: end for
```

descending order and get the sorted index vector (line 2). Then, from the stage with the highest temperature, the hardware constraint (28) is checked with respect to the real-time constraint (30) of previous stages to determine if the stage is APTM-feasible (lines 4 and 5). Since the first checked stage has the highest priority in adopting APTM, the current stage is put into set **F** if there is no conflict (line 6). If there is a conflict, the stage certainly cannot adopt an APTM scheme and thus is an infeasible stage (line 8). Note that the corresponding t_i^{inv} should be set as t_i^{swon} to comprise the switching overhead, otherwise set as zero (line 9).

Then, constraint (30) in APTM constraint set of stage $\mathbb{S}_i$ is revised as:

$$\sum_{j\in J_i \bigcap \mathbf{F}} t_j^{inv} \le \bar{b}_i - \sum_{j\in J_i \bigcap \mathbf{A}} t_j^{inv}. \tag{31}$$

Note that for different stages, the left side of constraint (31) can be same with each other. For example, consider a 3-stage system, and $\mathbf{F} = \{1, 3\}$ and $\mathbf{A} = \{2\}$. Then we can simplify the total three inequalities into two inequalities: $t_1^{inv} + t_3^{inv} \le \bar{b}_1 - t_2^{inv}$ and $t_3^{inv} \le \min(\bar{b}_2 - t_2^{inv}, \bar{b}_3)$. In conclusion, constraints (31) for all the n stages can be simplified into n_f inequalities:

$$\sum_{j\in\mathbf{F}(k)} t_j^{inv} \le \Theta_k, \quad \forall k = 1, 2, \ldots, n_f, \tag{32}$$

where integer n_f represents the number of feasible stages, $\mathbf{F}(k)$ denotes the max k elements in the set $\mathbf{F}$, and constant Θ_k is obtained from merging right side of (31) of different stages by `min` operation. For the above example, we have:

$$\begin{aligned} n_f &= 2, \\ \mathbf{F}(1) &= \{3\}, \\ \mathbf{F}(2) &= \{1, 3\}, \\ \Theta_1 &= \min(\bar{b}_2 - t_2^{inv}, \bar{b}_3), \\ \Theta_2 &= \bar{b}_1 - t_2^{inv}. \end{aligned}$$

Finally, constraint (30) in APTM constraint set can be replaced by the following constraint for feasible stages.

$$\sum_{j\in\mathbf{F}(k)} t_j^{inv} \le \Theta_k, \quad \forall k = 1, \ldots, n_f \tag{33}$$

B. APTM SCHEMES FOR APTM-FEASIBLE STAGES

Now, we only need to consider real-time constraints for feasible stages.

One can observe that (33) comprises n_f inequalities and each of them involves the sleep intervals of k stages. This indicates, as mentioned in Section III, the sleep intervals of the stages can be influenced by each other. Computing the APTM schemes for all stages in **F** simultaneously is time-consuming and complicated to analyze. To simplify the problem, we calculate APTM schemes in a recursive manner. The algorithm is described as follows.

The pseudo code is depicted in algorithm 2. Each iteration, we first find the minimal value, Θ_m, in the set of Θ for the stages in **F** (line 2). Then we determine APTM schemes only for the stages in $\mathbf{F}(m)$ (line 3). Since Θ_m is the minimal value, the other constraints in (33) are certainly satisfied and thus can be safely ignored. The final step at each iteration is removing already determined stages from **F** (line 4). The iteration continues until there is not stages in **F** (line 1).

Algorithm 2 Assign APTM for all APTM-Feasible Stages

```
Input: $\mathbf{F}$, $\mathbf{K^m} = \{K_i^m | i \in \mathbf{F}\}$, $\Theta = \{\Theta_1, \ldots, \Theta_{n_f}\}$
Output: APTM schemes
1: while $\mathbf{F} \ne \emptyset$ do
2:     find $m$ that $\Theta_m = \min(\Theta)$
3:     assign APTM schemes for stages in $\mathbf{F}(m)$.
4:     $\mathbf{F} \leftarrow \mathbf{F} \setminus \mathbf{F}(m)$, $\Theta \leftarrow \Theta \setminus \{\Theta_1, \ldots, \Theta_k\}$
5: end while
```

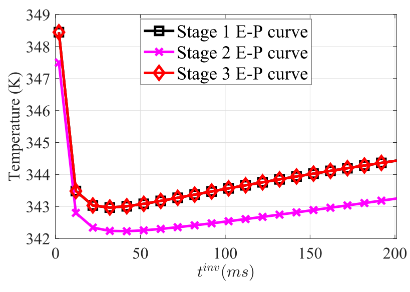
This algorithm illustrates the big picture of how we determine APTM schemes for feasible stages. In the next section, we will expand the details of line 3.

C. ASSIGN APTM SCHEMES FOR STAGES IN F(M)

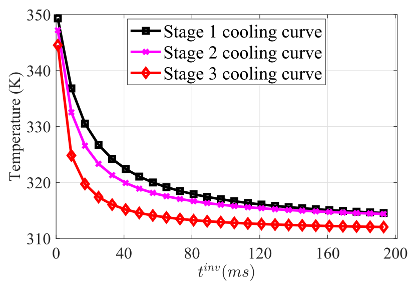
Let **G** denote the set of stages in $\mathbf{F}(m)$ at each step of line 3 in algorithm 2. Now, we investigate how to assign t^{inv}s and t^{vld}s for stages in **G**. The problem can be formatted as follows.

$$\begin{aligned} \text{minimize} \quad & T^\star(\mathbf{t^{on}}, \mathbf{t^{off}}) = \max\{\max \mathbf{T}(t)\} \\ \text{subject to} \quad & \sum_{i\in\mathbf{G}} t_i^{inv} \le \Theta_m && (34) \\ & t_i^{vld} = g_i \cdot c_i, \quad g_i \in \mathbb{N}, \ \forall i \in \mathbf{G} && (35) \\ & K_i \ge \max_{j=1}^{i}(\tilde{\rho}_j)c_i, \quad \forall i \in \mathbf{G} && (36) \\ & t_i^{inv} \ge t_i^{swon} + t_i^{swoff}, \quad \forall i \in \mathbf{G} && (37) \end{aligned}$$

According to constraint (35), the valid time length t_i^{vld} should be integer times of c_i: $t_i^{vld} = g_i c_i$. Suppose all the g_i are already known, then, it's intuitive that the ideal



(a) *E-P curve*



(b) *cooling curve*

FIGURE 5. **The *E-P curves* and *cooling curves* of a 3-stage pipelined system. (a) The valid partition of all the stages are set as $K_i = 0.75$. (b) The t^{vld}s of the stages are {15.2, 10, 4.6} ms.**

solution for the corresponding t_i^{inv} is $\frac{t_i^{vld}}{\max_{j=1}^{i}(\tilde{\rho}_j)c_i}/ - t_i^{vld}$, since this makes the valid partition K_i equal its lower bound, which means the lowest peak temperature for the same t_i^{vld}. However, the t_i^{inv}s obtained by this method must meet the real-time constraint (34), that is:

$$\sum_{i\in\mathbf{G}} t_i^{inv} = \sum_{i\in\mathbf{G}} g_i c_i \left(\frac{1}{\max_{j=1}^{i}(\tilde{\rho}_j)c_i} - 1\right) \le \Theta_m. \quad (38)$$

As smaller g_i means smaller left hand side of (38), we can get the minimal value of the left hand side $\mathcal{S}_{min} = \sum_{i\in\mathbf{G}} c_i(\frac{1}{\max_{j=1}^{i}(\tilde{\rho}_j)c_i} - 1)$, when all valid time lengths equal their corresponding WCET, i.e., $g_i = 1, \forall i \in \mathbf{G}$. Then, we can discuss the problem in two cases according to if the following inequality holds.

$$\mathcal{S}_{min} \le \Theta_m \quad (39)$$

Case A [$\mathcal{S}_{min} \le \Theta_m$]: In this case, at least one set of g_i can be found for each stage in $\mathbf{G}$ so that the constraint (38) is met. Therefore, all t^{inv}s can be set as their ideal solutions for low peak temperature. Then, the problem is solved once the set of g_i is determined.

Setting the set of g_i must be done carefully since very large or small values of g_i can cause high peak temperature. For very large values of g_i, the 'active' or 'idle' state lasts very long in each period, which may cause a high temperature at the end of valid time intervals. On the other hand, smaller g_i means higher switching frequency, which involves more switching overheads and thus also may cause high peak temperature. We name it the *extremum pessimistic phenomenon* for processors which are periodically switched to 'sleep' state. One can comprehend it according to its literal meaning: extreme small or large switching periods lead to pessimistic (high) temperatures. Fig. 5a presents an example of this phenomenon. For the sake of brevity, we also name the curves in this figure the *E-P curve*, which will be utilized as the guidance in determining the set of g_i.

Def. 1 (E-P Curve): the *E-P curve* of stage $\mathbb{S}_i$ with respect to a valid partition k, denoted as $\Upsilon_{i,k}$, is defined as the relationship between peak temperature and the sleep interval t_i^{inv} when, (1) the valid partition $K_i = k$, that is, $t_i^{vld} = \frac{k}{1-k} t_i^{inv}$, and (2) other stages always stay at 'sleep' state.

Generally, the *E-P curve* models the individual influence of the corresponding stage to the peak temperature when we apply same APTM scheme on each stage. From Fig. 5a, one can observe that the curves of two stages could be different.

Now we declare a set of notations that will be used in the following sections. The lower bound of K_i, which is $\max_{j=1}^{i}(\tilde{\rho}_j)c_i$ from (36), is denoted by K_i^m. The set of *E-P curves* of all stages in $\mathbf{G}$ is termed by $\mathbf{R}$, and $\Upsilon_{i,k}(t_i^{inv})$ denotes the value of $\Upsilon_{i,k}$ at point t_i^{inv}.

Algorithm 3 presents the pseudo code of assigning t^{inv}s for the stages in $\mathbf{G}$ in this case. Firstly, we calculate the constant step size for t_i^{inv}, which is the ideal solution when g_i equals 1 (line 1). We also initialize $g_i = 1$ and $t_i^{inv} = \tau_i$ because they are guaranteed to be feasible for constraints (34) and (35). Then, the values of t_i^{inv}s are increased with corresponding constant step sizes in a while loop until all of them cannot be increased any more (line 5) or further enlarging any of them will cause higher temperature (line 7). In each loop, we first find the valid stages set, i.e., the t_i^{inv} of which can be added by one step size τ_i without violating constraint (34). Then, the increasing reward ω_i is derived (line 6) according to Υ_{i,K_i^m}. The reward ω_i normalizes the benefit of increasing t_i^{inv} in lowering the temperature of the system. Finally, we add one step size to the t_i^{inv} having the highest increasing reward ω_i (lines 8 and 9), and update the corresponding g_i.

Algorithm 3 Assign APTM for $\mathbf{G}$ in Case A

```
Input: $\mathbf{G}$, $\Theta_m$, $\mathbf{K^m} = \{K_i^m | i \in \mathbf{G}\}$, $\{\Upsilon_{i,K_i^m} | i \in \mathbf{G}\}$
Output: $\{(t_i^{vld}, t_i^{inv}) | i \in \mathbf{G}\}$
 1: $\forall i \in \mathbf{G}$, set step size $\tau_i \leftarrow c_i(1/K_i^m - 1)$
 2: $\forall i \in \mathbf{G}$, $g_i \leftarrow 1$, $t_i^{inv} \leftarrow \tau_i$
 3: while true do
 4:     valid set $\mathbf{V} \leftarrow \{i | \sum_{i\in\mathbf{G}} t_i^{inv} + \tau_i \le \Theta_m\}$.
 5:     $\mathbf{V} = \emptyset$ ? break : goto next line
 6:     reward $\omega_i \leftarrow \frac{\Upsilon_{i,K_i^m}(t_i^{inv}) - \Upsilon_{i,K_i^m}(t_i^{inv}+\tau_i)}{\tau_i}$, $\forall i \in \mathbf{V}$
 7:     $(\forall i \in \mathbf{V}, \omega_i \le 0)$ ? break : goto next line
 8:     find $i$ where $\omega_i = \max_{i\in\mathbf{V}} \omega_i$
 9:     $t_i^{inv} \leftarrow t_i^{inv} + \tau_i$, $g_i \leftarrow g_i + 1$.
10: end while
11: $t_i^{vld} \leftarrow g_i \times c_i$, $\forall i \in \mathbf{G}$
```

Case B [$S_{min} > \Theta_m$]: In this case, it's clear that all t^{inv}s cannot be their ideal values simultaneously even all valid time lengths t^{vld}s equal their corresponding c. Therefore, the t^{vld}s are set as their lower bound c_i. Then, we also assign t^{inv}s according to the unique thermal properties of the stages, which is the *cooling curves*.

Def. 2 (cooling Curve): The *cooling curve* of stage $\mathbb{S}_i$ with respect to an application Φ, denoted as $\Omega_{i,\Phi}$, is defined as the relationship between peak temperature and the sleep interval t_i^{inv} when, (1) $t_i^{vld} = c_i$, and other stages always stay at 'sleep' state.

The *cooling curves* can model the abilities of different stages in cooling the system when they are assigned the same t^{inv} while the active intervals t_i^{vld} are set as the corresponding c_i. For example, consider $\Theta_m = 10$ ms, then we have a 'budget' of 10 ms for the stages in **G**. However, giving a same 'budget' to different stages may have different effects in lowering the temperature due to (1) the different locations of the stages in the floorplan; (2) the stages are handling different tasks. The *cooling curves* is used to describe such properties of the stages. Note that although it is based on the WCET and obtained offline, it can still offer acceptable accuracy during assigning APTM schemes online, due to the following reasons.

- As the stage locations are fixed, the first factor plays the same effect during offline and online phases.
- Although WCET and online execution times are different in most cases, WCET could also give a qualitative prediction about real execution times.
- The difference between offline WCET and online execution time makes little influence on the allocation of sleep times to stages, because the cooling curves just use their slopes to determine the relative weight of allocating sleep times to each stage, as shown in Algorithm 4.
- The cooling curve is utilized to minimize online calculation overhead. Thus, a modicum of degradation in accuracy is still acceptable.

The pseudo codes of assigning APTM schemes in this case are listed in Algorithm 4. The general procedure is similar to that in case A, except that: (1) the incremental step size is a parameter given by designers to control the assigning rate; (2) at each iteration, we increase all t^{inv}s by the process of weighting instead of only change one t^{inv} (lines 8 and 11). The weight of each t_i^{inv} is determined by the current cooling effect of stage $\mathbb{S}_i$ (line 6), which is represented by the current slope of the linear approximation of its corresponding *cooling curve*. Moreover, the step size is also checked whether increasing t^{inv}s by it will violate Θ_m (line 7), and is revised when necessary (line 10). The algorithm continues until all of t^{inv}s reach their upper bounds (line 4) or their sum equals Θ_m (line 11).

Algorithm 4 Assign APTM for **G** in Case B

```
Input: G, $\Theta_m$, $\{\Omega_{i,\Phi} | i \in \mathbf{G}\}$,, step size $z$
Output: $\{(t_i^{vld}, t_i^{inv}) | i \in \mathbf{G}\}$
1: $\forall i \in \mathbf{G}$, $t_i^{vld} \leftarrow c_i$, $t_i^{inv} \leftarrow t_i^{swon} + t_i^{swoff}$
2: while true do
3:     valid set V $\leftarrow \{i | t_i^{inv} \leq \frac{t_i^{vld}}{\max_{j=1}^{i}(\tilde{\rho}_j)c_i} / - t_i^{vld}\}$.
4:     $\mathbf{V} = \emptyset$ ? break : goto next line
5:     get the gradient $\eta_i$ at $t_i^{inv}$ from $\Omega_{i,\Phi}$, $\forall i \in \mathbf{V}$
6:     assigning weight $\omega_i \leftarrow |\frac{\eta_i}{\sum_{i \in V} \eta_i}|$, $\forall i \in \mathbf{V}$
7:     if $\sum_{i \in \mathbf{G}} t_i^{inv} + z < \Theta_m$ then
8:         $t_i^{inv} \leftarrow t_i^{inv} + z \times \omega_i$, $\forall i \in \mathbf{V}$
9:     else
10:        $z \leftarrow \Theta_m - \sum_{i \in \mathbf{G}} t_i^{inv}$
11:        $t_i^{inv} \leftarrow t_i^{inv} + z \times \omega_i$, $\forall i \in \mathbf{V}$, then break.
12:    end if
13: end while
```

VIII. OFFLINE PART

So far, the online part of our approach has been elaborated. Our approach is an offline and online combined one for effective thermal optimization and high efficiency. The offline part is discussed in this section.

The offline part includes three parts: acquiring the (1) *E-P curves* and (2) *cooling curves* of the system, and (3) finding the best parameter λ for online adapting. The *E-P curves* and *cooling curves* of the system can be easily acquired by implementing different APTM schemes on the system according to their definition and then recording the peak temperature. Due to the simplicity, the detailed algorithms are omitted for brevity.

The pseudo codes of finding λ are listed in algorithm 5. For a task application Φ, the parameter λ is found by brutally searching its feasible range (0, 1) with a constant step size ϵ (line 2). The λ is set as the one which leads to the minimal peak temperature when the system running application Φ (line 6). To diminish the influence of the variability in event arrivals and executions, the peak temperature is calculated as the average values of N_s experiments (line 4).

Algorithm 5 Find the Parameter λ

```
Input: $\Phi$, cooling curves, E-P curves, step $\epsilon$, $N_s$
Output: $\lambda$
1: $T_{best} \leftarrow \infty$.
2: for $\bar{\lambda} = \epsilon$ to $1 - \epsilon$ with step $\epsilon$ do
3:     run the application $\Phi$ $N_s$ times with approach APTM
   which uses $\bar{\lambda}$ in (25).
4:     get the average peak temperature $T_{avg}$.
5:     if $T_{avg} \leq T_{best}$ then
6:         $T_{best} \leftarrow T_{avg}$, and $\lambda \leftarrow \bar{\lambda}$
7:     end if
8: end for
```

IX. EVALUATION

In this section, we report results of experiments and simulations that conducted to evaluate the effectiveness and efficiency of APTM. We first present the results of experiments on a desktop computer with a four-core processor. Then, we perform simulations in MATLAB for pipelined systems with more stages, i.e., from four to sixteen stages. The simulations complement experiments results by allowing us to examine APTM performance on processors with different numbers of cores. The proposed approach APTM is compared with two existing approaches, i.e., (1) the Balanced Workload Scheme (BWS) proposed in [1]. (2) the Pay-Burst-Only-Once based offline approach (O-PBOO) presented in [13]. BWS is an online energy-optimization approach based on adaptive dynamic power management. O-PBOO is an offline peak-temperature optimization approach.

A. EXPERIMENTS ON REAL PLATFORM

We have implemented APTM on top of Linux, using the framework Multi-core Fast Thermal Prototyping (McFTP) [29]. McFTP is a general-purpose framework designed for fast prototyping and evaluating thermal managements on computers with temperature sensors. The min-plus and max-plus algebra operators for arrival curves and service curves are already implemented in the RTC toolbox [30] with Java. For an n-stage system, n worker threads are created to represent the stages. The threads are attached to different cores according to the pipelined architecture. The jobs executed by a worker could be user-defined functions or predefined thirty-three benchmarks selected from the *Stress-ng* tool. The power dissipation mode of the cores is controlled via the Advanced Configuration and Power Interface (ACPI), i.e., the P-states and C-states. When there is no job to execute, a worker switches to the 'idle' process, i.e., continuously checking the job queue in a while loop.

The used hardware platform is a Dell Optiplex 9020 desktop computer equipped with an Intel I7-4770k processor and 16 GB memory. The processor has four physical cores that can run with frequency from 800 MHz to 3.4 GHz. Four physical temperature sensors are built in the processor to read the temperature of the cores. The Hyper Thread feature of the processor is disabled in the BIOS (Basic Input/Output System) of the computer. The linux kernel version is 3.16.0-53-generic. The McFTP framework and C++ programs are compiled by g++ 4.8.4 with optimization level O3 turned on. When the intel-power driver is applied, the C-states defined in the Advanced Configuration and Power Interface (ACPI) for every core are C0, POLL, C1, C3, C6, and C7. The cores can switch to C-state C7 and significantly lower its temperature.

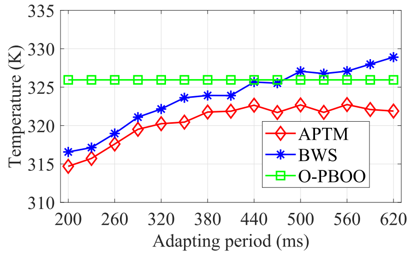
1) SETUP

During the experiments, the system runlevel of the operating system is set to the lowest level 1 so that only essential system services are running. The frequencies of all cores are set as 3.4 GHz in the experiments. Four worker threads are created and each one is attached to one core. The other threads such as dispatcher, power manager are evenly attached to all the cores. During the experiments, the period of sampling the temperature is set as 200 ms to collect enough data for analysis. The adopted arrival curve in this paper can model many common timing models of event stream, such as the sporadic event model and the periodic event models. For clarity of demonstration, we adopt the event stream modeled by the PJ timing model in case studies. The PJ timing model specifies an event stream by the period p, jitter j. The release period and jitter of the studied task, i.e., the benchmark *sqrt_rand*, are set as one second and 1.5 seconds, respectively. The WCETs of the task on the cores of platform are [142, 90, 36, 57] ms. Each experiment is run for one hundred seconds. During two experiments, we also cool the computer for one hundred seconds to limit the thermal influence from the previous one to the later one.

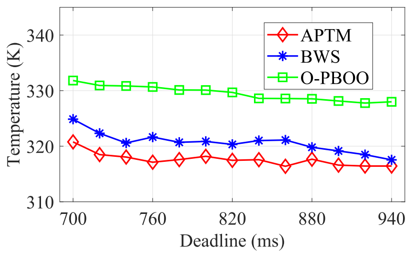
2) RESULTS

We show the results of experiments on the hardware platform in this section. The temperatures of the cores are obtained from the physical sensor in the processor. We study how the temperature of our approach varies when the online adapting period, the relative deadline and the tasks are changing, respectively. Different benchmarks are adopted as different tasks. In the experiments of benchmarks, the release periods are also set as one second. Since some benchmarks finish in less than one ms, we run the benchmark N times and consider it as one job. The number of N is randomly determined during execution. The averages peak temperature of the four cores in the three scenarios are plotted in Fig. 6a, Fig. 6b and Fig. 7, respectively.

From the results shown in the figures, we can make two observations. (1) Our approach APTM outperforms the other



(a) The relative deadline is 900 ms.



(b) The adapting period is 200 ms.

FIGURE 6. The average peak temperature of the three approaches for different settings.

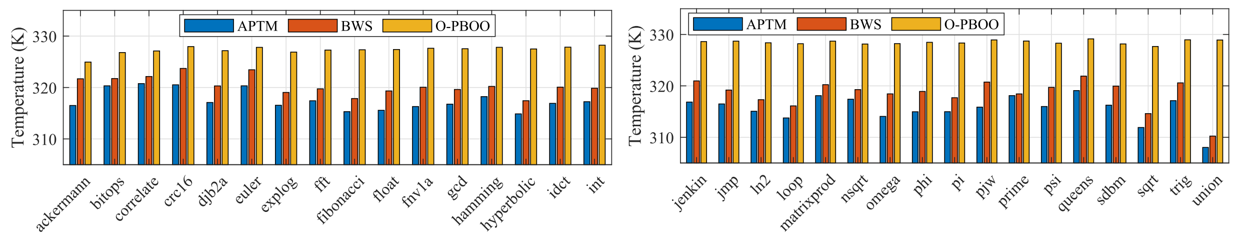


FIGURE 7. The average peak temperature of the three approaches for the different benchmarks. The adapting period and relative deadlines are 200 ms and 900 ms, respectively.

TABLE 1. The number of adapting overheads in different ranges.

approach	> 900 us	> 800 us	> 700 us	> 600 us	> 500 us	≤ 500 us
APTM	56	160	173	218	631	188189
BWS	40	95	177	178	212	188608

two approaches in all cases. As seen in Fig. 6a and Fig. 7, APTM achieves maximal 8 K and 5 K temperature reductions compared to BWS, respectively. (2) The difference between the temperatures of APTM and BWS becomes larger when the adapting period increases. This strengthens the conclusion we made in Section II that the effectiveness of APTM receives little influence from the adapting period while BWS does not. Note that this phenomenon is much more obvious for large adapting periods. (3) For large relative deadline scenarios, the temperature gaps between BWS and APTM are narrower. This is expected since the processor is not sufficiently stressed in these scenarios, leading to smaller temperature differences.

We also report the efficiency of APTM in experiments on the hardware platform. In the experiments, we recorded the adapting online overheads of the two approaches at every adapting instant. In the experiments, totally 188820 samples have been recorded. The maximal overheads of APTM and BWS in the samples are 2047 us and 1996 us, respectively. The average overheads of the two approaches are only 199.3 us and 162.3 us. The numbers of the adapting overheads in different ranges are listed in Table. 1. Observe that 99.6% overheads of the two approaches locate in range [0, 500] us. Fig. 8 displays the normalized probability histograms in this

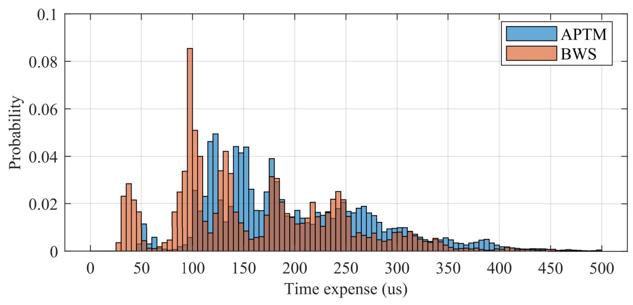


FIGURE 8. The probability density histogram for the adapting overheads of the two approaches in range [0, 500] us. They are represented with a bin width 5 us.

range for the two approaches. One can observe that most adapting overheads of BWS are distributed in [30, 350] us while those of APTM are in [50, 400] us, which indicates that, generally, APTM requires more time to calculate thermal managing schemes than BWS. This is expected because APTM considers the unique thermal properties of the stages and thus requires more calculations. However, 99.6% adapting overheads of APTM are less than 500 us, which is acceptable as an online approach.

B. SIMULATION

In this section, we evaluate the effectiveness and feasibility of APTM for pipelined systems having more stages. The three approaches are implemented in MATLAB with the RTC-toolbox and RTS-toolbox. All the simulations are performed on the same computer. The thermal model for calculating temperature evolution is defined as below. It is worth noting that the thermal model is only required for simulation scenarios or system using soft temperature sensors. For systems with hardware temperature sensors, thermal model and calibrating the parameters are not necessary.

C. THERMAL MODEL

The multi-core processor is modeled by the celebrated thermal model HotSpot [31]. We consider the vertical layout of the processor is composed of four layers, i.e., the heat sink, heat spreader, thermal interface and silicon die layers [32]. Every layer is divided into a set of blocks according to the processing components on the silicon die layer, i.e., the floorplan. We consider heat can flow among blocks in the same layer as well as different layers. We denote the number of total blocks as N. The cooling and heating phenomena is modeled by applying the well-known Fourier model. In order to calculate thermal parameters, we take the advantage of the well-known electro-thermal analogy [9], [12], [31]–[36], i.e., the RC thermal network. Finally, the temperature of all blocks $\mathbf{T}(t) = [T_1(t), \ldots, T_N(t)]$ can be given by a set of first-order differential equations [32]:

$$\mathbf{C} \cdot \frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} = (\mathbf{P}(t) + \mathbf{M} \cdot \mathbf{T}^{amb}) - (\mathbf{G} + \mathbf{M}) \cdot \mathbf{T}(t), \quad (40)$$

where $\mathbf{C}$ is the thermal capacitance matrix, $\mathbf{P}$ is the power dissipation vector, $\mathbf{M}$ is the thermal ground conductance matrix, $\mathbf{G}$ is the thermal conductance matrix and $\mathbf{T}^{amb} = T^{amb} \cdot [1, \ldots, 1]'$, where T^{amb} is the ambient temperature. The total power dissipation P_i of the i^{th} block contains two parts: dynamic power P_i^d and leakage power P_i^l [11], [12], [37].

$$P_i(t) = P_i^d(t) + P_i^l(t) \quad (41)$$

The dynamic power depends on the supply voltage and clock frequency of the core. Therefore, we consider P_i^d keeps constant when i^{th} block stays in each state, i.e., 'active', 'idle' or 'sleep'. The leakage power, also named as the static power, can also be influenced by temperature. The dependency relationship between the leakage power and temperature can be closely approximated by a linear function of the processor temperature [12], [32], [34], [38], [39]:

$$P_i^l(t) = w_i T_i(t) + v_i, \quad (42)$$

where w_i is a constant coefficient and v_i keeps unchanged in each power dissipation mode, i.e., v_i has different values in 'active', 'idle' and 'sleep' modes.

1) SETUP

To investigate the effectiveness and efficiency of our approach on pipelined architectures with different stage numbers, a real life platform is used in our case studies: the Single-Chip Cloud Computer (SCC), a processor created by Intel that has 48 distinct physical cores [40]. The power and thermal parameters to construct the thermal model of the platform come from [41]. Moreover, the well-known HotSpot toolbox is adopted to construct the thermal model from the floorplan and thermal parameters of the platform. The processors in simulation and experiment are different in floorplans and thermal parameters, thus their timing parameters are quite different. The SCC thermal model in simulation has smaller thermal timing parameters. Therefore, the parameters such as task period, task jitter and adapting period are changed in simulations. The adopted input stream is set as: $p = 100$ ms, $j = 150$ ms. The event trace length of the application is set as 15 seconds. The event trace is generated by the RTS toolbox according to its arrival curve α^u. During the simulation, the real execution time of an event is randomly chosen in range $[0.5e^w, e^w]$. where e^w is the WCET of the event. All the mode-switching overheads are set as $\mathbf{t}^{\mathbf{swoff}} = \mathbf{t}^{\mathbf{swon}} = (1, \ldots, 1)$ ms.

2) RESULTS

We report the performance of our approach with respect to the stage number in this section. The three approaches are tested from 4-stage up to 16-stage scenarios. The WCETs of all sub-tasks are randomly generated in range $[5, 9]$ ms. We set the deadline for n-stage scenarios as $(20 + 15n)$ ms, and set the adapting period as 5 ms to investigate the effectiveness of our approaches with small adapting periods. The peak temperatures in the simulation are plotted in Fig. 9a. Fig. 9b reports the average computing times of approach APTM and BWS with different stage numbers.

We first discuss the results in Fig. 9a. The peak temperatures of the three approaches grow as the stage number gets bigger. This is expected because as more cores are activated, more heat is generated. Observe that: (1) our approach APTM gives lower peak temperature than the other approaches, which further strengthens the effectiveness of our approach for pipeline architectures with different stage numbers and small adapting periods. (2) The temperature from APTM grows slowly while the temperature from BWS increases faster, which demonstrates the scalability of APTM with respect to the stage number.

Finally, let's discuss the overhead of online adapting. As shown in Fig. 9b, the average adapting overheads of APTM in different cases are less than 0.5 ms and grow approximated linearly. We can also observe that the average overheads of APTM are higher than that of BWS, this agrees with the experiment results. However, as demonstrated above, our approach works better than BWS, especially in scenarios having large adapting periods. Therefore, the online adapting overhead of APTM is acceptable considering the achievement in lowering peak temperature. Note that the overheads of both approaches are slightly different with those in experiments.

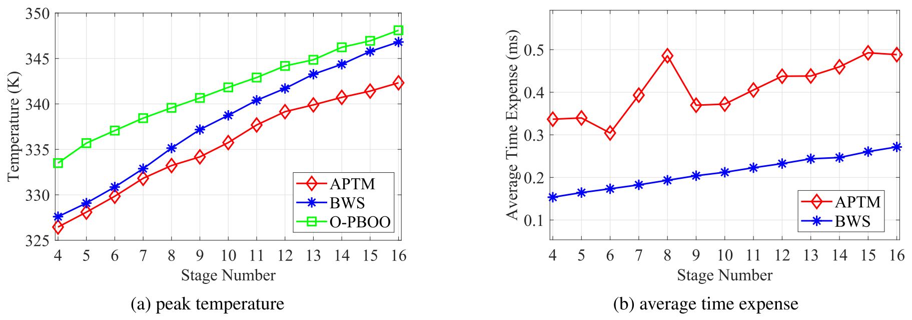


FIGURE 9. (a) The peak temperature from three approaches and (b) the average adapting overheads of online adapting of APTM and BWS, on platform IntelSCC with different active stage numbers.

This is expected because the evaluations are implemented in different software environments, i.e., C++ in experiment and MATLAB in simulation.

X. CONCLUSION AND FUTURE WORK

We have presented a new Dynamic Power Management approach named Adaptive Periodic Thermal Management to minimize the peak temperature of pipelined hard real-time systems. We implement our approach on a real hardware platform and compare it with two existing approaches. Results show that our approach can reduce the peak temperature more effectively than the two approaches. The peak temperature can be reduced by up to 8 K for the Intel I7 processor, and 7.5 K for the Intel SCC using four to sixteen processor cores. We also investigate the online adapting overhead of our approach. From 188820 samples, we obtain the average adapting overhead of APTM, which is only 199.3 us. This indicates the high efficiency of APTM. To investigate the performance of our approaches on platforms with different numbers of stages, we also simulate APTM on the Single-Chip Cloud Computer platform. Case studies results demonstrate our approach is scalable w.r.t the stage number, in terms of both temperature optimization and computation efficiency.

In this paper, we focused on using dynamic power management to optimize the temperature. There are also other factors that have notable impacts on temperature and worth further research. In the future work, we plan to study combining APTM with choosing optimal task binding policy in thermal management of MPSoC. In addition, applying dynamic frequency scaling on pipelined real-time systems is also an interesting topic.

REFERENCES

[1] G. Chen, K. Huang, and A. Knoll, "Adaptive dynamic power management for hard real-time pipelined multiprocessor systems," in *Proc. IEEE 20th Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2014, pp. 1–10.

[2] M. Ahmed, N. Fisher, S. Wang, and P. Hettiarachchi, "Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources," *Sustain. Comput. Inform. Syst.*, vol. 1, no. 3, pp. 226–240, 2011.

[3] A. Alimonda, A. Acquaviva, and S. Carta, "Temperature and leakage aware power control for embedded streaming applications," in *Proc. 11th EUROMICRO Conf. Digit. Syst. Design Archit. Methods Tools*, Sep. 2008, pp. 107–114.

[4] F. Mulas, D. Atienza, A. Acquaviva, S. Carta, L. Benini, and G. D. Micheli, "Thermal balancing policy for multiprocessor stream computing platforms," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 28, no. 12, pp. 1870–1882, Dec. 2009.

[5] X. Qin and P. Mishra, "TECS: Temperature-and energy-constrained scheduling for multicore systems," in *Proc. 27th Int. Conf. VLSI Design 13th Int. Conf. Embedded Syst.*, Jan. 2014, pp. 216–221.

[6] M. Mohaqeqi, M. Kargahi, and K. Fouladi, "Stochastic thermal control of a multicore real-time system," in *Proc. 24th Euromicro Int. Conf. Parallel Distrib. Netw.-Based Process. (PDP)*, Feb. 2016, pp. 208–215.

[7] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *Proc. 15th IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2009, pp. 131–140.

[8] Y. Fu, N. Kottenstette, C. Lu, and X. D. Koutsoukos, "Feedback thermal control of real-time systems on multicore processors," in *Proc. 10th Int. Conf. Embedded Softw.*, 2012, pp. 113–122.

[9] P. M. Hettiarachchi, N. Fisher, and L. Y. Wang, "Achieving thermal-resiliency for multicore hard-real-time systems," in *Proc. 25th Euromicro Conf. Real-Time Syst.*, Jul. 2013, pp. 37–46.

[10] M. Cox, A. K. Singh, A. Kumar, and H. Corporaal, "Thermal-aware mapping of streaming applications on 3D multi-processor systems," in *Proc. 11th IEEE Symp. Embedded Syst. Real-time Multimedia*, Oct. 2013, pp. 11–20.

[11] T. Wang, M. Fan, G. Quan, and S. Ren, "Heterogeneity exploration for peak temperature reduction on multi-core platforms," in *Proc. 15th Int. Symp. Qual. Electron. Design*, Mar. 2014, pp. 107–114.

[12] P. M. Hettiarachchi, N. Fisher, M. Ahmed, L. Y. Wang, S. Wang, and W. Shi, "A design and analysis framework for thermal-resilient hard real-time systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 5s, p. 146, 2014.

[13] L. Cheng, K. Huang, G. Chen, B. Hu, and A. Knoll, "Minimizing peak temperature for pipelined hard real-time systems," in *Proc. Design Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2016, pp. 1090–1095.

[14] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *Proc. TCAD*, Nov. 2007, pp. 281–288.

[15] N. Bansal and K. Pruhs, "Speed scaling to manage temperature," in *Proc. Annu. Symp. Theor. Aspects Comput. Sci.* Berlin, Germany: Springer, 2005, pp. 460–471.

[16] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Proc. 21st IEEE Int. Symp. Circuits Syst. Emerg. Technol.*, vol. 4, May 2000, pp. 101–104.

[17] J. Y, Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, Vol. 2050. Springer, 2001.

[18] L. Thiele, E. Wandeler, and N. Stoimenov, "Real-time interfaces for composing real-time systems," in *Proc. 6th ACM IEEE Int. Conf. Embedded Softw.*, 2006, pp. 34–43.

[19] K. Lampka, K. Huang, and J.-J. Chen, "Dynamic counters and the efficient and effective online power management of embedded real-time systems," in *Proc. 7th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, 2011, pp. 267–276.

[20] L. Cheng, "System level periodic thermal management for hard real-time system," Ph.D. dissertation, Technische Univ. München, Munich, Germany, 2018.

[21] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 3s, p. 111, 2014.

[22] K. Lampka and B. Forsberg, "Keep it slow and in time: Online DVFS with hard real-time workloads," in *Proc. Design Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2016, pp. 385–390.

[23] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature-aware DVFS," *IEEE Trans. Comput.*, vol. 59, no. 1, pp. 127–133, Jan. 2010.

[24] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *Proc. IEEE 32nd Real-Time Syst. Symp.*, Nov./Dec. 2011, pp. 34–43.

[25] K. Huang, G. Chen, C. Buckl, and A. Knoll, "Conforming the runtime inputs for hard real-time embedded systems," in *Proc. 49th Annu. Design Automat. Conf.*, 2012, pp. 430–436.

[26] M. Fidler, "Extending the network calculus pay bursts only once principle to aggregate scheduling," in *Proc. Int. Workshop Qual. Service Multiservice IP Netw.* Berlin, Germany: Springer, 2003, pp. 19–34.

[27] G. Chen, K. Huang, C. Buckl, and A. Knoll, "Applying pay-burst-only-once principle for periodic power management in hard real-time pipelined multiprocessor systems," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 20, no. 2, p. 26, 2015.

[28] L. Cheng, K. Huang, G. Chen, B. Hu, and A. Knoll, "Periodic thermal management for hard real-time systems," in *Proc. 10th IEEE Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2015, pp. 1–10.

[29] L. Cheng, G. Chen, J. Li, Z. Zhao, A. Knoll, and K. Huang, "McFTP: A framework for fast thermal managements prototyping on real multi-core processors," *Sustain. Comput. Inform. Syst.*, vol. 22, pp. 191–205, Jun. 2018.

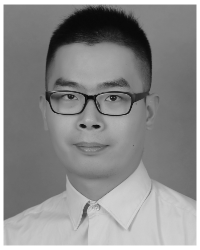
[30] E. Wandeler and L. Thiele. (2006). *Real-Time Calculus (RTC) Toolbox*. [Online]. Available: http://www.mpa.ethz.ch/Rtctoolbox

[31] 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. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[32] L. Thiele, L. Schor, I. Bacivarov, and H. Yang, "Predictability for timing and temperature in multiprocessor system-on-chip platforms," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 1s, p. 48, 2013.

[33] T. Chantem, X. S. Hu, and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 10, pp. 1884–1897, Oct. 2011.

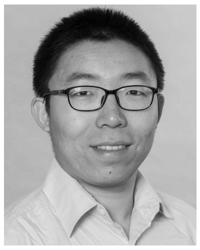
[34] V. Hanumaiah and S. Vrudhula, ''Temperature-aware DVFS for hard real-time applications on multicore processors,'' *IEEE Trans. Comput.*, vol. 61, no. 10, pp. 1484–1494, Oct. 2012.
[35] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. De Micheli, ''Temperature-aware processor frequency assignment for MPSoCs using convex optimization,'' in *Proc. 5th IEEE/ACM Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, 2007, pp. 111–116.
[36] S. Pagani, H. Khdr, W. Munawar, J.-J. Chen, M. Shafique, M. Li, and J. Henkel, ''TSP: Thermal safe power: Efficient power budgeting for many-core systems in dark silicon,'' in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, 2014, p. 10.
[37] R. Rao and S. Vrudhula, ''Fast and accurate prediction of the steady-state throughput of multicore processors under thermal constraints,'' *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1559–1572, Oct. 2009.
[38] Y. Liu, R. P. Dick, L. Shang, and H. Yang, ''Accurate temperature-dependent integrated circuit leakage power estimation is easy,'' in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Apr. 2007, pp. 1–6.
[39] L. Schor, I. Bacivarov, H. Yang, and L. Thiele, ''Worst-case temperature guarantees for real-time applications on multi-core systems,'' in *Proc. IEEE 18th Real Time Embedded Technol. Appl. Symp.*, Apr. 2012, pp. 87–96.
[40] J. Howard *et al.*, ''A 48-core IA-32 message-passing processor with DVFS in 45 nm CMOS,'' in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2010, pp. 108–109.
[41] M. Sadri, A. Bartolini, and L. Benini, ''Single-chip cloud computer thermal model,'' in *Proc. 17th Int. Workshop Therm. Invest. ICs Syst. (THERMINIC)*, Sep. 2011, pp. 1–6.



LONG CHENG received the B.S. degree in mechanical design manufacturing and automation and the M.Eng. degree in mechanical engineering from the Harbin Institute of Technology, China, in 2011 and 2013, respectively, and the Ph.D. degree in computer science from the Technical University of Munich, in 2018. He is currently with the School of Data and Computer Science, Sun Yat-sen University. His research interests include power and thermal management of real-time systems, snake-like robots, and artificial neural networks.

KAI HUANG received the B.Sc. degree from Fudan University, China, in 1999, the M.Sc. degree from the University of Leiden, The Netherlands, in 2005, and the Ph.D. degree from ETH Zurich, Switzerland, in 2010. He joined Sun Yat-sen University as a Professor, in 2015, where he was appointed as the Director of the School of Data and Computer Science, Institute of Unmanned Systems, in 2016. He was a Senior Researcher with the Computer Science Department, Technical University of Munich, Germany, from 2012 to 2015, and a Research Group Leader of fortiss GmbH, Munich, Germany, in 2011. His research interests include techniques for the analysis, design, and optimization of embedded systems, particularly in the automotive and robotics domains. He received the Program of Chinese Global Youth Experts 2014 Award, the Chinese Government Award for Outstanding Self-Financed Students Abroad 2010, the Best Paper Awards at the ESTC 2017, ESTIMedia 2013, SAMOS 2009, the Best Paper Candidate ROBIO 2017, ESTMedia 2009, and the General Chairs' Recognition Award for Interactive Papers in CDC 2009. He has been a member of the Technical Committee on Cybernetics for CyberPhysical Systems of the IEEE SMC Society, since 2015.

GANG CHEN received the B.E. degree in biomedical engineering, the B.S. degree in mathematics and applied mathematics, and the M.S. degree in control science and engineering from Xi'an Jiaotong University, China, in 2008, 2008, and 2011, respectively, and the Ph.D. degree in computer science from Technische Universität München, Germany, in 2016. He is currently an Associate Professor with Sun Yat-sen University, China. His research interests include mixed-criticality systems, energy-aware real-time scheduling, certifiable cache architecture design, and high-performance computing.



ZHENSHAN BING received the B.S. degree in mechanical design manufacturing and automation and the M.Eng. degree in mechanical engineering from the Harbin Institute of Technology, China, in 2013 and 2015, respectively, and the Ph.D. degree from the Faculty of Informatics, Technical University of Munich, Germany, in 2019. His research interest includes snakelike robot, which is controlled by artificial neural networks and its related applications.

ALOIS C. KNOLL received the Diploma (M.Sc.) degree in electrical and communications engineering from the University of Stuttgart, Germany, in 1985, and the Ph.D. degree *(summa cum laude)* in computer science from the Technical University of Berlin, Germany, in 1988. He served in the Faculty of the Computer Science Department, TU Berlin, until 1993. He joined the University of Bielefeld, Germany, as a Full Professor and served as the Director of the Technical Informatics Research Group, until 2001. Since 2001, he has been a Professor with the Department of Informatics, Technical University of Munich (TUM), Germany. He was also on the Board of Directors of the Central Institute of Medical Technology at TUM (IMETUM). From 2004 to 2006, he was the Executive Director of the Institute of Computer Science, TUM. From 2007 to 2009, he was a member of the EU's Highest Advisory Board on Information Technology, Information Society Technology Advisory Group (ISTAG), and a member of its subgroup on Future and Emerging Technologies (FET). In this capacity, he was actively involved in developing the concept of the EU's FET Flagship projects. His research interests include cognitive, medical, and sensor-based robotics, multi-agent systems, data fusion, adaptive systems, multimedia information retrieval, model-driven development of embedded systems with applications to automotive software and electric transportation, as well as simulation systems for robotics and traffic.