2013 26th International Conference on VLSI Design and the 12th International Conference on Embedded Systems

Scheduling Aperiodic Tasks in Next Generation Embedded Real-Time Systems

Rehan Ahmed Parameswaran Ramanathan Kewal K. Saluja
Department of Electrical and Computer Engineering
University of Wisconsin Madison, Madison,WI 53706, USA
rahmed3@wisc.edu, parmesh@ece.wisc.edu, saluja@ece.wisc.edu

Chunhua Yao
Marvell Semiconductor Inc.
5488 Marvell Lane, Santa Clara, CA 95054, USA
chunhua@marvell.com

***Abstract*—In this paper we investigate a method of scheduling aperiodic tasks with a given periodic schedule, in a thermally constrained embedded real-time system. Considering thermal constraints is important for current and future processing systems because of the rapid rise in the power densities of processors; especially in multicore environments. Therefore, if thermal constraints are not considered, processor temperatures may increase to levels at which the reliability of the underlying semiconductor devices is compromised. As part of the proposed approach, an accurate and non-extensive temperature estimation scheme is presented. Based on the temperature estimation scheme, a scheduling scheme for aperiodic tasks in the presence of periodic tasks is proposed. The primary advantage of the proposed scheme is that it can account for temporal variations in task power consumption. The results show that our approach can provide significant improvements in aperiodic task response times, over schemes that do not allow tasks to have temporal variations in power consumption.**

***Index Terms*—real-time systems; thermal-aware scheduling; aperiodic tasks.**

I. Introduction

Tasks in embedded applications such as process control systems, multimedia communication, biomedical systems and automotive systems usually have deadline constraints by which they must complete their computations. Extensive research has been done on the scheduling tasks in real-time applications with timing constraints. However, there are certain critical issues which have, until recently, not received much attention. Specifically, with the scaling of Integrated Circuit technology, thermal constraints of the processing system have become increasingly important and must be considered for embedded real-time systems running on current and next generation processors. This is because of rapid and continuing rise in power densities of processors. The Assembly and Packaging Tables in International Technology Roadmap for Semiconductors (ITRS) 2010 update [1] project nearly doubling of average power density from $0.55 - 0.7\ W/mm^2$ in 2011 to $1.2 - 1.35\ W/mm^2$ by 2024. Furthermore, the peak power densities can be two orders of magnitude higher than these average power density numbers [2]. Higher power densities lead to increase in operating temperatures of the devices which may adversely affect device reliability and performance [3].

In general purpose computing, Dynamic Thermal Management (DTM) is widely used to ensure that the system's thermal constraints are not violated [4]. However, the problem becomes more complex for real-time systems since timing constraints for all tasks also have to be considered. Ahn et al. [5] use reactive speed scaling and a server based scheme to schedule aperiodic tasks in a system with identical period periodic tasks. In several works [6]–[11], Dynamic Voltage Scaling (DVS) is used to schedule tasks such that their timing and thermal constraints are satisfied. In these works, parameters such as response time and schedulable utilization are optimized while considering timing and thermal constraints. Thidapat et al. [12] used integer linear programming to minimize the maximum temperature for a given set of real-time tasks running on a multicore system. Fisher et al. [13] use the thermal characteristics of processing cores to derive preferred frequencies for processing elements in a homogeneous multicore system to minimize system temperature.

A major limitation of all above approaches is their inability to account for temporal variations in the power consumption of tasks. In these works, the processor power consumption is a direct function of the operating frequency and does not depend on the variations in activity of tasks. This proves to be a significant limitation, since the power of each task must be conservatively assumed to be the maximum value in order to guarantee hard thermal constraints. In contrast, our scheme does not require a conservative constant power assumption. Instead, it utilizes an accurate, worst-case, time-varying power consumption profile for each task. The challenge with considering such a detailed power consumption model is that runtime computation overhead of accurately estimating the thermal impact of executing a given task instance using conventional methods is very large. In this paper, we propose a scheme which does not have this limitation. By exploiting superposition principle along with certain pre-computed thermal characteristics of each task, accurate thermal impact estimation is done sufficiently fast at runtime.

We assume that the embedded application is composed of periodic and aperiodic tasks; with the periodic schedule pre-computed using existing algorithms [12]. Aperiodic tasks are activated in response to external events. Therefore, scheduling of aperiodic task instances must be done at runtime; which has two main challenges. First, the scheduler must ensure that temperature increase due to the execution of an aperiodic task instance does not cause thermal violation during its execution and during executions of all future periodic task instances.

1063-9667/13 $26.00 © 2013 IEEE
DOI 10.1109/VLSID.2013.157

This problem is particularly challenging in multicore systems because executing a task on a particular core not only increases the temperature of that core, but also the temperature of other cores. Second, the time required to decide where and when the aperiodic task instance will execute must be small because it strongly determines the average rate at which new aperiodic task instances can arrive.The approach proposed in this paper addresses both these challenges.

Organization: The paper is organized as follows: Section II covers the preliminaries by way of defining system and task models. This is followed by temperature estimation scheme in section III. Scheduling scheme for aperiodic tasks in the embedded application along with accompanying theory is given in section IV. This section also provides an illustrative example in which we demonstrate how scheduling decisions for aperiodic tasks are made. Section V compares the scheduling results of our scheme with scheduling schemes that do not allow temporal variations in task power consumption.

II. Preliminaries

This section presents the model for the processing system and real-time tasks comprising the embedded application.

A. Processor Model

The computer system considered in this paper is uniform multicore with n cores. For clean representation of equations, we define Z_n as a set of all integers from 0 to $n-1$ where n is the number of cores in a given multicore system, i.e. $Z_n = \{0, 1 \ldots n-1\}$. The temperature of each core must never exceed threshold temperature Δ for proper functioning of the system. Each core also consumes power due to leakage current. Let p_{idle} be the leakage power at temperature Δ.

The cooling model used in this work assumes a simple first-order approximation of the cooling process [7]–[10]. This model also accounts for multicore environment. Specifically, suppose that we have to estimate temperature for an n core system and that $\mathbf{\Theta}(\mathrm{t}_0)$ is a vector which represents the temperature of all cores at time t_0. Also suppose that the ambient temperature is Θ_{a} and that all cores are idle after t_0. Then, the processor temperature at $t > t_0$ is modeled as:

$$\mathbf{\Theta}(t) = (\max_{z \in Z_n} \{(\Theta(t_0)_z - \Theta_{\mathrm{idle}_z} - \Theta_a) \cdot e^{-\beta(t-t_0)} + \Theta_{\mathrm{idle}_z}\} + \Theta_a) \cdot \mathbf{1}$$

Where β is a known cooling coefficient and $\mathbf{\Theta}_{\text{idle}}$ is a vector representing the steady state temperature of all cores when they are idle and therefore consuming only leakage power (p_{idle}), at initial and ambient temperature of $0°$C. $\Theta(t_0)_z$ and Θ_{idle_z} represent the z^{th}entry of $\mathbf{\Theta}(t_0)$ and $\mathbf{\Theta}_{\text{idle}}$ vector respectively. $\mathbf{1}$ is a $1 \times n$ vector comprising of all 1s. $\mathbf{\Theta}(t)$ is a vector representing the temperature of all cores at time t given that all cores are idle after t_0. In this model, the maximum of initial temperatures of all cores is used to conservatively bound the temperature impact of a given core on other cores. For improving the readability of equations in the rest of this paper, we define the following function:

$$\mathbf{cool}(\boldsymbol{\theta}, u) = (\max_{z \in Z_n} \{(\theta_z - \Theta_{\mathrm{idle}_z}) \cdot e^{-\beta u} + \Theta_{\mathrm{idle}_z}\}) \cdot \mathbf{1},$$

TABLE I: Notations used to characterize real-time tasks

C_i	Worst case computation time of a task i.
T_i	Period of periodic task i.
D_i	Hard deadline of periodic task i.
A_i	Arrival time of aperiodic task i.
$p_i(u)$	Worst case estimate of the power consumed by task i after u units of its computation, where $0 \leq u \leq C_i$. This power consumption is assumed to be the sum of both dynamic and leakage power.
$\boldsymbol{\eta}_{i,y}(u)$	A vector with entries representing temperature of each core after u units of execution of task i on core y assuming that: (i) the task instance starts to execute when the processor temperature is $0°$C, (ii) the ambient temperature is $0°$C, (iii) the power consumed by the task on core y is $p_i(u) - p_{\text{idle}}$ and (iv) the power consumed by the task on core $z \neq y$ is 0.
L	Least common multiple of the periods (T_i) of all periodic tasks. In the remaining document L shall also be referred as hyperperiod.
ψ	Timing feasible schedule of periodic tasks. The periodic task schedule is assumed to be cyclic, i.e. the task instance being executed at any time t is same as the one executing at time $t \bmod L$.

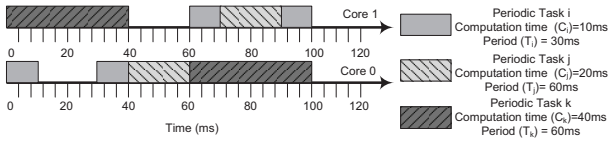


Fig. 1: Superposition Example: Periodic Schedule

where $\boldsymbol{\theta}$ is an n valued vector and θ_z represents its z^{th} element. Based on this function, the cooling from $\mathbf{\Theta}(t_0)$ can be represented as:

$$\mathbf{\Theta}(t) = \mathbf{cool}(\mathbf{\Theta}(t_0) - \Theta_a \cdot \mathbf{1}, t - t_0) + \Theta_a \cdot \mathbf{1} \qquad (1)$$

The technique proposed in this paper is complementary to conventional use of Dynamic Voltage and Frequency Scaling (DVFS) [6]–[11]. Hence, for clarity of presentation DVFS control capabilities available for thermal management are not included in the proposed technique.

B. Task Model

The embedded application is comprised of real-time periodic and aperiodic tasks. All task instances are non-preemptive. The notations used to characterize the application tasks are given in table I along with explanations.

III. Temperature Estimation Scheme

The framework used for estimating the temperature of all cores at a given time instant has two primary components. They are: 1) Superposition Principle and 2) Sensor Readings. These components are discussed below.

A. Using Superposition Principle

Several works in literature exploit the duality between heat flow and electrical phenomenon to model the thermal characteristics of an IC by an RC grid [7], [12]–[14]. It is well known that RC grid is a linear time invariant system. In this work, we exploit this property of the model by using superposition principle to estimate processor temperature after execution of multiple tasks. Let us explain this by an example

in which we construct the hyperperiod thermal profile ($\boldsymbol{\eta}_\psi$) using a given periodic schedule (ψ) and the individual thermal profiles of periodic tasks. The application consists of three periodic tasks, i, j and k. A timing feasible schedule for these two tasks executing on a dual-core system is given in figure 1. Since the system has two cores, there are two separate thermal profiles for each periodic task, i.e. for periodic task i, $\boldsymbol{\eta}_{i,0}$ is the thermal profile of periodic task i executing on core 0, while $\boldsymbol{\eta}_{i,1}$ is the thermal profile of task i executing on core 1. Similarly, $\boldsymbol{\eta}_{j,0}$ and $\boldsymbol{\eta}_{j,1}$ are the thermal profiles of task j and $\boldsymbol{\eta}_{k,0}$ and $\boldsymbol{\eta}_{k,1}$ are the thermal profiles of task k executing on core 0 and core 1 respectively. To generate hyperperiod thermal profile, the thermal profiles of individual periodic tasks are time shifted, as per the periodic schedule (ψ) given in figure 1, and added. Based on the periodic schedule shown in figure 1, following equation can be used to evaluate hyperperiod thermal profile:

$$\begin{aligned}\boldsymbol{\eta}_\psi(u) =&\boldsymbol{\eta}_{i,0}(u) + \boldsymbol{\eta}_{i,0}(u-30) + \boldsymbol{\eta}_{i,1}(u-60) + \boldsymbol{\eta}_{i,1}(u-90) + \\ &\boldsymbol{\eta}_{j,0}(u-40) + \boldsymbol{\eta}_{j,1}(u-70) + \boldsymbol{\eta}_{k,1}(u) + \boldsymbol{\eta}_{k,0}(u-60)\end{aligned} \tag{2}$$

where $u \in [0, 120)$ and $\boldsymbol{\eta}_{\_,\_}(x) = 0$ for $x < 0$

Now suppose that we need to evaluate the temperature of the processor during a given hyperperiod and that the initial processor temperature is $\boldsymbol{\Theta}(0)$. Assuming that only periodic tasks are executed in the interval $t \in [0, L)$, we can use the following equation to evaluate temperature:

$$\boldsymbol{\Theta}(t) = \mathbf{cool}(\boldsymbol{\Theta}(0) - \Theta_a \cdot \mathbf{1}, t) + \Theta_a \cdot \mathbf{1} + \boldsymbol{\eta_\psi}(t) \tag{3}$$

where $t \in [0, L)$.

B. Sensor Reading

The temperature estimation scheme presented in this work has several sources of error. These sources include process variations, runtime variations in the behavior of real-time tasks, conservative leakage power assumption, inaccuracies in the cooling model etc. To ensure that the temperature estimation errors do not accumulate, on-chip sensors are used to periodically read actual core temperature and correct any error in temperature estimation. For the evaluations conducted in this work, the sensors are read periodically at the start of every hyperperiod. HotSpot simulator readings act as proxy for temperature readings from on-chip sensors. It is assumed that sensor readings are available for each processing core.

IV. Aperiodic task scheduling: Theory and Algorithm

We now present the algorithm used to make scheduling decisions for aperiodic tasks. To make scheduling decisions under thermal constraints, we need a mechanism to rapidly estimate the thermal impact of an aperiodic task. Therefore, we first specify the equation used to estimate processor temperature after execution of aperiodic tasks. The algorithm used for scheduling aperiodic tasks is given in section IV-A.

Suppose that ψ defines a given periodic schedule with thermal profile $\boldsymbol{\eta_\psi}$. Consider an aperiodic task τ arriving at time A_τ. Suppose that the scheduler is making scheduling decision for τ at time $t_{\text{sch}} \geq A_\tau$ and is trying to determine whether τ can be scheduled within a contiguous idle time starting at time s on core y.

Let kL be the start time of the hyperperiod containing t_{sch}, i.e., $k = \lfloor \frac{t_{\text{sch}}}{L} \rfloor$. Recall that, on-chip thermal sensor is read at start of each hyperperiod. Let the most recent temperature measurement from on-chip thermal sensor be $\boldsymbol{\Theta}_{\mathbf{meas}}(kL)$. Then, by using superposition and cooling model, the processor temperature due to execution of τ and all periodic tasks is:

$$\boldsymbol{\Theta}(t) = \begin{cases} \mathbf{cool}(\boldsymbol{\Theta}_{\mathbf{meas}}(kL), t-kL) + \boldsymbol{\eta}_{\tau,y}(t-s) + \boldsymbol{\eta}_\Psi(t \bmod L) \\ +\boldsymbol{\Theta}_{\text{Idle}} + \Theta_{\text{a}} \cdot \mathbf{I} + \sum_{i=k+1}^{\lfloor t/L \rfloor} \max_{z \in Z_n} \{\boldsymbol{\eta}_\psi(L)\} e^{-\beta(t-iL)} \cdot \mathbf{1} \\ \qquad\qquad\qquad\qquad\qquad\qquad \textbf{If } t \in [s, s + C_\tau) \\ \mathbf{cool}(\boldsymbol{\Theta}_{\mathbf{meas}}(kL), t-kL) + \boldsymbol{\eta}_\Psi(t \bmod L) + \Theta_{\text{a}} \cdot \mathbf{1} \\ +\boldsymbol{\Theta}_{\text{Idle}} + \max_{z \in Z_n} \{\boldsymbol{\eta}_\tau(C_\tau)_z\} \cdot e^{-\beta(t-s-C_\tau)} \cdot \mathbf{1} \\ +\sum_{i=k+1}^{\lfloor t/L \rfloor} \max_{z \in Z_n} \{\boldsymbol{\eta}_\psi(L)\} e^{-\beta(t-iL)} \cdot \mathbf{1} \qquad \textbf{Otherwise} \end{cases} \tag{4}$$

However, we also have to account for the thermal effect of previously accepted aperiodic tasks. Let Π denote the set of all previously admitted aperiodic task instances scheduled to execute after time kL. Given a time $t \geq t_{\text{sch}}$, define:

- $U(t)$ Subset of Π containing all instances scheduled to complete before time t
- $V(t)$ Subset of Π containing either zero or one instance scheduled to be executing at t

For each scheduled aperiodic task, the core on which it is scheduled to execute is represented by u. Then, through superposition and cooling model, the temperature estimate at time t can be written as:

$$\begin{aligned}\boldsymbol{\Theta}(t) = &\sum_{j \in U(t)} \max_{z \in Z_n} \{\boldsymbol{\eta}_{j,u}(C_j)_z\} \cdot e^{-\beta(t-s_j-c_j)} \cdot \mathbf{1} \\ &+ \boldsymbol{\Theta}_{\textbf{Equation4}}(t) + \sum_{j \in V(t)} \boldsymbol{\eta}_{j,u}(t-s_j)\end{aligned} \tag{5}$$

Where s_j is the start time of task $j \in \Pi$ and C_j is the computation time of task $j \in \Pi$.

A. Aperiodic task scheduling algorithm

We now present the algorithm for scheduling aperiodic tasks. The algorithm is divided into online and offline phases:

1) Offline: In the offline phase, we determine a common maximum initial temperature for each core at the start of the hyperperiod. For a given periodic schedule ψ with hyperperiod length L and corresponding thermal profile $\boldsymbol{\eta_\psi}(u)$, Θ^*_ψ is a scalar defined as the common maximum temperature for each core at the start of the hyperperiod such that the thermal constraints of the system are not violated due to all future executions of periodic tasks. Θ^*_ψ is used in making scheduling decisions for aperiodic tasks.

Suppose that the initial temperature at time 0 is x for every core and only periodic tasks are executed after time 0. In this scenario, using superposition principle, the temperature of the cores at any time t is:

$$\begin{aligned}\boldsymbol{\Theta}(t) =&\mathbf{cool}((x - \Theta_a) \cdot \mathbf{1}, t) + \Theta_a \cdot \mathbf{1} + \boldsymbol{\eta_\psi}(t - kL) \\ &+ \sum_{j=1}^{k} \max_{z \in Z_n} \{\boldsymbol{\eta}_\psi(L)_z\} e^{-\beta(t-jL)} \cdot \mathbf{1}\end{aligned} \tag{6}$$

where $k = \lfloor \frac{t}{L} \rfloor$.

In this offline phase, equation 6 is used to search for the maximum value of x such that there are no thermal violations between time $[0, \infty)$ due to execution of periodic tasks. The largest value of x is Θ^*_ψ. Note that the search for Θ^*_ψ requires evaluating temperature using equation 6 for all times until ∞ and checking whether the temperature is below Δ at all times, for all cores. i.e.

$$\Theta^*_\psi = \max \left\{ x : \max_{\substack{0 \le t < \infty \\ z \in Z_n}} \Theta(t)_z \text{ from equation 6} \le \Delta \right\}$$

It is impractical to check for thermal constraint violation until ∞. However, given that the initial temperature for all cores is x, we only need to check for thermal violations until the start of some later hyperperiod at which the temperature for all cores becomes less than x. No thermal violations would occur beyond that point due to execution of periodic tasks.

2) Online: The online phase of the algorithm covers the scheduling scheme for aperiodic tasks. The process is explained further by an illustrative example which goes though all steps of the scheduling algorithm. We notice that when an aperiodic task is scheduled and then executed, the processor temperature after its execution is never lower than the scenario in which the aperiodic task was not executed. Therefore, while scheduling aperiodic tasks, we have to ensure that 1) thermal constraints for the system are not violated during aperiodic task execution, 2) thermal constraints for the system are not violated due to execution previously scheduled aperiodic tasks and 3) thermal constraints for the system are not violated due to the execution of all future periodic task instances. The value of Θ^*_ψ, as determined by offline algorithm in section IV-A1, is used to guarantee the third condition.

Aperiodic Scheduler: Since the scheduling of aperiodic tasks is done at run-time, it is important to reserve computation time for making the scheduling decisions and also take into account the thermal impact of those computations. For this purpose, a special *Aperiodic Scheduler* (AS) task is created and reserved as part of the periodic schedule. Like other periodic tasks, AS task is assigned worst case computation Time C_{AS}, period T_{AS}, worst case power profile $p_{\text{AS}}(u)$ and corresponding thermal profile $\eta_{\text{AS}}(u)$. Each AS task can perform a finite number of computations, in its assigned computation time (C_{AS}), for making scheduling decisions. We call the number of computations AS task can perform its *Computation Capacity*. The functioning of AS task is explained in figure 2.

Suppose that an AS task starts at time s_{AS}. The AS task takes as an input a list of all aperiodic tasks which arrived prior to s_{AS} and for which scheduling decision has not been made (*active_tasks*). From this list, the AS task picks the aperiodic task (τ) with earliest start time and generates a list of finite number of possible start times and corresponding cores (*possible_starts*) for τ after s_{AS}. Once the list is generated, the AS task searches for a thermally feasible start time/core until a valid value is found or the computation capacity of the AS task is exhausted.

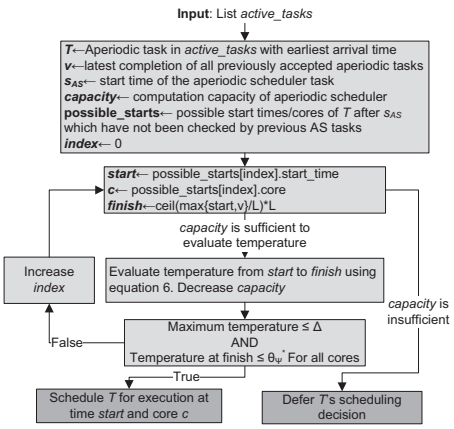


Fig. 2: Functioning of aperiodic scheduler

Algorithm in figure 2 uses equation 5 to estimate the processor temperature profile after execution of τ at a given start time/core. For a given start time/core combination to be thermally feasible, the maximum temperature as a result of $\tau's$ execution must be less than Δ for each core. Furthermore, the temperature at the end of the hyperperiod in which the aperiodic task with latest completion time executes must be less than Θ^*_ψ for each core. The second condition ensures that future executions of periodic tasks do not cause thermal violations. If an AS task is unable to find a feasible start within the limited number of computations, the scheduling decision is deferred so that a later AS task instance can search for a valid start time/core. We now illustrate the aperiodic task scheduling process by the following example:

Example: System Model: Number of cores = 2, $\Theta^*_\psi = 95°\text{C}$, $\Delta = 100°\text{C}$

Suppose an aperiodic task τ with worst case execution time of 15ms arrives at time 6ms. Also suppose that the next three AS task instances after the arrival time of τ start at 30ms, 60ms and 90ms. Each AS task instance has a computation capacity of 60 units; where a single computation involves evaluation of temperature for one core for a single time step. In this example, a single time step is equivalent to 3ms. Also suppose that hyperperiod length (L) is 90ms. It is assumed that the latest completion time for already scheduled aperiodic tasks is before time 90ms (i.e the value of *finish* in figure 2 is $\lceil (\max(start, 90ms))/L \rceil \times L$ ms where *start* is the time at which the AS task is trying to schedule τ). This schedule is illustrated in figure 3. Based on the periodic schedule and scheduled aperiodic tasks, three possible start times of τ and corresponding cores are (48ms,0), (51ms,0) and (66ms,1).

Figure 4 shows how the processor temperature is estimated at each scheduling attempt. The first scheduling attempt is made by AS task 1 at time 48ms on core 0. In this attempt, the maximum temperature ($100.2°\text{C}$) exceeds Δ. Therefore, τ cannot be scheduled at this point. After this attempt, AS task 1 has 32 units of computations left. This remaining computation capacity is sufficient to make a second scheduling attempt at time 51ms on core 0. However, in this scheduling attempt, the

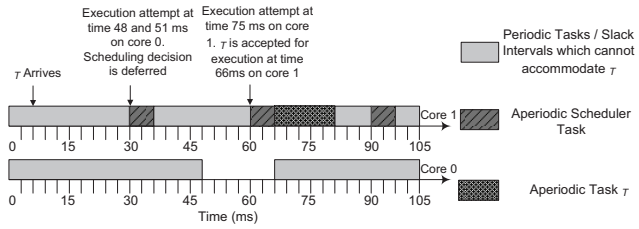


Fig. 3: Aperiodic task scheduling example: Timeline

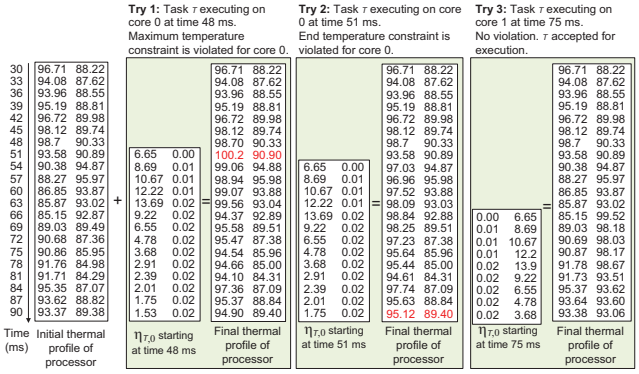


Fig. 4: Aperiodic task scheduling example: Thermal profile evaluation

temperature of core 0 at time *finish*$= 90ms$ (95.12°C) exceeds $\Theta^*_\psi = 95$°C. Therefore, τ cannot be scheduled for execution at this point. After the second attempt, AS task 1 has 6 units of computation left. A third possible start time for τ is 66ms on core 1. However, this scheduling attempt requires 16 units of computation. Since AS task 1 has only 6 units of computation capacity left, this scheduling attempt cannot be made by AS task 1. Therefore, the scheduling decision is *Deferred* by AS task 1. The third scheduling attempt is made by AS task 2 and τ is accepted for execution at time 66ms on core 1 since both conditions for scheduling an aperiodic tasks are satisfied (i.e. Maximum temperature $\leq \Delta$ AND $\Theta(finish) \leq \Theta^*_\psi$ for all cores). The response time of τ in this example is the time interval between its arrival (A_τ) and completion time ($s_\tau + C_\tau$). i.e Response Time $= 66\text{ms} + 15\text{ms} - 6\text{ms} = 75\text{ms}$.

V. Results and Conclusion

In this section, we demonstrate, through a series of experiments, that our scheduling and estimation scheme can give significantly better response times for aperiodic tasks compared to an approach which does not allow temporal variations in task power consumption. The parameters of processor, periodic tasks and aperiodic tasks for the results presented in this section are as follows.

System Model: Number of cores = 9, Δ = 100°C, β = 13.33, Θ_a = 45°C, $\mathbf{\Theta}_{\text{idle}}$ = [13.23 13.63 13.23 13.63 14.11 13.63 13.23 13.63 13.23]^T

Note that the temperature of the center core is somewhat higher during the idle operation of the system as it is surrounded by other cores, and therefore, cools at a slower rate.

Periodic Tasks: Experiments are conducted with three periodic task sets. Important parameters for all schedules are given in the following table:

Periodic Schedule	Periodic Utilization	Avg Power (W)	Θ^*_ψ ($^\circ C$)	Avg Temperature ($^\circ C$)
1	69.8%	152.17	89.95	82.9
2	64.2%	136.1	85.30	71.5
3	73.3%	161.4	89.95	85.0

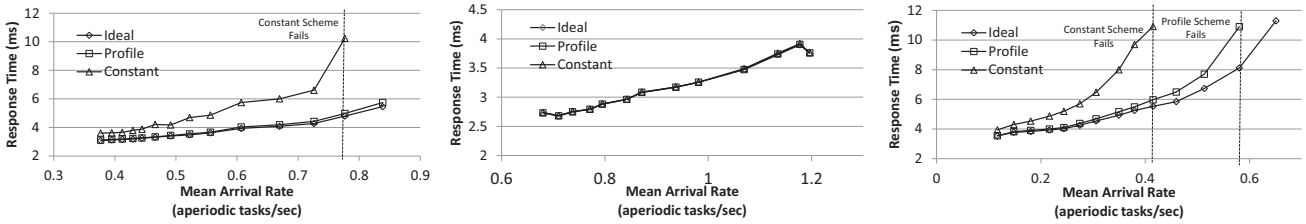
These parameters are provided because they are important in explaining the results presented in this section. The hyperperiod length for each periodic schedule is 36ms. The Aperiodic Scheduler task in each periodic schedule has a computation time of 300μs and a period of 600μs. Each AS task is assigned a computation capacity of 300000 units; where one unit of computation involves computing temperature of one core for one time unit (3μs). The periodic task power profiles are synthetically generated using a three state Markov model which emulates phases of low, medium and high task activity.

Aperiodic Tasks: We consider multiple instances of 30 different aperiodic tasks. These aperiodic tasks differ significantly in their power consumption profile and computation times. As was the case with periodic tasks, the power consumption profiles of aperiodic tasks are synthetically generated using a three-state Markov model. This again is assumed to be the worst case power consumption.

The experiments conducted in this section compare the aperiodic task response times of the proposed scheduling and estimation approach (*Profile Based* Scheme) with an ideal scheduling approach having no temperature estimation error (*Ideal* Scheme) and approaches that do not allow power consumption of tasks to have temporal variations (*Constant Power* Scheme) and therefore assume power consumption to be its maximum value. In all three schemes, the temperature estimation errors are corrected by reading sensors at the start of every hyperperiod. The arrival times of aperiodic task instances follow a Poisson process. Experiments are conducted for several aperiodic task mean inter-arrival times varying between 0.75ms and 9ms. The corresponding mean arrival rates are 0.1 to 1.3 aperiodic instances per millisecond (where mean arrival rates are the inverse of mean aperiodic task inter-arrival times). For each inter-arrival time/arrival rate, 1000 instances of aperiodic tasks are randomly generated.

Figure 5a illustrates how aperiodic task response time varies as the aperiodic task arrival rate is increased on a system running periodic tasks as per periodic schedule 1. This periodic schedule corresponds to a moderately loaded system. The results in figure 5a show that the aperiodic task response times for *Profile Based* scheme are significantly lower than their corresponding values for *Constant Power* scheme. Furthermore, *Profile Based* scheme performs close to *Ideal* scheme. Also note that the *Constant Power* scheme fails if the aperiodic task arrival rate is increased beyond 0.77 aperiodic tasks per millisecond. By failure, it is implied that the system has more load than it can sustain. The response times of aperiodic tasks keep increasing unboundedly as more and more aperiodic tasks enter the system.

However, if the system load is decreased, the difference between the three schemes becomes less pronounced. This can



(a) Average response time of aperiodic tasks on a system running periodic schedule 1 with aperiodic task mean inter-arrival times varying between 1.1ms and 3ms

(b) Average response time of aperiodic tasks on a system running periodic schedule 2 with aperiodic task mean inter-arrival times varying between 0.8ms and 1.5ms

(c) Average response time of aperiodic tasks on a system running periodic schedule 3 with aperiodic task inter-arrival times varying between 1.5ms and 9ms

Fig. 5: Aperiodic task response times for periodic schedule 1, 2 and 3

be seen in the results presented in figure 5b. These results show how the response times of aperiodic tasks increase as their arrival rate is increased on a system running periodic tasks as per periodic schedule 2. Note that because of low periodic task utilization (64.2%) and average system temperature due to execution of periodic tasks (71.4°C), the results shown in figure 5a correspond to a lightly loaded system.

On a system with high load, the difference between *Profile Based* and *Constant Power* scheme becomes even more pronounced compared to a system with medium load (periodic schedule 1). This is shown in figure 5c. The results in this figure 5c show how the response times of aperiodic tasks increase as the aperiodic task mean arrival rate is increased on a system running periodic tasks as per periodic schedule 3. This scenario corresponds to a highly loaded system due to high periodic task utilization (73.3%) and average system temperature (85.0°C). In this experiment, *Constant Power* scheme fails when aperiodic task arrival rate is increased beyond 0.41 aperiodic tasks per millisecond. *Profile Based* scheme can support higher aperiodic task arrival rates and fails when the aperiodic task arrival rate is increased beyond 0.57 aperiodic tasks per millisecond. For low arrival rates, *Profile Based* scheme performs close to *Ideal* scheme. However, at higher arrival rates, the difference between *Profile Based* and *Ideal* scheme becomes more pronounced.

Figure 6 shows the temperature of a given core between time 300ms and 400ms, for a given experiment set. The figure compares the temperature estimated by the proposed scheme with the actual processor temperature given by HotSpot. Hyperperiod boundaries are also marked. As shown in the figure, the proposed temperature estimation scheme is always conservative and the temperature estimation errors do not accumulate.

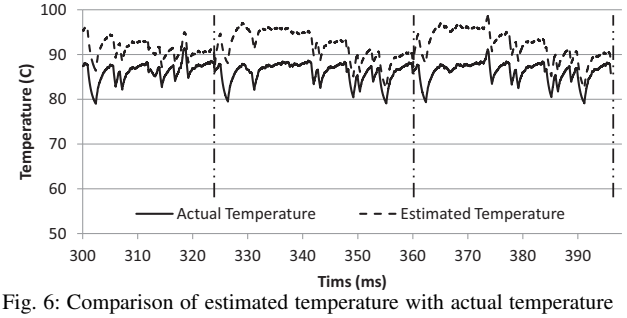


Fig. 6: Comparison of estimated temperature with actual temperature

Conclusion: This paper covers a scheme for making admission control decisions for aperiodic tasks in the presence of periodic tasks executing on a thermally constrained embedded multicore system. The proposed scheme allows temporal variations in task power consumption allowing a much less conservative task power characterization. The results show that significant scheduling gains can be realized, especially in systems with high load.

REFERENCES

[1] "Internationl technology roadmap for semiconductors," 2010. [Online]. Available: http://www.itrs.net/links/2010itrs/home2010.htm

[2] G. Link and N. Vijaykrishnan, "Thermal trends in emerging technologies," in *International Symposium on Quality Electronic Design*, march 2006, pp. 8 pp. –632.

[3] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur, "Thermal performance challenges from silicon to systems," *Intel Technology Journal*, vol. 3, pp. 1–16, 2000.

[4] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *High-Performance Computer Architecture. HPCA.*, 2001, pp. 171 –182.

[5] Y. Ahn and R. Bettati, "Transient overclocking for aperiodic task execution in hard real-time systems," in *Proceedings of the Euromicro Conference on Real-time Systems*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 102–111.

[6] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. ACM*, vol. 54, no. 1, pp. 1–39, 2007.

[7] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," in *Proceedings of the Euromicro Conference on Real-time Systems*, 2006, pp. 160–170.

[8] J.-J. Chen, S. Wang, and L. Thiele, "Proactive speed scheduling for real-time tasks under thermal constraints," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, April 2009, pp. 141–150.

[9] S. Wang and R. Bettati, "Delay analysis in temperature-constrained hard real-time systems with general task arrivals," in *Proceedings of the Real-time Systems Symposium*, 2006, pp. 323–334.

[10] J. Chen, C. Hung, and T. Kuo, "On the minimization fo the instantaneous temperature for periodic real-time tasks," in *13th IEEE Real Time and Embedded Technology and Applications Symposium, 2007. RTAS'07*, 2007, pp. 236–248.

[11] J.-J. Chen and T.-W. Kuo, "Voltage scaling scheduling for periodic real-time tasks in reward maximization," in *Proceedings of the Real-time Systems Symposium*, 2005, pp. 345–355.

[12] T. Chantem, R. Dick, and X. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 288–293.

[13] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, April 2009, pp. 131–140.

[14] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 1, no. 1, pp. 94–125, 2004.