Temperature- and Energy-Constrained Scheduling in Multitasking Systems: A Model Checking Approach*

Weixun Wang, Xiaoke Qin, and Prabhat Mishra
Department of Computer and Information Science and Engineering, University of Florida
Gainesville, FL, USA
{wewang,xqin,prabhat}@cise.ufl.edu

ABSTRACT

The ongoing scaling of semiconductor technology is causing severe increase of on-chip power density and temperature in microprocessors. This has raised urgent requirement for both power and thermal management during each level of system design. In this paper, we propose a formal technique based on model checking using extended timed automata to solve the processor frequency assignment problem in a temperature- and energy- constrained multitasking system. The state space explosion problem is alleviated by transforming and solving a Pseudo-Boolean satisfiability problem. Our approach is capable of finding efficient solutions under various constraints and applicable to other problem variants as well. Our method is independent of any system and task characteristics. Experimental results demonstrate the usefulness of our approach.

Categories and Subject Descriptors

C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Real-time systems and embedded systems

General Terms

Algorithm, Design

Keywords

Temperature-aware, Low Power Design, DVS, Model Checking

1. INTRODUCTION

Along with the performance improvement in state-of-art microprocessors, power densities are rising more rapidly due to the fact that feature size scales faster than voltages [27]. 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 $250W/cm^2$ [12]. 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 [6]. This trend is observed in both desktop and embedded processors [29] [38].

Thermal increase will lead to reliability and performance degradation since CMOS carrier mobility is dependent on the operating temperature. High temperature can result in more frequent transient errors or even permanent damage. Industrial studies have shown that a small difference in operating temperature (10-15°C) can make 2 times difference in the device lifespan [29]. Yeh et al. [35] also estimate that more than half of the electronic failures are caused by over-heated circuits. Furthermore, leakage power is exponentially proportional to temperature, which potentially results in more thermal runaway [34]. Studies also show that cooling cost increases super-linearly with the thermal dissipation [14].

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 limits 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 [9] and temperature management [38]. In existing literatures, ***temperature (energy)- constrained*** means there is a temperature threshold (energy budget) which cannot be exceeded, while ***temperature (energy)- aware*** means 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 problems in multitasking systems. We extend the classical timed automata [1] with notions of task scheduling, voltage scaling, system temperature and energy consumption. To the best of our knowledge, our approach is the first attempt on solving TCEC problem which is meaningful (especially in embedded systems) and as difficult as other problems including temperature-constrained (**TC**) scheduling, temperature-aware (**TA**) scheduling, temperature-constrained energy-aware (**TCEA**) scheduling and energy- constrained temperature-aware (**TAEC**) scheduling. A novel contribution of the paper is the development of a flexible and automatic design flow which models the TCEC problem in timed automata and solves it using formal verification techniques. Our approach is also capable of solving other problem variations mentioned above. Furthermore, our approach is applicable to a wide variety of system and task characteristics. Runtime voltage scaling overhead and leakage power consumption can also be easily incorporated.

The rest of the paper is organized as follows. Section 2 introduces relevant existing research works. Section 3 provides related background information. Section 4 describes our approach in details. Experimental results are presented in Section 5. Finally Section 6 concludes the paper.

*This work was partially supported by NSF grants CCF-0903430 and CNS-0746261.

ISLPED'10, August 18–20, 2010, Austin, Texas, USA.

2. RELATED WORK

Energy-aware scheduling techniques for real-time systems have been widely studied to reduce energy consumption. While several works employed dynamic cache reconfiguration [33] [31], most of them are based on DVS. Aydin et al. [3] addressed both static and dynamic slack allocation problems for periodic task sets, while Shin et al. [25] also considered aperiodic tasks. Jejurikar et al. focused on energy-aware scheduling for non-preemptive task sets [16] and leakage power minimization [17]. Zhong et al. [39] solved a system-wide energy minimization problem with consideration of other components. Wang et al. [31] proposed a leakage-aware energy saving technique based on DVS as well as cache reconfiguration. As shown in [37], applying DVS in real-time systems is a NP-hard problem. Optimal and approximation algorithms are given in [39] [37] [32], while other works proposed heuristics. A survey on recent works can be found in [9]. 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. [30] introduced a simple reactive DVS scheme aiming at meeting task timing constraints and maintaining processor safe temperature. Zhang et al. [38] proved the NP-hardness of temperature-constrained performance optimization problem in real-time systems and proposed an approximation algorithm. Yuan et al [36] considered both temperature and leakage power impact in DVS problem for soft real-time systems. Chen et al. [8] explored temperature-aware scheduling for periodic tasks in both uniprocessor and homogeneous multiprocessor DVS-enabled platforms. Liu et al. [18] proposed a design-time thermal optimization framework which is able to solve problem variants EA, TA and TCEA scheduling in embedded system with task timing constraints. Jayaseelan et al. [15] exploited different task execution orders, in which each task has distinct power profile, to minimize peak temperature. However, none of these techniques solves TCEC problem. Moreover, they all make certain assumptions on system characteristics that limits their applicability.

Timed automata [1] has been widely adapted in real-time system researches. Norstorm et al. [22] first extended timed automata with a notion of real-time tasks and showed that the traditional schedulability analysis can be transformed to a decidable reachability problem in timed automata, which can be solved using model checking tools. Fersman et al. [13] further generalized [22] with asynchronous processes and preemptive tasks in continuous-time model. Abdeddaïm et al. However, none of these techniques considered energy or temperature related issues.

There are several studies on dynamic power management (DPM) using formal verification methods for embedded systems [26] and multiprocessor platforms [19]. Shukla et al. [26] provided a preliminary study on evaluating DPM schemes using an off-the-shelf model checker. Lungo et al. [19] tried to incorporate verification of DPM schemes in the early design stage. They showed that tradeoffs can be made between design quality and verification efforts. None of these approaches considers temperature management in such systems. Moreover, they did not account for energy and timing constraints, which makes our methodology different from theirs.

3. BACKGROUND

3.1 Timed Automata

A classical timed automaton [1] 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. Formally, we can define it as follows:

DEFINITION 3.0.1. *A timed automaton $\mathcal{A}$ over clock set $\mathcal{C}$, state set $\mathcal{S}$ and transition set $\mathcal{T}$ is a tuple $\{\mathcal{S},\mathcal{C},\mathcal{T},s_0\}$ where s_0 is the initial state. Transition set is represented as $\mathcal{T}\subseteq\mathcal{S}\times\Phi(\mathcal{C})\times 2^{\mathcal{C}}\times\mathcal{S}$, where each element ϕ in clock constraint (guard) set $\Phi(\mathcal{C})$ is a conjunction of simple conditions on clocks ($\phi := c\leqslant t\,|\,t\leqslant c\,|\,\neg\phi\,|\,\phi_1\wedge\phi_2$ where $c\in\mathcal{C}, t\in R$). $2^{\mathcal{C}}$ represents the subset of clock variables that will be reset in each transition and we term it as ρ.*

Semantically, the current configuration of a timed automaton $\mathcal{A}$ is decided by a state $s\in\mathcal{S}$ and the clock valuations $\mathcal{V}$ in the form of $\mathcal{C}\rightarrow R_+\bigcup\{0\}$. Therefore, a legal execution of $\mathcal{A}$ consists of a sequence of transitions:

$$(s_0,\mathcal{V}_0)\xrightarrow{\phi,\rho}(s_1,\mathcal{V}_1)\xrightarrow{\phi,\rho}\cdots\xrightarrow{\phi,\rho}(s_n,\mathcal{V}_n) \quad (1)$$

3.2 Thermal Model

A thermal RC circuit is normally utilized to model the temperature variation behavior of a microprocessor [38]. We adopt the RC circuit model proposed in [28], which is widely used in recent researches [38] [15], 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_0 are the ambient and initial temperature, respectively, the temperature at the end of the time interval t can be calculated as:

$$T = P\cdot R + T_{amb} - (P\cdot R + T_{amb} - T_{init})\cdot e^{\frac{-t}{RC}} \quad (2)$$

where t is the length of the time interval. If t is long enough, T will approach a steady-state temperature $T_s = P\cdot R + T_{amb}$.

3.3 Energy Model

We adapt the energy model proposed in [20]. Processor's dynamic power can be represented as $P_{dyn} = \alpha\cdot C\cdot V_{dd}^2\cdot f$. 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 [2]. In other words, task's power profile can be different from each other. 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 $P = P_{dyn} + P_{sta}$. Our technique is, however, independent of the power model and thermal model.

3.4 System Model

The system we consider can be modeled as:

- A voltage scalable processor which supports l discrete voltage levels $V\{v_1, v_2, \ldots, v_l\}$
- A set of m independent tasks $T\{\tau_1, \tau_2, \ldots, \tau_m\}$.
- Each task $\tau_i \in T$ 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.

4. TCEC SCHEDULING APPROACH

4.1 Overview

Figure 1 illustrates the workflow of our approach. The task information describes the characteristics of the tasks running in the system and is fed into the scheduler along with the scheduling policy. Any scheduling algorithm is applicable in our approach. 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. Each execution block is essentially a whole task instance in non-preemptive systems. However, in preemptive scheduling, tasks could be preempted during execution hence one block can be a segment of one task. The scheduler records runtime information for each block including its corresponding task, required workload, arrival time and deadline, if applicable.

The task execution trace, along with system specification (processor voltage/frequency levels), thermal/power models and design objective (not shown in Figure 1), are fed into the timed automata generator (TAG) that we have developed. Here the design objective decides the nature of the problem, e.g. TCEC. TAG generates two important outputs. One is the description of our timed automata model, which will be discussed in Section 4.2, and the other one contains the properties reflecting the design objectives. We use a script based program to drive the model checker to solve the problem. Finally, the results and/or solutions are collected. Our methodology is flexible, completely automatic, based on formal technique and hence suitable in early design stages.

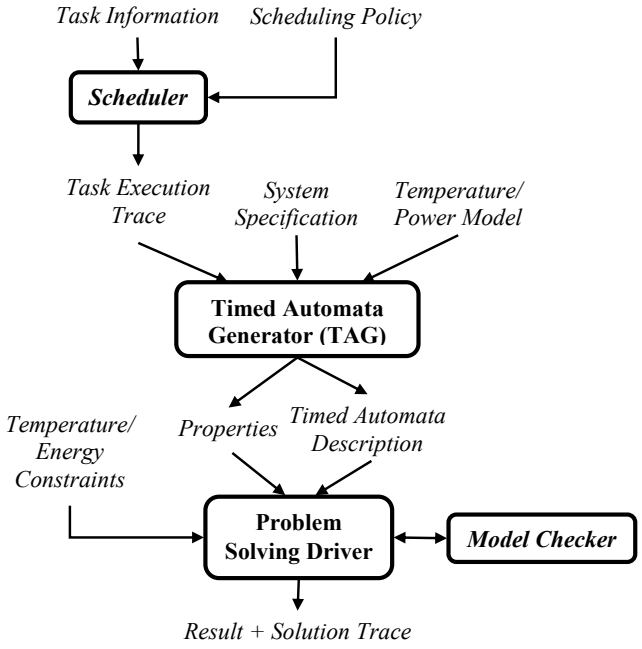

Figure 1: Workflow of our model checking approach.

4.2 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 [38]. 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. Our model supports both scenarios in which task set has a common deadline and each task has its own deadline. For ease of discussion, the terms of *task*, *job* and *execution block* refer to the same entity in the rest of the paper.

Task set with common deadline: TAG is given a trace of n execution blocks $B\{b_1, b_2, ..., b_n\}$. If tasks are assumed to have the same power profile (i.e. α is constant), the energy consumption and execution time for b_i under voltage level $v_k \in V$, denoted by e_i^k and t_i^k 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 ψ_{v_i,v_j} and ω_{v_i,v_j} denote runtime energy and time overhead, respectively, for scaling from voltage v_i to v_j. Since the power is constant during one execution block, the temperature is monotonically either increasing or decreasing [15]. We denote T_i as the final temperature of b_i. If the task set has a common deadline D, the safe temperature threshold is T_{max} and the energy budget is $\mathcal{E}$, TCEC scheduling problem can be represented as finding a voltage assignment $\mathcal{K}\{k_1, k_2, ..., k_n\}$[1] such that:

$$\sum_{i=1}^{n} (e_i^{k_i} + \psi_{v_{k_{i-1}}, v_{k_i}}) \leqslant \mathcal{E} \quad (3)$$

$$T_i \leqslant T_{max}, \forall i \in [1, n] \quad (4)$$

$$\sum_{i=1}^{n} (t_i^{k_i} + \omega_{v_{k_{i-1}}, v_{k_i}}) \leqslant D \quad (5)$$

where T_i is calculated based on Equation (2). Here Equation (3), (4) and (5) denote the energy, temperature and common deadline constraints, respectively.

For illustration, an extended timed automata $\mathcal{A}$ generated by TAG is shown in Figure 2 assuming that there are three 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 which is reached whenever there is deadline miss. There are also a source state and a destination state denoting the beginning and the end of the task execution. Therefore, there are totally $(n \cdot l + 4)$ states. 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 T and E, respectively. The execution time for every task under each voltage level is precalculated and stored in a global array $c[\,]$. The common deadline D is stored in variable *deadline*. Constants such as processor power values, thermal capacitance/resistance, ambient temperature and initial temperature 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. The **time** variable is never reset and elapses uniformly in every state. Both clock variables are initially set to 0.

The transition from the source state carries a function *initialization()* which contains updates to initialize all the variables and constants. Each state is associated with an invariant condition, in the form of **exec** $\leqslant c[\,]$, 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** $\leqslant$ *deadline* && **exec** $== c[\,]$. The former one ensures that the deadline is observed and the latter one actually triggers the transition, reflecting the fact 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, $T = calcTemperature(P[\,], T, c[\,])$, 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, $E = calcEnergy(P[], c[])$, adds the energy consumed by last task to E. 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** $>$ *deadline*, which triggers the transition whenever the deadline is missed during task execution. Note that not all the transition labels are shown in Figure 2.

[1] k_i denote the index of the processor voltage level assigned to b_i.

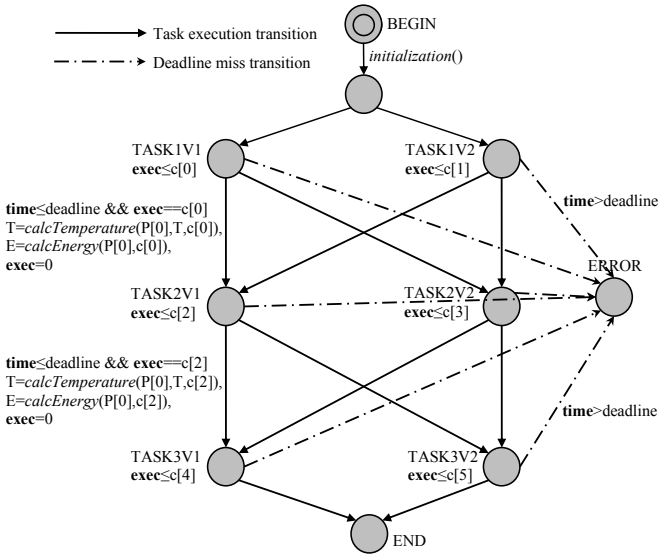

Figure 2: TCEC problem modeled in extended timed automata.

The extended timed automata's current configuration is decided by valuations of clock variables (**time** and **exec**) and global variables (T and E). Therefore, the system execution now is transformed into a sequence of states from the source state to the destination state[2]. The sequence consists of one and only one state from each disjoint set which represents a task. Solving the TCEC problem as formulated above is equal to finding such a sequence with the following properties. First, the final state is the destination state which guarantees the deadline constraint. Next, the temperature T is always below T_{max} in every state. Finally, the energy consumption E is no larger than $\mathcal{E}$. We can write this requirement as a property in computation tree logic (CTL) [11] as:

$$\mathbf{EG}((T < T_{max} \wedge E < \mathcal{E})\ \mathbf{U}\ \mathcal{A}.end) \tag{6}$$

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 scheduling that we want.

However, it is possible that the model checker's property description language does not support the operator of "until" (**U**), e.g. UPPAAL [4]. In that case, we can add two Boolean variables, $isTSafe$ and $isESafe$, to denote whether T and E are currently below the constraints. These two Boolean variables are updated in functions *calcTemperature*() and *calcEnergy*(), respectively, whenever a transition is performed. Once the corresponding constraint is violated, they are set to $false$. We can express our requirement in CTL as:

$$\mathbf{EF}(isTSafe \wedge isESafe \wedge \mathcal{A}.end) \tag{7}$$

Note that in the timed CTL that UPPAAL uses, the above property can be written as follows, where *Proc* represents the timed automata $\mathcal{A}$, which is called a "Process" in UPPAAL.

$$\mathbf{E} <> (Proc.End\ \mathbf{and}\ Proc.isTSafe\ \mathbf{and}\ Proc.isESafe) \tag{8}$$

Task set with individual deadlines: In the scenario where each task has its own deadline, e.g. periodic tasks, we have to make sure the execution blocks finish no later than their corresponding task's deadline. A global array, $d[\,]$, is used to store the deadline constraints of each execution block. If not applicable, i.e. the block does not end that task instance, its entry in $d[\,]$ is set to -1. Therefore, instead of Equation (5), we have:

$$\sum_{j=1}^{i}(t_j^{k_j} + \omega_{v_{k_{j-1}},v_{k_j}}) \leqslant d[i], \forall d[i] > 0 \tag{9}$$

Figure 3 shows part of the new timed automata. The difference lies in the guard of transitions. Instead of **time** $\leqslant$ *deadline*, the guard for transitions between task states is in the form of $((d[\,] > 0$ && **time** $\leqslant d[\,])\ ||\ d[\,] < 0)$. The transition from task state to error state now carries a guard of $(d[\,] > 0$ && **time** $> d[\,])$.

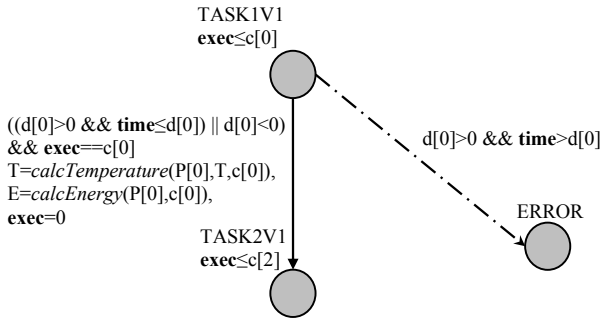

Figure 3: Problem modeling when every task has own deadline (partial graph).

4.3 Problem Variants

Our approach is also applicable to other problem variants by modifying the property and making suitable changes to invocation of the model checker.

TC: Temperature-constrained scheduling problem is a simplified version of TCEC. It only needs to ensure that the maximum instantaneous temperature is always below the threshold T_{max}. Therefore, the property can be written in CTL as:

$$\mathbf{EG}(T < T_{max}\ \mathbf{U}\ \mathcal{A}.end) \tag{10}$$

TA: To find a schedule so that the maximum temperature is minimized, we can employ a binary search over the temperature value range. Each iteration invokes the model checker to test the property (10) parameterized with current temperature constraint T_{max}. Initially, T_{max} is set to the mid-value of the range. If the property is unsatisfied, we search in the range of values larger than T_{max} in the next iteration. If the property is satisfied, we continue to search in the range of values lower than T_{max} to further explore better results. This process continues until the lower bound is larger than the upper bound. The minimum T_{max} and associated schedule, which makes the property satisfiable during the search, is the result. Note that the temperature value range for microprocessors is small in practice, e.g. $[30°C, 120°C]$. Hence, the number of iterations is typically no more than 7.

TAEC: TAEC has the same objective as TA except that there is an energy budget constraint. Therefore, we can solve the problem by using property (6) during the binary search.

TCEA: TCEA can be solved using the same method as TAEC except that the binary search is carried on energy values and temperature acts as a constant constraint. Since energy normally has a much larger value range, to improve the efficiency, we can discretize energy value to make trade-off between solution quality and design time. Since the number of iterations has a logarithmic relationship with the length of energy value range, only moderate discretization is enough.

4.4 Using SAT solver

Timed automata can be used to model the TCEC problem effectively. However, when the number of tasks is large, it can be

[2]The sequence of states follows the same characteristics of Equation (1).

time consuming to check the properties on the timed automata directly. The reason is that the underlying symbolic model checker like UPPAAL sometimes cannot handle large problems due to the state space explosion problem. Fortunately, we can alleviate this problem by transforming our model checking problem to a Pseudo-Boolean satisfiability problem [7]. Similar to SAT-based techniques [10] [24], which are used to reduce the test generation time, Pseudo-Boolean satisfiability solvers usually has better scalability over symbolic model checker and therefore can be used to perform model checking on timed automata more efficiently.

We use Boolean variable x_i^k to indicate wether block i is executed at voltage level v_k. The original model checking problem is equivalent to finding an assignment to $x_i^k, 1 \leqslant i \leqslant n, 1 \leqslant k \leqslant l$, which satisfies:

$$\sum_{k=1}^{l} x_i^k = 1, \forall i \in [1,n] \tag{11}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{l} x_i^k \cdot (e_i^k + \sum_{k'=1}^{l} x_{i-1}^{k'} \cdot \psi_{v_{k'},v_k}) \leqslant \mathcal{E} \tag{12}$$

$$T_i \leqslant T_{max}, \forall i \in [1,n] \tag{13}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{l} x_i^k \cdot (t_i^k + \sum_{k'=1}^{l} x_{i-1}^{k'} \cdot \omega_{v_{k'},v_k}) \leqslant D \tag{14}$$

Since there are production terms in the constraints, this problem is a nonlinear Pseudo-Boolean satisfiability problem. It can be solved by normal linear solvers like PBclasp [23] after standard normalization [5]. As we have shown in Section 5, compared to direct model checking of the original timed automata, it is more scalable to perform model checking using Pseudo-Boolean solver.

5. EXPERIMENTS

5.1 Experimental Setup

In this section, we describe the experimental setup for evaluation of our approach. A DVS-capable processor StrongARM [21] 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 [38]. We adopt the thermal resistance (R) and thermal capacitance (C) values from [15], which are $1.83^\circ C/Watt$ and $112.2mJoules/^\circ C$, respectively. The ambient temperature and initial temperature of the processor are set to $32^\circ C$ and $60^\circ C$, respectively. The scheduler and TAG shown in Figure 1 are both implemented in C++.

5.2 Result

5.2.1 Solving TCEC Problems

Table 1 shows the results on task sets with different number of blocks and constraints. The first and the second column are the index and number of blocks of 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 wether the there exists a schedule which satisfies all the constraints. The last three columns give the actual maximum temperature (AT), total energy cost (AE), and time required to finish all blocks (AD) using the schedule found by the model checker (UPPAAL). It can be observed that our approach can find the solution (if exists) which satisfied all the constraints.

5.2.2 Running Time Variations

Table 1: TCEC results on different task sets

TS	#Blk	TC	EC	DL	Found?	AT	AE	AD
1	10	85	180000	7000	Y	77	171612	6865
		85	150000	8000	Y	77	149623	7966
		80	140000	8000	N			
2	12	85	70000	2500	Y	79	66375	2499
		85	60000	2700	Y	76	59911	2667
		80	60000	2500	N			
3	14	90	90000	2600	Y	90	81287	2540
		85	80000	2800	Y	79	71649	2702
		90	80000	2700	N			

We have studied the impact of constraint variations on the running time required by UPPAAL. To achieve this, we measure the model checking time using task set 2 with two constraints kept constant while let the third one vary (TC, EC and DL). Figure 4 summarizes the results.

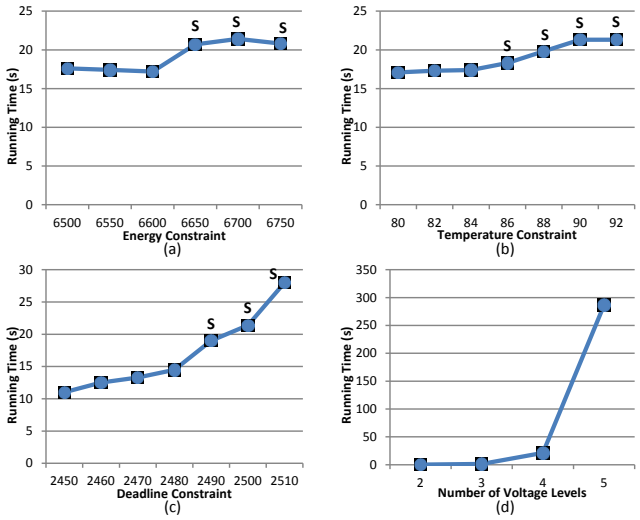

Figure 4: Running time with different constraints.

For energy constraint (Figure 4(a)) and temperature constraint (Figure 4(b)), we can observe that time requirement are not notably affected by the variation of these constraints. In general, it takes more time when the constraint can be satisfied (labeled "**S**" in Figure 4). When the constraint goes below or beyond the range shown in Figure 4, the running time remains the same or slightly decreases in both cases because the constraint will either be easily falsified or no longer limit the search space, respectively. However, for the deadline constraint (Figure 4(c)), our experimental results show that the running time requirement will increase with the deadline, because larger time budget yields a larger solution search space for the model checker. We have also investigated the relation between the number of voltage levels and the time required for model checking. As shown in Figure 4(d), model checker's running time grows rapidly when more voltage levels are employed. This is due to the exponential growth of the search space.

5.2.3 Comparison of UPPAAL and SAT Solver

We also compared the efficiency of conventional symbolic model checker (UPPAAL) with our Pseudo-Boolean satisfiability based model checking algorithm (PB approach) on task sets with different number of blocks. The first six columns of Table 2 are same as Table 1. The last two columns of Table 2 shows the results (running time in seconds). Since UPPAAL failed to produce result for task set 4 and 5, we only report the running time of the PB-based approach. It can be seen that PB-based approach outperforms UPPAAL by more than 10 times on average. Moreover, PB-based approach can solve much larger problems in reasonable running time.

Table 2: Running time comparison on different task sets

TS	#Blk	TC	EC	DL	Found?	UPPAAL	PB
1	10	85	180000	7000	Y	9.6	0.1
		85	150000	8000	Y	9.9	0.1
		80	140000	8000	N	9.4	0.1
2	12	85	70000	2500	Y	18.5	0.9
		85	60000	2700	Y	106.6	0.1
		80	60000	2500	N	17.5	0.8
3	14	90	90000	2600	Y	65.1	9.7
		85	80000	2800	Y	648.3	3.1
		90	80000	2700	N	208.6	14.3
4	50	85	380000	39500	Y	-	104.8
5	100	85	720000	83800	Y	-	428.2

6. CONCLUSION

In this paper, we proposed a model checking approach for temperature and energy-constrained scheduling problem in multitasking systems based on processor voltage scaling. We modeled the problem using extended timed automata which is solved by a model checker as well as a SAT solver. We proposed a flexible and automatic framework which makes our approach applicable to temperature or energy-constrained problem as well as other variants and independent of any system characteristic. Extensive experimental results demonstrate the effectiveness of our approach.

7. REFERENCES

[1] R. Alur et al., A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[2] H. Aydin et al., Determining optimal processor speeds for periodic real-time tasks with different power characteristics. *ECRTS*, 2001.

[3] H. Aydin et al., Power-aware scheduling for periodic real-time tasks. *IEEE Trans. Comput.*, 53(5):584–600, 2004.

[4] J. Bengtsson et al., Uppaal—a tool suite for automatic verification of real-time systems. *DIMACS/SYCON workshop on Hybrid systems III : verification and control*, 1996.

[5] T. Berthold et al., Nonlinear pseudo-boolean optimization: Relaxation or propagation? *SAT*, 2009.

[6] S. Borkar et al., Parameter variations and impact on circuits and microarchitecture. *DAC*, 2003.

[7] E. Boros et al., Pseudo-boolean optimization. *Discrete Appl. Math.*, 123(1-3):155–225, 2002.

[8] J.-J. Chen et al., On the minimization fo the instantaneous temperature for periodic real-time tasks. *RTAS*, 2007.

[9] J.-J. Chen et al., Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. *RTCSA*, 2007.

[10] M. Chen et al., Efficient decision ordering techniques for SAT-based test generation. *DATE*, 2010.

[11] E. Clarke et al., Model Checking. MIT Press, 1999.

[12] M. J. Ellsworth. Chip power density and module cooling technology projections for the current decade. *ITHERM*, 2004.

[13] E. Fersman et al., Timed automata with asynchronous processes: Schedulability and decidability. *TACAS*, 2002.

[14] S. Gunther et al., Managing the impact of increasing microprocessor power consumption. *ITJ*, 5(1):1–9, 2001.

[15] R. Jayaseelan et al., Temperature aware task sequencing and voltage scaling. *ICCAD*, 2008.

[16] R. Jejurikar et al., Energy aware non-preemptive scheduling for hard real-time systems. *ECRTS*, 2005.

[17] R. Jejurikar et al., Leakage aware dynamic voltage scaling for real-time embedded systems. *DAC*, 2004.

[18] Y. Liu et al., Thermal vs energy optimization for dvfs-enabled processors in embedded systems. *ISQED*, 2007.

[19] A. Lungu et al., Multicore power management: Ensuring robustness via early-stage formal verification. *MEMOCODE*, 2009.

[20] S. M. Martin et al., Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. *ICCAD*, 2002.

[21] Marvell. *Marvell StrongARM 1100 processor*. www.marvell.com.

[22] C. Norström et al., Timed automata as task models for event-driven systems. *RTCSA*, 1999.

[23] PBclasp. *PBclasp*. http://potassco.sourceforge.net/labs.html.

[24] X. Qin et al., Synchronized generation of directed tests using satisfiability solving. *VLSI Design*, 2010.

[25] D. Shin et al., Dynamic voltage scaling of periodic and aperiodic tasks in priority-driven systems. *ASP-DAC*, 2004.

[26] S. Shuklaet et al., A model checking approach to evaluating system level dynamic power management policies for embedded systems. *HLDVT*, 2001.

[27] K. Skadron et al., Temperature-aware computer systems: Opportunities and challenges. *IEEE Micro*, 23(6):52–61, 2003.

[28] K. Skadron et al., Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim.*, 1(1):94–125, 2004.

[29] R. Viswanath et al., Thermal performance challenges from silicon to systems. *ITJ*, 4(3):1–16, 2000.

[30] S. Wang et al., Reactive speed control in temperature-constrained real-time systems. *ECRTS*, 2006.

[31] W. Wang et al., Leakage-aware energy minimization using dynamic voltage scaling and cache reconfiguration in real-time systems. *VLSI Design*, 2010.

[32] W. Wang et al., PreDVS: preemptive dynamic voltage scaling for real-time systems using approximation scheme. *DAC*, 2010.

[33] W. Wang et al., SACR: scheduling-aware cache reconfiguration for real-time embedded systems. *VLSI Design*, 2009.

[34] N. Weste et al., CMOS VLSI Design: A Circuits and Systems Perspective. Addison Wesley, 2004.

[35] L.-T. Yeh et al., Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices. ASME Press, 2002.

[36] L. Yuan et al., Alt-dvs: Dynamic voltage scaling with awareness of leakage and temperature for real-time systems. *AHS*, 2007.

[37] S. Zhang et al., Approximation algorithms for power minimization of earliest deadline first and rate monotonic schedules. *ISLPED*, 2007.

[38] S. Zhang et al., Approximation algorithm for the temperature aware scheduling problem. *ICCAD*, 2007.

[39] X. Zhong et al., System-wide energy minimization for real-time tasks: Lower bound and approximation. *ICCAD*, 2006.