TCEC: Temperature- and Energy-Constrained Scheduling in Real-Time Multitasking Systems

Xiaoke Qin, Weixun Wang, *Student Member, IEEE*, Prabhat Mishra, *Senior Member, IEEE*

***Abstract*—The ongoing scaling of semiconductor technology is causing severe increase of on-chip power density and temperature in microprocessors. This urgently requires both power and thermal management during system design. In this paper, we propose a model checking based technique using extended timed automata to solve the processor frequency assignment problem in a temperature- and energy-constrained multitasking system. We also develop a polynomial time approximation algorithm to address the state space explosion problem caused by symbolic model checker. Our approximation scheme is guaranteed to not generate any false positive answer, while it may return false negative answer in rare cases. Our method is universally applicable since it is independent of any system and task characteristics. Experimental results demonstrate the usefulness of our approach.**

I. Introduction

Along with the performance improvement in state-of-the-art microprocessors, power densities are rising rapidly due to the fact that feature size scales faster than voltages. In last five years, though the processor frequency is only improved by 30%, the power density is more than doubled and expected to reach over $250W/cm^2$ [1]. Since energy consumption is converted into heat dissipation, high heat flux increases the on-chip temperature. The "hot spot" on current microprocessor die, caused by nonuniform peak power distribution, could reach up to 120°C [2]. This trend is observed in both desktop and embedded processors [3].

Since high on-chip thermal dissipation has severe detrimental impact, we have to control the instantaneous temperature so that it does not go beyond a certain threshold. Thermal management schemes at all levels of system design are widely studied for general-purpose systems. However, in the context of embedded systems, traditional packaging and cooling solutions are not applicable due to the constraints on device size and cost. Moreover, embedded systems normally have limited energy budgets since most devices are driven by batteries. Multitasking systems with real-time constraints add another level of difficulty since tasks have to meet their deadlines. Since such systems normally have well-defined functionalities, this multi-objective problem admits design-time algorithms.

Dynamic voltage scaling (DVS) is acknowledged as one of the most efficient techniques used in both energy optimization and temperature management. In existing literatures, ***temperature (energy)-constrained*** means that there is a temperature threshold (energy budget) which cannot be exceeded, while ***temperature (energy)-aware*** means that there is no constraint but maximum instantaneous temperature (total energy consumption) needs to be minimized. In this paper, we propose a formal method based on model checking for temperature- and energy-constrained (**TCEC**) scheduling in multitasking systems.

There are two important contributions in this paper. We developed a flexible and automatic design flow which models the TCEC problem in timed automata and solves it using formal verification techniques. Our approach is applicable to a wide variety of system and task characteristics as well as can incorporate runtime voltage scaling overhead. We also present an approximation algorithm, which effectively address the state space explosion problem. The approximation scheme will give no false positive answer, while its possibility to report false negative answer can be small enough for practical usage.

The rest of the paper is organized as follows. Section II introduces relevant existing research works. Section III provides related background information. Section IV provides an overview of our framework. Section V and Section VI describes our contribution in details. Experimental results are presented in Section VIII. Section IX concludes the paper.

II. Related Work

Energy-aware scheduling techniques for real-time systems have been widely studied to reduce energy consumption. For example, Jejurikar et al. solved the energy-aware scheduling problem for non-preemptive task sets [4]. Zhang et al. [5] has shown that applying DVS in real-time systems is a NP-hard problem. Optimal and approximation algorithms are given in [5] [6], while other works proposed heuristics. However, these techniques are not aware of controlling the operating temperature.

Temperature-aware scheduling in real-time systems has drawn significant research interests in recent years. wang et al. [7] introduced a simple reactive DVS scheme aiming at meeting task timing constraints and maintaining processor safe temperature. Zhang et al. [8] proved the NP-hardness of temperature-constrained performance optimization problem in real-time systems and proposed an approximation algorithm. However, none of these techniques solves TCEC problem.

Existing research formulated the voltage/frequency assignment problems in different models. For example, Chantem et al. [9] used ILP to model scheduling problem with steady-state temperature constraints. Unfortunately, when transient

The authors are with the Department of Computer and Information Science and Engineering at the University of Florida, Gainesville FL 32611-6120, USA (email:{xqin, wewang, prabhat}@cise.ufl.edu).

temperature is considered, the expansion of the temperature constraint introduces a large number of product terms, which prevent us to solve the problem efficiently using ILP solvers. Coskun et al. [10] circumvented this problem using an iterative ILP and thermal simulation approach, although the convergence to the optimal solution is not guaranteed. Although it is possible to accelerate the solving process by converting the problem into Pseudo-Boolean satisfiability problem, like SAT-based test generation techniques [11] [12], its applicability to large problems is limited by the capacity of solvers.

There are several studies on dynamic power management (DPM) using formal verification methods for embedded systems [13] and multiprocessor platforms [14]. Shukla et al. [13] provided a preliminary study on evaluating DPM schemes using an off-the-shelf model checker. Lungo et al. [14] tried to incorporate verification of DPM schemes in the early design stage. None of these approaches considers temperature management in such systems. Moreover, they did not account for energy and timing constraints, which are important in real-time embedded systems.

Temperature- or energy-constrained scheduling problems are also related to the multi-constrained path (MCP) problem for Quality of Service (QoS). MCP was extensively studied by network community. For example, Chen et al. [15] designed an approximation algorithm for MCP with two constraints. Xue et al. [16] proposed polynomial time approximation algorithms, which can be applied for more than two constraints. However, since the QoS costs are usually modeled as additive constants, these existing methods cannot be applied directly to solve TCEC problem due to the fact that the computation of the temperature is not additive.

III. Background and Problem Formulation

In this section, we will provide the formal description of the TCEC scheduling problem.

A. Thermal Model

A thermal RC circuit is normally utilized to model the temperature variation behavior of a microprocessor [8]. We adopt the RC circuit model proposed in [17], to capture the heat transfer phenomena in the processor. If P denotes the power consumption during a time interval, R denotes the thermal resistance, C represents the thermal capacitance, T_{amb} and T_{init} are the ambient and initial temperature, respectively, the temperature at the end of the time interval t can be calculated as:

$$\begin{aligned} T &= P \cdot R + T_{amb} - (P \cdot R + T_{amb} - T_{init}) \cdot e^{\frac{-t}{RC}} \\ &= (1 - e^{\frac{-t}{RC}})T_s + e^{\frac{-t}{RC}} T_{init} \end{aligned} \tag{1}$$

where t is the length of the time interval, $T_s = P \cdot R + T_{amb}$ is the steady-state temperature.

B. Energy Model

We adapt the energy model proposed in [18]. Processor's dynamic power can be represented as

$$P_{dyn} = \alpha \cdot C \cdot V_{dd}^2 \cdot f \tag{2}$$

Here V_{dd} is the supply voltage and f is the operation frequency. C is the total capacitance and α is the actual switching activity which varies for different applications [19]. Static power is given by $P_{sta} = V_{dd} \cdot I_{subth} + |V_{bs}| \cdot I_j$ where V_{bs}, I_{subth} and I_j denote the body bias voltage, subthreshold current and reverse bias junction current, respectively. Hence, we have $P = P_{dyn} + P_{sta}$.

C. System Model

The system we consider can be modeled as:

- A voltage scalable processor which supports l discrete voltage levels $\{v_1, v_2, \ldots, v_l\}$
- A set of m independent tasks $\{\tau_1, \tau_2, \ldots, \tau_m\}$.
- Each task $\tau_i \in \{\tau_1, \tau_2, \ldots, \tau_m\}$ has known attributes including worst-case workload, arrival time, deadline, period (if it is periodic) or inter-arrival time (if it is aperiodic/sporadic).

The runtime overhead of voltage scaling is variable and depends on the original and new voltage levels. The context switching overhead is assumed to be constant. For ease of discussion, the terms *task*, *job* and *execution block* refer to the same entity in the rest of this paper.

D. TCEC problem

The methodology described in this paper can be applied to both scenarios in which task set has a common deadline and each task has its own deadline. For ease of discussion, the following definition of TCEC problem is constructed for task sets with a common deadline. The second case will be discussed in Section VII.

Given a trace of m jobs $\{\tau_1, \tau_2, \cdots, \tau_m\}$, where task τ_{i+1} is executed after τ_i ($1 \leq i < m$). If tasks are assumed to have the same power profile (i.e., α is constant), the energy consumption and execution time for τ_i under voltage level v_j, denoted by $w_{i,j}$ and $t_{i,j}$ respectively, can be calculated based on the given processor model. Otherwise, they can be collected through static profiling by executing each task under every voltage level. Let $\psi_{i,j}$ and $\delta_{i,j}$ denote runtime energy and time overhead, respectively, for scaling from voltage v_i to v_j. Since power is constant during an execution block, temperature is monotonically either increasing or decreasing [20]. We denote $T(i)$ as the final temperature of τ_i. If the task set has a common deadline D, the safe temperature threshold is T_{max} and the energy budget is W, TCEC scheduling problem can be defined as follows.

Definition 1: **TCEC instance**: Is there a voltage assignment $\{l_1, l_2, ..., l_m\}$[1] such that:

$$\sum_{i=1}^{m} (t_{i,l_i} + \delta_{l_{i-1},l_i}) \leq D \tag{3}$$

$$\sum_{i=1}^{m} (w_{i,l_i} + \psi_{l_{i-1},l_i}) \leq W \tag{4}$$

$$T(i) \leq T_{max}, \forall i \in 1, ..., m \tag{5}$$

[1] l_i denote the index of the processor voltage level assigned to τ_i.

$T(i)$ is calculated based on Equation (1) for each i, i.e.,

$$T(i) = (1 - \beta_i)T_s^{l_i} + \beta_i T(i-1) \tag{6}$$

where $\beta_i = e^{-t_{i,l_i}/RC}$ (Recall that t_{i,l_i} is the worst case execution time of task τ_i under voltage level l_i), $T(0) = T_{init}$, and $T_s^{l_i}$ is the steady-state temperature of the system, when l_i is applied. Equation (3), (4) and (5) denote the common deadline, energy and temperature constraints, respectively.

When the workload is periodic, we also require the temperature at the end of the hyperperiod to be less than or equal to the initial temperature.

IV. Overview

Figure 1 illustrates the workflow of our approach, which accepts a task execution trace as input. The task execution trace can be produced by a scheduler with certain scheduling policy. The scheduler executes the task set under the highest voltage level and produces a trace of *execution blocks*. In this paper, an execution block is defined as a piece of task execution in a continuous period of time under a single processor voltage/frequency level. The task execution trace, along with system specification (processor voltage, frequency levels, temperature constraints or/and energy budget) and thermal/power models are fed into the timed automata generator (TAG) that we have developed. TAG generates two important outputs. One is the corresponding timed automata model, which will be discussed in Section V-B, and the other one is properties reflecting the temperature/energy/deadline constraints defined in system specification.

After that, a problem solver is applied to find a feasible schedule of the tasks, or confirm that the required constraints cannot be met. Based on the problem size, either a model checker or the approximation algorithm we developed in Section VI can be used to solve the problem. This methodology is flexible and completely automatic.

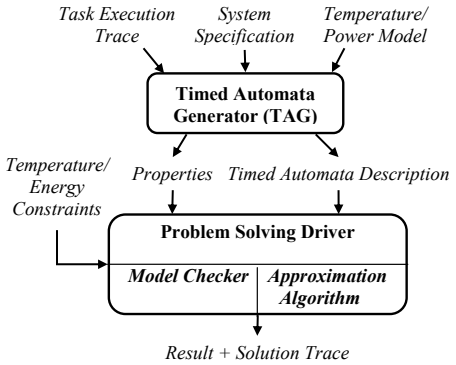

Fig. 1. Overview of our TCEC schedulability framework.

Section V describes a timed automata based model checking framework for TCEC. To address the state space explosion problem in model checking, Section VI proposes an approximation algorithm for TCEC scheduling. Finally, Section VII demonstrates the applicability of our approach to solve other problem variants.

V. TCEC Modeling with Timed Automaton

A. Timed Automata

A classical timed automaton [21] is a finite-state automaton extended with notion of time. A set of clock variables are associated with each timed automaton and elapse uniformly with time in each state (i.e., location). Transitions (i.e., edges) are performed instantaneously from one state to another. Each transition is labeled with a set of guards which are Boolean constraints on clock variables and must be satisfied in order to trigger the transition. Transitions also have a subset of clock variables that need to be reset by taking the transition.

B. Modeling with Extended Timed Automata

Our approach scales the processor voltage level on the granularity of each execution block. In other words, the frequency level is changed at the beginning of each execution block. This strategy can lead to more flexible energy and temperature management in preemptive systems since decisions are made upon a finer granularity compared to inter-task manner [8]. We utilize timed automata to model the voltage scaling problem in the execution trace and extend the original automata with notions of temperature and energy consumption.

For illustration, an extended timed automata $\mathcal{A}$ generated by TAG is shown in Figure 2 assuming that there are two tasks and two voltage levels. Generally, we use l states for each task, forming disjoint sets (horizontal levels of nodes in Figure 2) among tasks, to represent different voltage selections. We also specify an error state (ERROR) which is reached whenever there is a deadline miss. There are also a source state (BEGIN) and a destination state (END) denoting the beginning and the end of the task execution. Therefore, there are totally $(m \cdot l + 3)$ states[2]. There is a transition from every state of one task to every state of its next task. In other words, the states in neighboring disjoint sets are fully connected. There are also transitions from every task state to the error state. All the states of the last task have transitions to the end state.

The system temperature and cumulative energy consumption are represented by two global variables, named $temp$ and $energy$, respectively. Constants such as execution time $t_{i,j}$, energy consumption $w_{i,j}$, common deadline D, thermal capacitance/resistance, ambient temperature T_{amb} and initial temperature T_{init} are stored in respective variables. There are two clock variables, **time** and **exec**, which represent the global system time and the local timer for task execution, respectively. Both clock variables are initially set to 0.

The transition from the source state carries a function $initialize()$, which contains updates to initialize all the variables and constants. Each state is associated with an invariant condition, in the form of **exec** $\leq t_{i,j}$, which must be satisfied when the state is active. This invariant represents the fact that the task is still under execution. Each transition between task states carries a pair of guard: **time** $\leq D$ and **exec** $== t_{i,j}$. The former one ensures that the deadline is observed and the latter one actually triggers the transition, reflecting the fact

[2] m blocks with each supporting l voltage levels requires $m \cdot l$ nodes, plus BEGIN, END and ERROR

that the current task has finished execution. Note that the overhead can be incorporated here since we know the start and end voltage level, if they are different. Each transition is also labeled with three important updates. The first one, $temp = \mathit{calcTemp}(temp)$, basically updates the current system temperature after execution of one task based on the previous temperature, average power consumption and the task's execution time. The second one, $energy = \mathit{calcEnergy}(energy)$, adds the energy consumed by last task to $energy$. The third update resets clock **exec** to 0. All the transitions to the error state are labeled with a guard in the form of **time** $> D$, which triggers the transition whenever the deadline is missed during task execution. Note that not all the transition labels are shown in Figure 2.

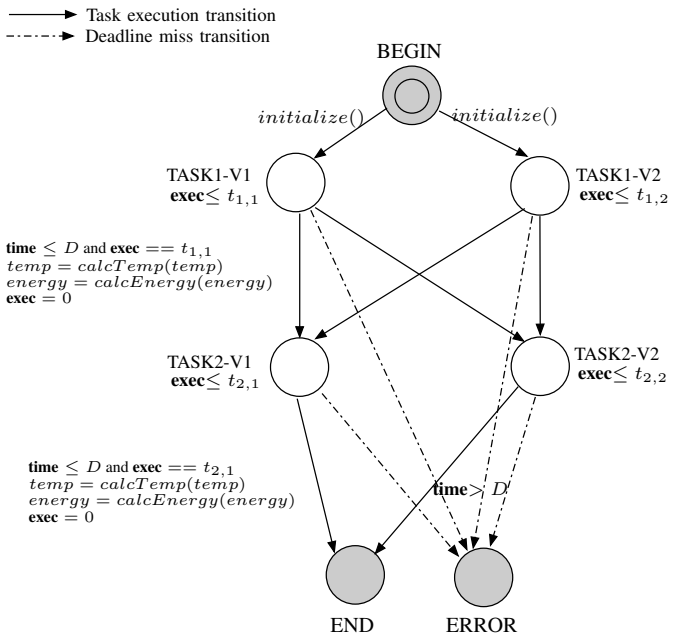

Fig. 2. TCEC problem modeled in extended timed automata.

To solve the TCEC problem as formulated above, we need to find a state sequence with the following properties. First, the final state is the destination state which guarantees the deadline constraint. Next, the temperature $temp$ is always below T_{max} in every state. Finally, the energy consumption $energy$ is no larger than W. We can write this requirement as a property in computation tree logic (CTL) as:

$$\mathbf{EG}((temp < T_{max} \ \wedge \ energy < W) \ \mathbf{U} \ \mathcal{A}.end) \quad (7)$$

where $\mathcal{A}.end$ means the destination state is reached. Now, we can use the model checker to verify this property and, if satisfied, the witness trace it produces is exactly the TCEC schedule that we want.

Timed automata can be used to model the TCEC problem effectively [22]. However, when the number of jobs is large, it can be time consuming to check the properties on the timed automata directly. The reason is that the underlying symbolic model checker sometimes cannot handle large problems due to the state space explosion problem. Therefore, a different algorithmic solution is desired to handle large scale TCEC problems.

VI. Approximation algorithm for TCEC scheduling

To alleviate the state explosion problem in TCEC scheduling, we can formulate our model checking problem as a Multi-Constrained Path problem (MCP). Although MCP is NP-Complete for more than one constraints [23], we are able to design polynomial time approximation scheme which can be tuned with enough accuracy for practical design usage. In this section, we first explain how to model TCEC problem as MCP. Next, we present our polynomial time approximation algorithm for TCEC.

A. Notations

Given a directed graph $G = (V, E)$, a path $p = s \rightarrow n_1 \rightarrow \cdots \rightarrow n_i$ and an edge $e_i = (n_i, n_{i+1}) \in E$, where $s, n_1, \cdots, n_i \in V$, the notation $p||e_i$ denotes the path $s \rightarrow n_1 \rightarrow \cdots \rightarrow n_i \rightarrow n_{i+1}$. In other words, p can also be expressed as $e_0||e_1|| \cdots ||e_i$, where $e_0 = (s, n_1)$, $e_1 = (n_1, n_2)$,$\cdots$,$e_i = (n_i, n_{i+1})$.

Given vectors $\boldsymbol{a}, \boldsymbol{b} \in R^N$, we say that $\boldsymbol{a}$ is dominated by $\boldsymbol{b}$, or $\boldsymbol{a} \leq \boldsymbol{b}$, iff each component of $\boldsymbol{a}$ is smaller or equal to the corresponding component in $\boldsymbol{b}$. For a vector $\boldsymbol{a}$, we use a_1, a_2, a_3 to denote the first, second and third component of $\boldsymbol{a}$.

B. TCEC as MCP

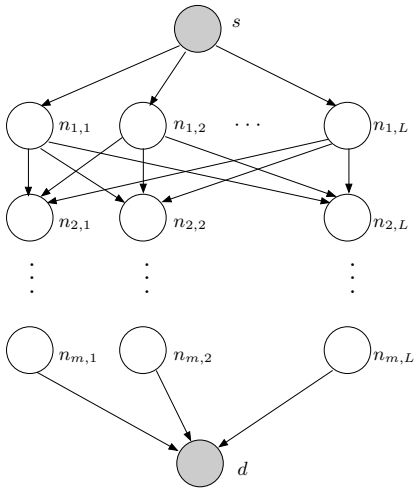

Fig. 3. Job execution graph

An instance TCEC can be reduced to an instance of MCP, if we view the execution jobs at different voltage levels as a path in job execution graph (JEG). As shown in Figure 3, a JEG contains a source node s, a destination node d, and m layers of job (task) nodes. In each layer, there are l nodes for each voltage level. Edges only exist between different layers of job nodes, or job nodes and source/destination nodes. Formally, we define JEG as follows.

Definition 2: **Job execution graph (JEG)** is an acyclic directed graph $G = (V, E)$ with following properties: $V = \{s, d\} \bigcup \{n_{i,j} | 1 \leq i \leq m, 1 \leq j \leq l\}$; $E = \{(s, n_{1,j}) | 1 \leq$

$j \leq l\} \bigcup \{(n_{m,j}, d) | 1 \leq j \leq l\} \bigcup \{(n_{i,j}, n_{i+1,j'}) | 1 \leq i < m, 1 \leq j, j' \leq l\}$.

In order to calculate the values of time, energy and temperature on JEG, we recursively define path transfer functions for path $p = e_0||e_1||\cdots||e_{i-1}||e_i$ $(1 \leq i \leq m)$ from s to $n_{i,j}$ as:

$$f_t^p(t_0) = f_t^q(t_0) + t_{i,j} + \delta(j', j) \tag{8}$$

$$f_w^p(w_0) = f_w^q(w_0) + w_{i,j} + \psi(j', j) \tag{9}$$

$$\begin{aligned} f_T^p(T_0) =& \beta \cdot f_T^q(T_0) + (1 - \beta) \cdot T_s, \\ & \beta = e^{-t_{i,j}/RC} \end{aligned} \tag{10}$$

where $q = e_0||e_1||\cdots||e_{i-1}$ is a prefix of p, which starts from s and ends at $n_{i-1,j'}$. For $p = e_0 = (s, n_{1,j})$, $f_t^p(t_0) = t_0$, $f_w^p(w_0) = w_0$, $f_T^p(T_0) = T_0$, where t_0, w_0 and T_0 are the time, energy consumption and temperature before the execution of the task set. Normally, we have $t_0 = w_0 = 0$ and $T_0 = T_{init}$. We can also write the path transfer functions in vector form

$$\boldsymbol{f}^p(\boldsymbol{I}) = [f_t^p(t_0) f_w^p(w_0) f_T^p(T_0)]^T \tag{11}$$

where $\boldsymbol{I} = [t_0 \quad w_0 \quad T_0]^T$.

Using the above definition, the value of time, energy consumption and temperature of first i jobs with voltage assignment $\{j_1, j_2, \cdots, j_i\}$ can be expressed as $\boldsymbol{f}^p(\boldsymbol{I})$, where $p = s \rightarrow n_{1,j_1} \rightarrow \cdots \rightarrow n_{i,j_i}$. We use the example in Figure 4

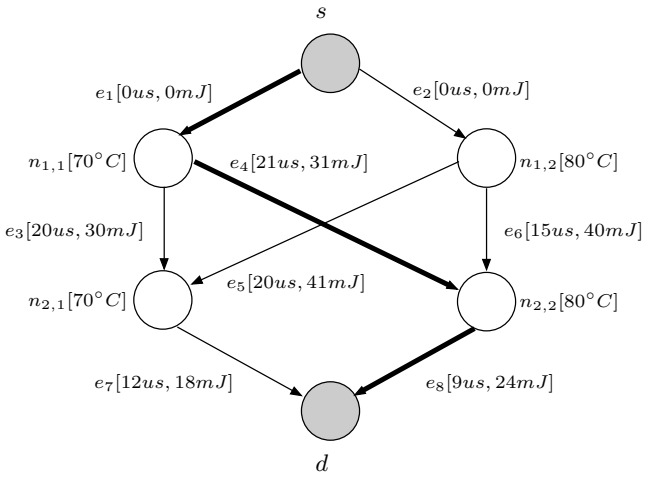

Fig. 4. JEG of TCEC. The values next to each edge are corresponding time and energy consumption.

to illustrate such computation in practice. In this case, we have $m = 2$ jobs and $l = 2$ voltage levels. Suppose that the initial temperature $T_0 = 65°C$ and constant $RC = 30us$. The design constraints are deadline $D = 32us$, energy budget $W = 55mJ$ and maximum temperature $T_{max} = 75°C$. Assume that we decide to use voltage level 1 and 2 to execute job 1 and 2 respectively. Based on the definition of JEG, this voltage assignment corresponds to $s - d$ path $p = e_1||e_4||e_8$ (highlighted). The time consumption after the execution of all jobs can therefore be computed as

$$\begin{aligned} f_t^{e_1||e_4||e_8}(0) &= f_t^{e_1||e_4}(0) + 9us = f_t^{e_1}(0) + 21us + 9us \\ &= 0us + 21us + 9us = 30us \end{aligned}$$

Similarly, we can compute the energy consumption of p as

$$f_w^{e_1||e_4||e_8}(0) = 0mW + 31mJ + 24mJ = 55mJ$$

and the final temperature of p as

$$\begin{aligned} f_T^{e_1||e_4||e_8}(0) = (e^{-\frac{9}{30}}(e^{-\frac{21}{30}} \cdot 65°C + (1 - e^{-\frac{21}{30}}) \cdot 70°C) \\ + (1 - e^{-\frac{9}{30}}) \cdot 80°C) = 70.8°C \end{aligned}$$

In other words, our schedule or path p satisfies the constraints $D = 32us$, $W = 55mJ$ and $T_{max} = 75°C$.

Clearly, the model checking problem discussed in Section V can be answered by checking whether there exists a path p, such that $\boldsymbol{f}^p(\boldsymbol{I}) \leq \boldsymbol{C}$, where $\boldsymbol{C} = [D \quad W \quad T_{max}]^T$. The formal definition of our MCP problem is as follows.

Definition 3: $MCP(G, \boldsymbol{I}, \boldsymbol{C})$ **instance**: Given a job execution graph G, an initial state vector $\boldsymbol{I} = [t_0, w_0, T_0]^T$, a constraint vector $\boldsymbol{C} = [D, W, T_{max}]^T$, is there an $s - d$ path $p = e_0||...||e_m$ such that for all $0 \leq i \leq m$

$$\boldsymbol{f}^{e_0||\cdots||e_i}(I) \leq \boldsymbol{C}$$

The definition above seems to be tighter than the definition of TCEC given in Section V-B, because all constraints are enforced after each job, while the deadline and energy constraint are enforced only after the last job in TCEC. However, they are essentially equivalent due to monotonic nature of execution time and energy consumption.

In the rest of the paper, we will use MCP to present $MCP(G, \boldsymbol{I}, \boldsymbol{C})$ for ease of illustration. Our definition of MCP differs from Quality of Service (QoS) MCP problems [16], [15] in networking, because the computation of the temperature is not additive. As a result, the existing techniques can not be applied directly to solve our problem.

C. Approximation Algorithm

Before we introduce our approximation scheme for MCP, we first present another problem MCP_ϵ, which is closely related to MCP.

Definition 4: $MCP_\epsilon(G, \boldsymbol{I}, \boldsymbol{C})$ **instance**: Given a positive constant $\epsilon > 0$, a job execution graph G; an initial state vector $\boldsymbol{I} = [t_0, w_0, T_0]^T$; a constraint vector $\boldsymbol{C} = [D, W, T_{max}]^T$, there exists an $s - d$ path $p = e_m||...||e_0$ such that for all $0 \leq i \leq m$

$$\begin{aligned} f_t^{e_0||\cdots||e_i}(t_0) &\leq D \\ f_w^{e_0||\cdots||e_i}(w_0) &\leq (1 - \epsilon)W \\ f_T^{e_0||\cdots||e_i}(T_0) &\leq (1 - \epsilon)T_{max} \end{aligned}$$

MCP_ϵ is tighter than MCP. Any $s - d$ path that satisfies the constraints in MCP_ϵ also satisfies the constraints in MCP, but not vice versa. In this section, we are going to develop an approximation algorithm EBF_ϵ to MCP, such that 1) EBF_ϵ is true implies MCP is true, and 2) EBF_ϵ is false implies MCP_ϵ is false. In other words, EBF_ϵ gives no false positive answer to MCP. It may give false negative answer when the exact answers to MCP and MCP_ϵ are true and false respectively (i.e., there are feasible paths for MCP, but no feasible path for MCP_ϵ). Since MCP_ϵ becomes MCP when $\epsilon = 0$, EBF_ϵ will be more and more accurate when $\epsilon \rightarrow 0$.

In a JEG $G = (V, E)$, we define functions h_1, h_2, h_3 on each edge to simplify the description of our approximation scheme. Here, h_1, h_2, and h_3 corresponds to the functions related to the functions of time, energy and temperature, respectively. For $e = (s, n_{1,j}) \in E$ $(1 \leq j \leq l)$, we define

$$h_1^e(x_1) = x_1 \tag{12}$$
$$h_2^e(x_2) = x_2 \tag{13}$$
$$h_3^e(x_3) = x_3 \tag{14}$$

For other $e \in E$,

$$h_1^e(x_1) = x_1 + t_{i,j} + \delta(j', j) \tag{15}$$
$$h_2^e(x_2) = x_2 + w_{i,j} + \psi(j', j) \tag{16}$$
$$h_3^e(x_3) = \beta \cdot x_3 + (1 - \beta) \cdot T_s, \tag{17}$$
$$\beta = e^{-t_{i,j}/RC} \tag{18}$$

Based on the definition of path transfer functions, it is easy to see that for path $p = e_0||\cdots||e_i$,

$$f_t^p(t_0) = h_1^{e_i} \circ \cdots \circ h_1^{e_0}(t_0)$$
$$f_w^p(w_0) = h_2^{e_i} \circ \cdots \circ h_2^{e_0}(w_0)$$
$$f_T^p(T_0) = h_3^{e_i} \circ \cdots \circ h_3^{e_0}(T_0)$$

where $\circ$ is the composition operation for successive invocation functions.

The basic idea of our approximation scheme is to build a table Z_n for each node n. Each cell in this table holds the least value of time consumption among all execution paths, which have the same energy and temperature value after scaling. In other words, each cell represents an optimal execution path. Dynamic programming is then applied to fill each Z_n. The approximated solution can be obtained by checking Z_d, which holds the approximated least time consumption of all possible execution paths.

Algorithm 1 shows the details of our approximation algorithm EBF_ϵ. Initially, we compute the table size M and the "step size" Δ_k for each constraint based on the value of ϵ (line 1 and 2 of EBF_ϵ), and then initialize $M*M$ tables Z_n for each node in G. Here, the "step size" Δ_k is used to scale the energy and temperature values as indices in the table. For example, cell $(\lceil I_2/\Delta_2 \rceil, \lceil I_3/\Delta_3 \rceil)$ in Z_s holds the time consumption before we execute any jobs, which is initialized as 0 in line 7. The rest of EBF_ϵ is similar to EBF. We use dynamic programming to fill each Z_n by calling $Relax_\epsilon$, which can be viewed as a scaled version of $Relax$. In $Relax_\epsilon(u, v)$, we traverse Z_u to fill Z_v by extending paths in Z_u. Since Z_u is an M by M table, we use c_2 and $c_3 \in \{0, 1, .., M\}$ as index variables (line 1). As we have discussed previously, each cell $Z_u(c_2, c_3)$ represents an execution path from s to u with time consumption $Z_u(c_2, c_3)$, energy consumption $c_2 * \Delta_2$ and temperature $c_3 * \Delta_3$ [3]. In line 2 of $Relax_\epsilon$, we first check whether the energy and temperature constraints are violated if the job is executed based on edge (u, v). If no violation occurs, we calculate the scaled version of the new energy and temperature values (b_2, b_3). After that, we compare the new time consumption $Z_{new} = h_1^{(u,v)}(Z_u(c_2, c_3))$ with the current value in $Z_v(b_2, b_3)$ and update Z_v when necessary. If we already reach destination d and the time consumption Z_{new} is still less than the required value C_1 [4], we have found the required schedule. Compared with $Relax$, $Relax_\epsilon$ does not store the paths explicitly as $Path(v)$ in $Relax$, but implicitly in different cells within each table.

Algorithm 1 .

$EBF_\epsilon(G, \boldsymbol{I}, \boldsymbol{C})$

1: $M = \lceil (m+1)/\epsilon \rceil$
2: $\Delta_k = \epsilon * C_k/(m+1), k = 2, 3$
3: **for** each $v \in G$ **do**
4: **for** each $(c_2, c_3) \in \{0, 1, .., M\}^2$ **do**
5: $Z_v(c_2, c_3) = \infty$
6: $\pi_v(c_2, c_3) = null$
7: $Z_s(\lceil I_2/\Delta_2 \rceil, \lceil I_3/\Delta_3 \rceil) = 0$
8: **for** $i = 1$ to $|m|$ **do**
9: **for** $j = 0$ to $|l|$ **do**
10: **for** each edge $(u, n_{i,j}) \in E$ **do**
11: $Relax_\epsilon(u, n_{i,j})$
12: **for** each edge $(u, d) \in E$ **do**
13: **if** $Relax_\epsilon(u, d)$ **then**
14: **return** $TRUE$
15: **return** $FALSE$

$Relax_\epsilon(u, v)$

1: **for** each $(c_2, c_3) \in \{0, 1, .., M\} \times \{0, 1, .., M\}$ **do**
2: **if** $h_k^{(u,v)}(c_k * \Delta_k) \leq C_k$ for $k = 2, 3$ **then**
3: $b_k = \lceil h_k^{(u,v)}(c_k * \Delta_k)/\Delta_k \rceil, k = 2, 3$
4: $Z_{new} = h_1^{(u,v)}(Z_u(c_2, c_3))$
5: **if** $Z_{new} < Z_v(b_2, b_3)$ and $Z_{new} \leq C_1$ **then**
6: $Z_v(b_2, b_3) = Z_{new}$
7: $\pi_v(b_2, b_3) = (u, c_2, c_3)$
8: **if** $v = d$ **then**
9: **return** $TRUE$
10: **return** $FALSE$;

EBF_ϵ is a polynomial time algorithm for a given ϵ, because the complexity of $Relax_\epsilon$ is M^2 or $(m/\epsilon)^2$. $Relax_\epsilon$ is executed for $m \cdot l$ times. Therefore, the overall time complexity is $O(m \cdot l \cdot (m/\epsilon)^2)$. Now, we show that EBF_ϵ is a polynomial time algorithm with the approximation properties as claimed by the following two theorems.

Theorem 6.1: Given an instance of $MCP(G, \boldsymbol{I}, \boldsymbol{C})$, if $EBF_\epsilon(G, \boldsymbol{I}, \boldsymbol{C})$ returns $TRUE$, $MCP(G, \boldsymbol{I}, \boldsymbol{C})$ is true.

Proof: When EBF_ϵ returns $TRUE$, let the path $p = e_0||...||e_m$ be the path constructed by tracing back using table π. Clearly, p is a $s - d$ path. We need to show that for all $0 \leq i \leq m$

$$h_k^{e_i} \circ ... \circ h_k^{e_0}(I_k) \leq C_k, k = 1, 2, 3 \tag{19}$$

[3] Recall that indices in table are scaled version of energy and temperature values. We can obtain the actual energy and temperature values by multiplying table indices with Δ_2 and Δ_3.

[4] C_1, C_2, and C_3 are the constraints for time, energy, and temperature, respectively.

Clearly, p satisfies Equation (19) for $k = 1$, because the condition on line 5 of $Relax_\epsilon$ guarantees that

$$h_1^{e_i} \circ ... \circ h_1^{e_0}(I_1) \leq C_1, 0 \leq i \leq m$$

Since p is constructed by π, it is easy to see that

$$C_k \geq h_k^{e_0}(\lceil I_k/\Delta_k \rceil * \Delta_k), k = 2, 3$$

Otherwise, condition on line 2 of $Relax_\epsilon$ would not be satisfied during $Relax_\epsilon(e_0)$, and line 7 of $Relax_\epsilon$ would not be executed. This contradicts the fact that e_0 is recorded in π.

Similarly, for $0 \leq i \leq m$ and $k = 2, 3$, we have

$$C_k \geq h_k^{e_1}(\lceil h_k^{e_0}(\lceil I_k/\Delta_k \rceil * \Delta_k)/\Delta_k \rceil * \Delta_k)$$
$$...$$
$$C_k \geq h_k^{e_i}(\lceil ...\lceil h_k^{e_0}(\lceil I_k/\Delta_k \rceil * \Delta_k)/\Delta_k \rceil * \Delta_k .../\Delta_k \rceil * \Delta_k)$$

Or

$$C_k \geq h_k^{e_i} \circ g_k \circ ... \circ h^{e_1} \circ g_k \circ h_k^{e_0} \circ g_k(I_k)$$

where g_k is a "ceiling" function

$$g_k(x) = \lceil x/\Delta_k \rceil * \Delta_k$$

Since $h_k^{e_i}$ and g_k are monotonically increasing functions and $g_k(x) \geq x$, we have following relations

$$\begin{aligned} \lceil I_k/\Delta_k \rceil * \Delta_k = g_k(I_k) &\geq I_k \\ h_k^{e_0} \circ g_k(I_k) &\geq h_k^{e_0}(I_k) \\ h_k^{e_1} \circ g_k \circ h_k^{e_0} \circ g_k(I_k) &\geq h_k^{e_1} \circ h_k^{e_0}(I_k) \\ ... & \\ h_k^{e_m} \circ g_k \circ ... \circ h_k^{e_0} \circ g_k(I_k) &\geq h_k^{e_m} \circ ... \circ h_k^{e_0}(I_k) \end{aligned}$$

Thus, for $0 \leq i \leq m$

$$\begin{aligned} C_k &\geq h_k^{e_i} \circ g_k \circ ... \circ h_k^{e_1} \circ g_k \circ h_k^{e_0} \circ g_k(I_k) \\ &\geq h_k^{e_i} \circ ... \circ h_k^{e_0}(I_k) \end{aligned}$$

Therefore, Equation (19) also holds on p for $k = 2, 3$. By the definition of MCP, $MCP(G, \boldsymbol{I}, \boldsymbol{C})$ is true. ■

Lemma 6.1.1: Given an instance of $MCP_\epsilon(G, \boldsymbol{I}, \boldsymbol{C})$, if there is an $s - d$ path $p = e_0||...||e_{m-1}||e_m$ such that

$$h_1^{e_i} \circ ... \circ h_1^{e_0}(I_1) \leq C_1 \tag{20}$$
$$h_k^{e_i} \circ g_k \circ ... \circ h_k^{e_0} \circ g_k(I_k) \leq C_k, k = 2, 3 \tag{21}$$

holds for $0 \leq i \leq m$, EBF_ϵ will return TRUE.

Lemma 6.1.1 can be proven by considering the following fact that if we only perform $Relax_\epsilon$ on edges that are in p, Equation (21) and Equation (20) guarantees that the conditions on line 2 and 5 in $Relax_\epsilon$ are satisfied and line 6 will be executed in each round. Eventually, EBF_ϵ will return true. If we perform $Relax_\epsilon$ on more edges, the minimal value in Z_d will not increase. As a result, EBF_ϵ still returns true.

Theorem 6.2: Given an instance of $MCP_\epsilon(G, \boldsymbol{I}, \boldsymbol{C})$, $MCP_\epsilon(G, \boldsymbol{I}, \boldsymbol{C})$ is true implies $EBF_\epsilon(G, \boldsymbol{I}, \boldsymbol{C})$ returns TRUE.

Proof: We just need to show that if there is an $s - d$ path

$$\begin{aligned} p &= s \to n_{1,j_1} \to \cdots \to n_{m,j_m} \to d \\ &= e_0||...||e_{m-1}||e_m \end{aligned}$$

such that for all $0 \leq i \leq m$

$$h_1^{e_i} \circ ... \circ h_1^{e_0}(I_1) \leq C_1 \tag{22}$$
$$h_2^{e_i} \circ ... \circ h_2^{e_0}(I_2) \leq (1 - \epsilon)C_2, k = 2, 3 \tag{23}$$

it also satisfies Equation (20) and (21).

Clearly, for any edge $e \in E$

$$h_2^e(c + \Delta) \leq h_2^e(c) + \Delta$$

For h_3^e, which represents the temperature constraints, we have

$$h_3^e(c + \Delta) = h_3^e(c) + \Delta * \beta \leq h_3^e(c) + \Delta$$

because $\beta = e^{\frac{-t}{RC}} \leq 1$.

Using ceiling functions $g_k(x) = \lceil x/\Delta_k \rceil * \Delta_k, k = 2, 3$, it is easy to verify

$$g_k(I_k) \leq I_k + \Delta_k$$

By applying $h_k^{e_i}$ on its both sides, we have

$$h_k^{e_0} \circ g_k(I_k) \leq h_k^{e_0}(I_k + \Delta_k) \leq h_k^{e_0}(I_k) + \Delta_k, k = 2, 3$$

because $h_k^{e_0}$ is a monotonic function. Therefore,

$$\begin{aligned} h_k^{e_0} \circ g_k(I_k) &\leq h_k^{e_0}(I_k) + \Delta_k, \\ g_k \circ h_k^{e_0} \circ g_k(I_k) &\leq h_k^{e_0}(I_k) + 2\Delta_k, \\ h_k^{e_1} \circ g_k \circ h_k^{e_0} \circ g_k(I_k) &\leq h^{e_1} \circ h_k^{e_0}(I_k) + 2\Delta_k \\ & ... \\ h_k^{e_m} \circ g_k \circ ... \circ h_k^{e_0} \circ g_k(I_k) &\leq h_k^{e_m} \circ ... \circ h_k^{e_0}(I_k) + (m + 1) * \Delta_k \end{aligned}$$

From Equation (23), we know that

$$h_k^{e_i} \circ ... \circ h_k^{e_0}(I_k) \leq (1 - \epsilon) * C_k, k = 2, 3$$

Thus, for $0 \leq i \leq m$ $k = 2, 3$, we have

$$\begin{aligned} h_k^{e_i} \circ g_k \circ ... \circ h_k^{e_0} \circ g_k(I_k) &\leq (1 - \epsilon) * C_k + (m + 1) * \Delta_k \\ &\leq (1 - \epsilon) * C_k + \epsilon * C_k = C_k \end{aligned}$$

Therefore, p satisfies Equation (20) and Equation (21). Using Lemma 6.1.1, EBF_ϵ will return true. ■

VII. Problem Variants

Our approach is also applicable to other problem variants by modifying the property and making suitable changes to invocation of the problem solving driver in Figure 1 (model checker and approximation algorithm).

Task set with individual deadlines: In the scenario where each task has its own deadline, we have to make sure that the execution blocks finish no later than their corresponding task's deadline. Suppose that the deadline of the i^{th} execution block is $D[i]$. Equation (3) is replaced by following constraints, for all $1 \leq i \leq m$:

$$\sum_{i=1}^{m} t_{i,l_i} + \delta_{l_{i-1},l_i} \leq D[i], \forall D[i] > 0 \tag{24}$$

Our timed automata model can be easily modified with the following changes in the guard of transitions: Instead of **time** $\leq D$, the guard for transitions between task states is in the form of (**time** $\leq D[\,]$). The transition from task state to error state now carries a guard of (**time** $> D[\,]$).

The approximation algorithm EBF_ϵ can also be modified sightly to the individual deadline case. We only need to replace C_1 (line 5 in $Relax_\epsilon$) with $D[i]$, when node u represents job i. Since the approximation is applied on the energy and temperature constraints, all the properties and related proofs of EBF_ϵ still hold.

Periodic Tasks: We solve the TCEC scheduling of periodic tasks by considering the scheduling of tasks within a hyperperiod. We have to make sure that a) all tasks meets their corresponding deadlines in every hyper-period, b) the temperature constraints are not violated after execution of any hyperperiod. Clearly, the first requirement can be achieved by adding the deadline constraints as we discussed in task set with individual deadlines. The second requirement is satisfied by only choosing the schedules, whose temperature at the end of the hyper-period is less than or equal to the initial temperature, as discussed in Section III-D.

Our timed automata model needs to be modified by setting the temperature at the end of the hyper-period $T(m_h)$ is less than the initial temperature T_{init}. We also need to modify the approximation algorithm EBF_ϵ. When $Relax_\epsilon$ is applied to the node corresponding to the last task, we need to ensure that $h_3^{(u,v)}(c_3 * \Delta_3) \leq T_{init}$ (line 2 in $Relax_\epsilon$). In addition, the step size of temperature should be calculated based on T_{init}, i.e., $\Delta_3 = \epsilon * T_{init}/(m+1)$ (line 2 of EBF_ϵ). we verified that all the properties and related proofs of EBF_ϵ still hold.

VIII. Experiments

A. Experimental Setup

In this section, we describe the experimental setup for evaluation of our approach. A DVS-capable processor StrongARM [24] is modeled with four voltage/frequency levels (1.5V-206MHz, 1.4V-192Mhz, 1.2V-162MHz and 1.1V-133MHz). We use synthetic task sets which are randomly generated with each of them having execution time in the range of 100 - 500 milliseconds. These are suitable and practical sizes to reflect variations in temperature, and millisecond is a reasonable time unit granularity [8]. We adopt the thermal resistance (R) and thermal capacitance (C) values from [20], which are $1.83°C/Watt$ and $112.2mJoules/°C$, respectively. The ambient temperature of the processor is $32°C$. The scheduler and TAG shown in Figure 1 are implemented in C++. The exact algorithm EBF and the approximation algorithm EBF_ϵ are also implemented in C++. All experiments are performed on a computer with AMD64 2GHz CPU and 16G RAM.

B. TCEC versus TC or EC

This section demonstrates that existing solutions based on TC or EC are not sufficient to find TCEC schedules. We compared the schedule generated by energy constrained scheduling algorithm [25] and our TCEC scheduling for the same set of jobs under the same energy constraint. We also require that the system temperature after the execution of task set does not exceed the initial temperature T_{init}, so that the temperature constraint is not violated even if the task set is executed repeatedly. The results are shown in Figure 5.

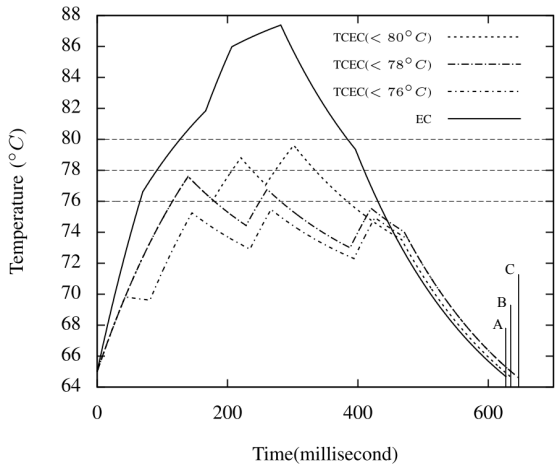

Fig. 5. EC vs TCEC. EC finishes at A. TCEC($< 80°C$) finishes at B. Both TCEC($< 78°C$) and TCEC($< 76°C$) finish at C. The execution time of both algorithms are 0.05s and 0.07s, respectively.

It can be seen that the schedule generated by [25], which considers only energy constraint takes less execution time. However, it violates temperature constraint. On the other hand, the schedules generated by our TCEC approach will not exceed the respective temperature constraints ($80°C$, $78°C$ and $76°C$, respectively), although it takes a little longer execution time. Therefore, scheduling algorithms that consider only energy constraint are not suitable, when we want to control the maximum temperature of the processor during job execution.

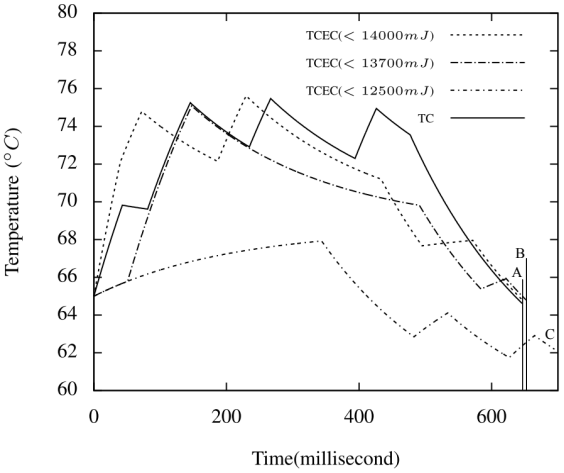

Fig. 6. TC vs TCEC. Both TC and TCEC($< 14000mJ$) finish at A. TCEC($< 13700mJ$) finishes at B. TCEC($< 12500mJ$) finishes at C. The execution time of both algorithms are 0.04s and 0.07s, respectively.

We also compared our TCEC scheduling with temperature constrained scheduling algorithm [8]. The experiments were performed on the same job set with the same temperature constraints. For TCEC, we applied three different energy constraints. We also require that the system temperature after the execution of task set does not exceed the initial temperature T_{init}. Figure 6 presents the results. Since TC has no constraint on energy consumption, it always tries to execute jobs with high voltage, which may lead to peak temperature several times. As a result, TC has the shortest execution time.

However, once we consider energy constraint, it may not be possible to execute some jobs at high voltage. When the energy budget is very tight, we may not be able to reach the maximum temperature during the entire execution, like curve "TCEC($<$12500mJ)" in Figure 6. In this case, TC will clearly violate the energy constraint, while our TCEC obtains a schedule within the energy budget.

C. TCEC using Approximation Algorithm

Our approximation algorithm should be used when the model checker (UPPAAL) cannot find a solution in reasonable time due to state space explosion. We compared the efficiency of conventional symbolic model checker (UPPAAL) with our approximation algorithm EBF_ϵ on task sets with different number of blocks. The results are shown in Table I. The first and the second column are the index and number of blocks in each task set, respectively. The next three columns present the temperature constraint (TC, in $^\circ C$), energy constraint (EC, in mJ), and deadlines (DL, in ms) to be checked on the model. The sixth column indicates whether there exists a schedule which satisfies all the constraints. The last three columns of Table I shows the results (running time in seconds) of UPPAAL, and our approach EBF_ϵ with $\epsilon = 0.02$. Since UPPAAL failed to produce result for task set 4 and 5, we only report the running time of EBF_ϵ. It can be seen that EBF_ϵ outperforms UPPAAL by more than 10 times on average. Moreover, EBF_ϵ can solve much larger problems in reasonable running time.

TABLE I
RUNNING TIME COMPARISON ON DIFFERENT TASK SETS

TS	#Blk	TC	EC	DL	Found?	UPPAAL	EBF_ϵ
1	10	85	180000	7000	Y	9.6	0.2
		85	150000	8000	Y	9.9	0.2
		80	140000	8000	N	9.4	0.2
2	12	85	70000	2500	Y	18.5	0.3
		85	60000	2700	Y	106.6	0.3
		80	60000	2500	N	17.5	0.3
3	14	90	90000	2600	Y	65.1	0.3
		85	80000	2800	Y	648.3	0.3
		90	80000	2700	N	208.6	0.3
4	50	85	380000	39500	Y	-	20.2
5	100	85	720000	83800	Y	-	102.5

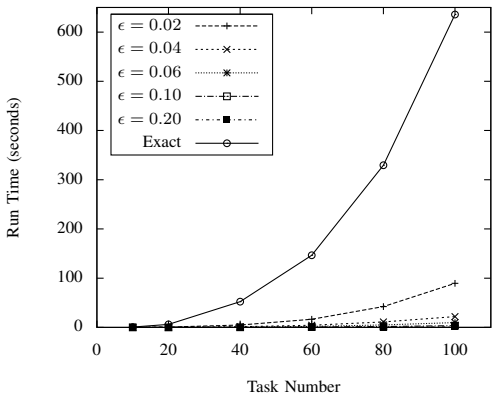

Fig. 7. Running time with different job set size and ϵ.

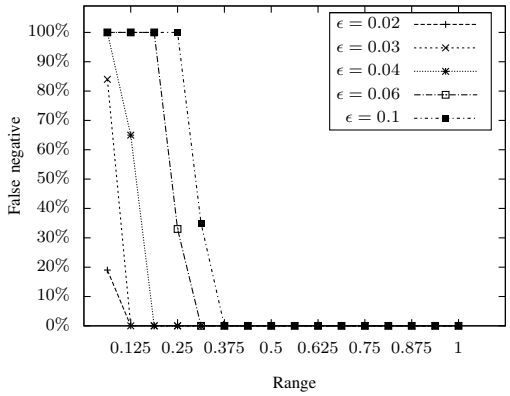

Fig. 8. Accuracy of EBF_ϵ.

We also evaluated the running time of our approximation scheme with different ϵ. The results are shown in Figure 7. Curve "Exact" represents the execution time of the exact algorithm EBF. Other curves present the running time of EBF_ϵ with different ϵ. As expected EBF_ϵ requires more time for smaller ϵ or larger job set size. But its time consumption is sill much smaller than the exact algorithm EBF.

To investigate the accuracy of our proposed approximation scheme, we evaluated the distribution of false negative ratio along different constraint values. In this experiment, we generated 1500 instance of TCEC problem as the test set. They share the same deadline and energy budget, while the temperature constraints are uniformly distributed within $1^\circ C$ above the lowest feasible temperature. For each instance, the exact algorithm EBF is applied first to determine whether the feasible schedule exists. Then we run EBF_ϵ on each instance and check the correctness of the return value. The experimental results are presented in Figure 8. Each point represent the false negative ratio of TCEC instances in each $0.0625^\circ C$ interval. For example, the false negative ratio is 30% for instances with in interval $[0.1875\ 0.25]$ when $\epsilon = 0.06$. As we discussed in Section VI-C, the false negative ratio curves behaves as step functions, which fall to zero when the temperature constraint is slightly larger ($0.125^\circ C$ for $\epsilon = 0.02$) than the lowest feasible temperature. In other words, EBF_ϵ produces false negative answers in rare cases.

IX. CONCLUSION

In this paper, we proposed a flexible and automatic framework to solve the temperature- and energy-constrained scheduling problem in multitasking systems with different voltage levels. We modeled the problem using extended timed automata and translated the energy/temperature constraints into CTL specifications. The user can employ a suitable model checker to determine whether there exists a schedule that satisfies the constraints. Due to the capacity limitations of symbolic model checker like UPPAAL, we also proposed a polynomial time approximation scheme that is guaranteed to generate results close to optimal value with reasonable running time. We proved mathematically that our approximation algorithm will give no false positive answer, while

the false negative ratio can be negligibly small in practical scenarios. Extensive experimental results demonstrated the effectiveness of our approach. In our future work, we plan to develop approximation algorithms to efficiently solve both task sequencing and voltage assignment together.

REFERENCES

[1] M. J. Ellsworth, "Chip power density and module cooling technology projections for the current decade," in *Proc. of Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, vol. 2, 2004, pp. 707–708.

[2] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proc. of Design Automation Conference*, 2003, pp. 338–342.

[3] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur, "Thermal performance challenges from silicon to systems," *Intel Technology Journal*, vol. 4, no. 3, pp. 1–16, 2000.

[4] R. Jejurikar and R. Gupta, "Energy aware non-preemptive scheduling for hard real-time systems," in *Proc. of Euromicro Conference on Real-Time Systems*, 2005, pp. 21–30.

[5] S. Zhang, K. Chatha, and G. Konjevod, "Approximation algorithms for power minimization of earliest deadline first and rate monotonic schedules," in *Proc. of International Symposium on Low Power Electronics and Design*, 2007, pp. 225–230.

[6] W. Wang and P. Mishra, "PreDVS: Preemptive Dynamic Voltage Scaling for Real-time Systems using Approximation Scheme," in *Proc. of Design Automation Conference*, 2010.

[7] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," in *Proc. of Euromicro Conference on Real-Time Systems*, 2006, pp. 10pp.–170.

[8] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature aware scheduling problem," in *Proc. of International Conference on Computer-Aided Design*, 2007, pp. 281–288.

[9] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs," in *Proc. of Design, Automation and Test in Europe*, 2008, pp. 288–293.

[10] A. Coskun, T. Rosing, K. Whisnant, and K. Gross, "Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 16, no. 9, pp. 1127–1140, 2008.

[11] X. Qin, M. Chen, and P. Mishra, "Synchronized generation of directed tests using satisfiability solving," in *Proceedings of International Conference on VLSI Design*, 2010, pp. 351–356.

[12] X. Qin and P. Mishra, "Directed test generation for validation of multicore architectures," *ACM Transactions on Design Automation of Electronic Systems*, 2012.

[13] S. Shukla and R. Gupta, "A model checking approach to evaluating system level dynamic power management policies for embedded systems," in *Proc. of High-Level Design Validation and Test Workshop*, 2001, pp. 53–57.

[14] A. Lungu, P. Bose, D. J. Sorin, S. German, and G. Janssen, "Multicore power management: Ensuring robustness via early-stage formal verification," in *Proc. of International Conference on Formal Methods and Models for Co-Design*, 2009, pp. 78–87.

[15] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," in *Proc. IEEE International Conference on Communications*, vol. 2, 1998, pp. 874 –879 vol.2.

[16] G. Xue, W. Zhang, J. Tang, and K. Thulasiraman, "Polynomial Time Approximation Algorithms for Multi-Constrained QoS Routing," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 656 –669, 2008.

[17] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, 2004.

[18] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proc. of International Conference on Computer Aided Design*, 2002, pp. 721–725.

[19] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Determining optimal processor speeds for periodic real-time tasks with different power characteristics," in *Proc. of Euromicro Conference on Real-Time Systems*, 2001, pp. 225–232.

[20] R. Jayaseelan and T. Mitra, "Temperature aware task sequencing and voltage scaling," in *Proc. of International Conference on Computer-Aided Design*, 2008, pp. 618–623.

[21] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, 1994.

[22] W. Wang, X. Qin, and P. Mishra, "Temperature- and energy-constrained scheduling in multitasking systems: a model checking approach," in *Proc. of the International Symposium on Low Power Electronics and Design*, 2010, pp. 85–90.

[23] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228 –1234, 1996.

[24] Marvell, *Marvell StrongARM 1100 processor*, www.marvell.com.

[25] F. Xie, M. Martonosi, and S. Malik, "Bounds on power savings using runtime dynamic voltage scaling: an exact algorithm and a linear-time heuristic approximation," in *Proc. of International Symposium on Low Power Electronics and Design*, 2005, pp. 287–292.

PLACE PHOTO HERE

Xiaoke Qin (S'08) Xiaoke Qin received the B.S. and M.S. degrees from Department of Automation, Tsinghua University, Beijing, China, in 2004 and 2007 respectively. He is currently a Ph.D. student in the Department of Computer and Information Science and Engineering, University of Florida. His research interests are in the area of model checking and system verification.

PLACE PHOTO HERE

Weixun Wang (S'08) received his B.E. degree from the Software Institute, Nanjing University, China and Ph.D. degree from Department of Computer and Information Science and Engineering, University of Florida. His research interests include the area of design automation of embedded systems with focus on dynamic cache reconfiguration, energy optimization, temperature management, design space exploration and lossless data compression.

PLACE PHOTO HERE

Prabhat Mishra (S'00-M'04-SM'08) received the B.E. degree from Jadavpur University, India, the M.Tech. degree from the Indian Institute of Technology, Kharagpur, and the Ph.D. degree from the University of California, Irvine – all in computer science. He is currently an Associate Professor with the Department of Computer and Information Science and Engineering, University of Florida. His research interests include design automation of embedded systems, and functional verification. He has published two books, nine book chapters and more than 80 research papers in premier journals and conferences. His research has been recognized by several awards including an NSF CAREER Award in 2008, two best paper awards (VLSI Design 2011 and CODES+ISSS 2003), and 2004 EDAA Outstanding Dissertation Award from the European Design Automation Association. Dr. Mishra currently serves as an Associate Editor of ACM Transactions on Design Automation of Electronic Systems, IEEE Design & Test of Computers, Journal of Electronic Testing, Guest Editor of IEEE Transactions of Computers, and as a program/organizing committee member of several ACM and IEEE conferences.