Temperature Minimization Using Power Redistribution in Embedded 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

Abstract—**Due to rapidly increasing power densities in Integrated Circuits (ICs) and the resulting increases in on-chip temperatures, thermal management has become an important issue in embedded system design. This paper focuses on thermal aware scheduling of periodic real-time tasks in embedded systems. The paper exploits a concept called Total Thermal Impact of a periodic task set to develop some theoretical results on thermal aware scheduling. Based on these results, the paper presents a scheduling heuristic called Power Redistribution Algorithm (PRA). The heuristic interleaves tasks with different power consumption and processor idle times in such a way that all real-time task deadlines are met and system temperature is minimized. The heuristic has small runtime complexity but results in substantial temperature reduction.**

I. Introduction

Integrated Circuit (IC) technology scaling has resulted in increasing power densities. The Assembly and Packaging tables in the 2010 International Technology Roadmap for Semiconductors (ITRS) projects nearly doubling of power densities from 0.65-0.8 W/mm^2 in 2013 to 1.2-135 W/mm^2 in 2024 [1]. Furthermore, localized power densities can be more than two orders of magnitude higher than these average power densities [2]. These large localized high power densities often result in thermal hot spots which can adversely affect the system reliability and performance. A $10° - 15°C$ change in processor temperature can cause $\sim$2X difference in its lifespan [3]. Leakage power also increases with temperature, which in turn further increases the temperature. This cyclic dependance between leakage power and temperature causes *thermal runaway*, which, if left unhandled, can cause significant damage to the IC.

Excessive heating of the IC is handled by various Dynamic Thermal Management Techniques [4]. These include Dynamic Voltage and Frequency Scaling (DVFS), Decode Throttling, Speculation Control, etcetera. Since DTM techniques control temperature at the cost of system performance/speed and since many embedded applications have hard deadline constraints, performance degradation may cause deadline misses with potentially catastrophic consequences. Most of recent works on DTM techniques for embedded real-time systems use DVFS schemes to keep the system temperature below specified threshold [5]–[9]. These DVFS schemes can be subdivided into reactive [5] and proactive schemes [7] [9]. Many of these schemes are for periodic tasks although more general task models have also been studied. Schor et al. [10] perform worst case temperature analysis for real-time tasks running on multicore systems. They use a more generalized task model by using the concepts of real-time calculus [11] to model arrivals and computation demands of real-time tasks.

In contrast, the solution proposed in this paper focuses on thermal-aware scheduling of periodic real-time tasks that is complementary to DVFS based schemes. That is, it does not rely on DVFS, but can utilize any solution that has a priori assigned different processor speeds (through DVFS) to tasks in the application. Since our solution does not require DVFS, it can also be used in embedded systems that do not support DVFS and/or in systems where the time overhead for changing the voltage and/or frequency is unacceptably large. The scheme proposed in this paper has two parts. First, the proposed scheme computes the time by which the execution of all ready periodic task instances, including partial instances, can be delayed without causing any deadline misses. This *Slack* value is then used to schedule/interleave different periodic tasks and idle times in order to reduce the variations in system temperature. The target temperature around which the variations are reduced is computed using a concept called Total Thermal Impact of the periodic task set. The theoretical results in the paper show that the chosen target temperature is the smallest possible for a given periodic task set and the processor system.

Another distinguishing aspect of the proposed scheme is that it models the power consumption of a task as function of both its activity and the processor speed. In contrast, most existing work assumes that the power consumption of a task is only a function of the processor speed at which the task is executed. The differences between the task activities and their resulting impact on power consumption is not modeled in the existing schemes. The existing schemes are thus compelled to use the worst-case activity factor among all the tasks. Since studies have shown that power consumption of different tasks can vary by a factor > 4 [12], the proposed scheme results in a superior thermal management solution.

Hence, the primary contribution of this work is the following: *We present a scheduling algorithm which redistributes power by interleaving periodic task instances with different power consumptions; in order to reduce system temperature.*

This paper is organized as follows: Section II covers the system and task model used in this work. This is followed by the proposed solution in Section III. The proposed solution uses Slack evaluation scheme presented in Section III-A and concept of Thermal Impact [13] in Section III-B. These concepts are leveraged to come up with a scheduling heuristic which minimizes processor temperature in Section III-C. Results are presented in Section IV followed by conclusion.

II. System Model and Problem Definition

A. Processor Model

This work considers uni-processor. Processor power consumption at time t, $P(t)$ is given by the following equation:

$$P(t, \Theta(t)) = s(t)^3 a(t) + \delta\Theta(t) + \rho, \tag{1}$$

where $s(t)$ is the speed of the processor, $a(t)$ is the switching activity, $\Theta(t)$ is the processor temperature and ρ and δ are constants. The first term models the dynamic power and the sum of second and third terms models the leakage power. Although leakage current increases rapidly with temperature, it is shown in [14] that within typical operating range of processor, leakage power can be modeled as a linear function of temperature as in Equation 1.

B. Thermal Model

To evaluate processor temperature, we adopt Fourier's Law [15]. The thermal behavior is, therefore, modeled as an RC circuit where heat flow is modeled as current (Power $P(t)$) passing through a thermal resistance. The delay in heat increase is modeled as a thermal capacitance. Therefore, thermal behavior is modeled by the following equation:

$$\Theta(t)' = \frac{s(t)^3 a(t)}{C} - (\frac{1}{RC} - \frac{\delta}{C})\Theta(t) + (\frac{\rho}{C} + \frac{\Theta_a}{RC}) \tag{2}$$

For notational brevity, we perform variable transformation as in [7]. Let $\beta = \frac{1}{RC} - \frac{\delta}{C}$ and $\theta(t) = C\Theta(t) - \frac{C(R\rho + \Theta_a)}{1 - R\delta}$. With this transformation, Equation 2 can be transformed as:

$$\theta(t)' = a(t)s(t)^3 - \beta\theta(t) \tag{3}$$

In the rest of this paper, term 'temperature' will refer to $\theta(t)$ and '*Actual Temperature*' will refer to $\Theta(t)$. Following thermal parameters are used for all results/examples presented in this paper: $R = 0.36$,$C = 0.8$, $\rho = 0.1$, $\delta = 0.001$, $\Theta_a = 40°C$

C. Task Model

We consider the scheduling of periodic tasks. Furthermore, it is assumed that all tasks are infinitely preemptible. A periodic task (τ) is characterized by worst-case computation demand (C_τ), period (T_τ) and worst case activity (A_τ). C_τ and A_τ may be obtained by profiling the real-time/embedded tasks. All periodic tasks are synchronous (Deadline of a given periodic task instance coincides with the arrival of its next instance). The power consumption of task τ at speed s is equal to $A_\tau s^3$. The computation time of τ at speed s is equal to C_τ/s. *Note that, in this power model, power consumption can, vary across different tasks due to difference in activity, even if speed is kept constant*. The thermal profile of periodic task τ at speed s ($\eta_{\tau,s}(u)$) is defined as the processor temperature after u units of execution of task τ at speed s, assuming that:

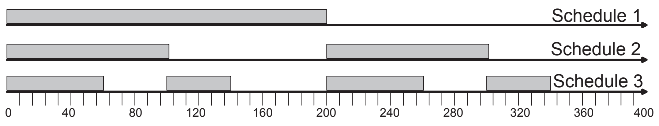

Fig. 1: Three possible schedules for periodic task τ

a) Task instance starts to execute when processor temperature is 0, b) Dynamic power consumed by the task is $A_\tau s^3$ until time C_τ/s, and c) Dynamic power consumed by the task is 0 after time C_τ/s. Thermal profile of a given task is evaluated at 0 initial temperature so that it can be shifted/superposed to any initial temperature. All evaluation conducted in the results section, is done at a temperature higher than 0. Solving Equation 3, thermal profile of τ at constant speed s is:

$$\eta_{\tau,s} = \begin{cases} \frac{A_\tau s^3}{\beta}(1 - \mathrm{e}^{-\beta t}) & 0 \leq t < C_\tau/s \\ \frac{A_\tau s^3}{\beta}(1 - \mathrm{e}^{-\beta c_\tau/s})\mathrm{e}^{-\beta(t - C_\tau/s)} & t \geq C_\tau/s \end{cases} \tag{4}$$

Periodic task set is represented by Γ. $\Pi(t)$ represents cyclic schedule of the tasks in periodic task set Γ. The value of $\Pi(t)$ is a tuple which identifies one of the n periodic tasks executing at time t at speed s, or $\emptyset$ if the system is idle at time t. The schedule is assumed to be cyclic i.e. $\Pi(t) = \Pi(t + kL)$ where $k = 0, 1, 2, ..$ and L is the hyperperiod of the periodic task set Γ. L is equal to the the least common multiple of the periods of all periodic tasks in Γ.

D. Problem Definition

Given a periodic task set Γ, the thermal aware scheduling problem is to minimize the maximum system temperature after repeated executions of periodic tasks without violating their deadline constraints. We now present a new scheduling algorithm for solving this problem.

III. Power Redistribution Algorithm (PRA)

PRA works by interleaving periodic tasks with different power consumption and slack intervals such that the system temperature is minimized. For tasks without deadlines the basic interleaving strategy is studied in [16]. The intuition behind the task interleaving scheme is that tasks with low *thermal footprint* (Activity) are executed when the system temperature is high while tasks with high *thermal footprint* are executed when the system temperature is low; where thermal footprint can be considered to be the thermal effect of executing the task. This scheduling problem is, however, more complex in real time systems because of existence of deadlines. Tasks cannot be interleaved arbitrarily and we have to ensure that no tasks miss their deadlines. Furthermore, the periodic nature of the tasks considered in this work introduces additional challenges. The maximum temperature across different hyperperiods may be different. To minimize system temperature, we have to reduce temperature across the hyperperiod which has the highest initial temperature.

To motivate the proposed scheme, we present the following example: Consider the scheduling of one periodic task τ ($C_\tau = 200$ ms, $T_\tau = 400$ ms, $A_\tau = 100$) executed at speed 1. Therefore, the dynamic power consumption for τ is 100 W.

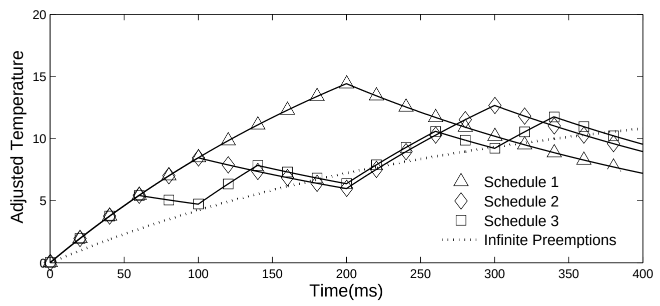

Fig. 2: Thermal profiles for hyperperiod at initial temperature = 0

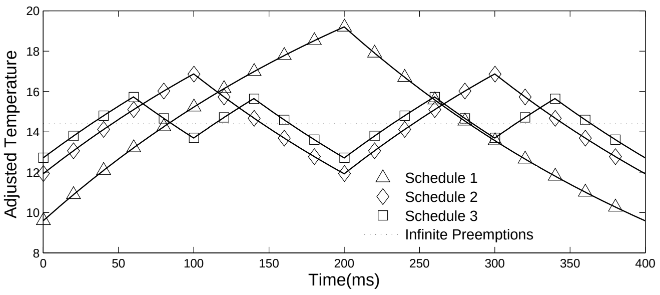

Fig. 3: Thermal profiles for hyperperiod at thermal steady state

We now analyze three different ways of executing τ (Figure 1) and the thermal effect of the scheduling decisions. Figure 2 shows the processor temperature after execution of each schedule in the first hyperperiod. Without loss of generality, the initial processor temperature is assumed to be 0. If the actual initial temperature is θ_0, then all temperatures shown here will be higher by $\theta_0 e^{-\beta t}$. Furthermore, if we periodically execute a given schedule, the temperature at the start of the hyperperiod increases monotonically until the temperature at the start and end of hyperperiod is approximately equal. This scenario where the temperature at the start of hyperperiod stops increasing is called *thermal steady state*. Figure 3 shows the temperature due to the execution of each of the three schedules at thermal steady state. It is known from [9] that $\lim_{m\to\infty} \theta(mL) = \dfrac{\eta_\Pi(L)}{1-\mathrm{e}^{-\beta L}} = \theta((m+1)L)$. For a given periodic schedule Π, we call the temperature at the start of the hyperperiod at thermal steady state $\theta_\Pi = \frac{\eta_\Pi(L)}{1-\mathrm{e}^{-\beta L}}$ We are interested in minimizing the temperature due to execution of periodic tasks at thermal steady state. This is because, if only periodic tasks are scheduled, the system temperature at the start of the hyperperiod will never exceed the corresponding temperature at thermal steady state; subject to the assumption that all tasks consume at most their worst case power and the ambient conditions do not change. Therefore, if no thermal violations occur at thermal steady state, the periodic schedule is deemed thermally feasible.

As shown in Figure 3, the maximum temperature temperature due to the execution of periodic task is reduced if power/execution is distributed more uniformly across the hyperperiod. Figure 2 and 3 also show thermal profile of the schedule with infinite number of preemptions and power evenly distributed across the entire hyperperiod. This schedule has the lowest temperature (14.39) and, as the next section will prove, is also the lower bound on maximum system temperature due to the execution of periodic task τ.

This example presents a very simple scenario where there is only one periodic task. Therefore, there is no power variation and the available slack can be interleaved with task execution arbitrarily without violation of task deadline. However, the situation becomes more complex if the periodic task set consists of multiple periodic tasks with varying activity factors and periods/deadlines. To apply the scheduling principle to this more general model we need:1) a mechanism to find if execution of periodic tasks can be delayed without causing any deadline violation (Slack value), and 2) if slack is available, a mechanism to find the periodic task for execution such that maximum temperature for the system can be reduced.

The next subsection covers an algorithm to estimate the available slack at any point in the periodic schedule. Section III-B presents the concept of thermal impact/utilization presented in [13]. This concept is used in this work for selection of periodic tasks to execute when slack is available; such that system temperature can be reduced.

A. Slack Evaluation

This section presents an algorithm for evaluating the time by which the execution of periodic tasks can be delayed without causing any deadline misses. We first present an exact algorithm for *slack* evaluation; followed by a slack approximation scheme, with reduced computation complexity.

Definition 1: Given a periodic schedule Π, $\mathrm{Slack}_\Pi(t)$ is defined as the maximum time for which the processor can idle at time t without causing any future deadline violation.

1) Exact Slack Evaluation Scheme (ESES): ESES is based on the Earliest Deadline Latest (EDL) algorithm [17]. In this algorithm, each task instance is executed as late as possible and tasks are given priority based on their deadlines. An important property for understanding EDL schedule is that it is the *mirror image* of the Earliest Deadline First (EDF) schedule taken across the end of the hyperperiod. i.e., suppose that $\Pi_{\mathrm{EDF}}(t)$ represents the EDF schedule and $\Pi_{\mathrm{EDL}}(t)$ represents the EDL schedule, then: $\Pi_{\mathrm{EDL}}(t) = \Pi_{\mathrm{EDF}}(L-t)$

Suppose that $\Omega_\Pi(t)$ is a set of all periodic task instances, including partial instances, within the hyperperiod which have not been scheduled until time t in schedule Π. To find the value of slack at time t, EDL schedule is constructed for all task instances and partial instances in $\Omega_\Pi(t)$. Suppose that the constructed EDL schedule starts execution of the first task instance in $\Omega_\Pi(t)$ at time t_{EDL}, then the slack at time t is given by: $\mathrm{Slack}_\Pi(t) = t_{\mathrm{EDL}} - t$. Computation complexity of ESES is dependent on the number of periodic task instances in the hyperperiod. Depending on hyperperiod length, this can be very high; making the application of ESES infeasible at run time. We now present a scheme which gives an estimate of the available slack with reduced computation overhead.

2) Slack Approximation Scheme (SAS): SAS is also based on EDL scheduling scheme [17]. However, in the approximation scheme, the EDL schedule needs to be computed only once (can be done at design time). At runtime, evaluating the amount of slack available involves $O(n)$ comparison

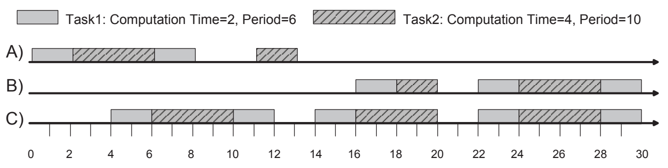

Fig. 4: A) Schedule Π until time t. B) EDL schedule for task instances not executed in Π until time t. C) EDL schedule for the entire hyperperiod

operation, where n is the number of tasks in the periodic task set. Note that this is significantly less compared to ESES, whose complexity was dependent on the number of periodic task *instances* in the hyperperiod. Suppose that for a given periodic task set Γ, Π_{EDL} represents the EDL schedule and Π represents a given schedule. Also suppose that $\Lambda_{i,\Pi}(t)$ represents the computations performed for task i in schedule Π until time t. Likewise $\Lambda_{i,\mathrm{EDL}}(t)$ represents the computation performed for i in schedule Π_{EDL} until time t. Then we can estimate slack by the following theorem:

Theorem 1: The available slack at time t is at least equal to the difference in time between t and the time at which at least one of the periodic tasks has more computations performed in EDL schedule compared to the schedule Π i.e.,

$$\mathrm{Slack}_{\Pi}(t) \geq \min_{i\in\Gamma}\{u_i\} - t \tag{5}$$

where $u_i = \max_{0\leq u\leq L}\{u : \Lambda_{i,\Pi}(u) < \Lambda_{i,\mathrm{EDL}}(u)\}$.

Proof: If R.H.S of Equation 5 is positive at time t, it is clear that schedule Π has performed at least as many computations for all periodic tasks as the computations performed in schedule Π_{EDL}. Furthermore, the value of u returned by Equation 5 is the first instant in time where the computations performed for one of the tasks is greater in EDL compared to its computations performed in periodic schedule Π. Since Π_{EDL} has no deadline violations, Equation 5 gives a lower bound on the available slack. ■

3) Example: We now present an example to illustrate the use of ESES and SAS schemes. Suppose that the periodic task set consists of two periodic tasks *Task1* and *Task2* with computation times 2 and 4 and periods/relative deadlines 4 and 10 respectively. Also suppose that we have scheduled the periodic tasks until time 13 as shown in Figure 4A. At time 13, we wish to evaluate the available slack. For this evaluation, ESES will construct the EDL schedule for all periodic task instances which have not been executed in Π until time 13. This EDL schedule is shown in Figure 4B. As shown in the figure, this partial EDL schedule starts executing its first task at time 16. Therefore, the available slack: $\mathrm{Slack}_{\Pi}(13) = 16 - 13 = 3$ units. SAS estimates slack based on a precomputed EDL schedule for the entire hyperperiod. This EDL schedule is shown in Figure 4C. Based on the figure the maximum value of u for *Task1* is 14. The corresponding maximum value for *Task2* is 18. Since we take the minimum of these values, the estimated value of slack is $14 - 13 = 1$.

B. Thermal Utilization/Impact

This section summarizes the relevant aspects of the characterization called Total Thermal Impact (TTI) [13] defined as follows.

$$\omega_{\tau,s}(\infty) = \int_0^{\infty} \eta_{\tau,s}(u)\mathrm{d}u \tag{6}$$

In our paper, TTI is used for selecting a task to execute when slack is available; such that the system temperature is reduced. Based on the thermal model,

$$\omega_{\tau,s}(\infty) = \frac{A_\tau C_\tau s^2}{\beta} \tag{7}$$

Theorem 2: Given a periodic task set Γ and schedule $\Pi(t)$. Also suppose that all tasks are executed at speed s and $\theta(0) = \theta_{\Pi}$, then

$$\int_0^L \frac{\eta_{\Pi}(L)}{1-\mathrm{e}^{-\beta L}} + \eta_{\Pi}(u) = \sum_{i\in\Gamma} \frac{L}{T_i}\omega_{i,s}(\infty) \tag{8}$$

The LHS of Equation 8 represents the integral of processor temperature during one hyperperiod at thermal steady state. Note that the value of this integral (RHS of Equation 8) is independent on the scheduling scheme used. The value of this integral is only dependent on the TTI of individual periodic tasks and their periods.

Theorem 3: Given a periodic task set Γ with schedule Π. Let $\phi(\Pi)$ be the maximum temperature during the periodic schedule when initial temperature $\theta(0) = \theta_{\Pi}$, the steady state temperature i.e., $\phi(\Pi) = \max_{0\leq t\leq L}(\theta_{\Pi}\mathrm{e}^{-\beta t} + \eta_{\Pi}(t))$, then, assuming all tasks are executed at speed s:

$$\phi(\Pi) \geq \sum_{i=0}^{n-1} \frac{\omega_{i,s}(\infty)}{T_i} \tag{9}$$

This theorem gives us a lower bound on the maximum temperature of periodic task set Γ at thermal steady state when all tasks are executed at speed s. This lower bound is used in the proposed scheduling scheme to schedule periodic tasks such that maximum temperature is minimized.

C. Proposed Solution

In this section, we first formulate our problem as a Mixed Integer Linear Programming optimization. We then present an algorithm called Power Redistribution Algorithm (PRA) and compare the results with an optimal solution.

MILP Formulation: Let Ω be a set of all periodic task instances in task set Γ; where each periodic task instance (i) is defined by Release Time (R_i), absolute deadline (D_i), Activity (A_i) and Computation demand (C_i) . It is also assumed that the scheduler can make scheduling decisions for contiguous time periods of length ϵ, i.e., if the scheduler decides to execute task τ at time t, τ will be executed in the interval $[t, t+\epsilon)$ without any preemptions. At time $t+\epsilon$, the scheduler will again pick a task to schedule for the next ϵ length time interval and so on. The value of ϵ is dependent of the scheduling overhead which can be tolerated. Generally, a low value of ϵ improves the solution since power can be distributed at a finer granularity. However, for a given hyperperiod length, lower ϵ value also means larger number of scheduling intervals; increasing the scheduling overhead/problem size. Based on this, we can have

Algorithm 1 Power Redistribution Algorithm for minimizing maximum temperature of a given periodic task set

```
Input: Periodic task set $\Gamma$ and temperature reading $\theta_\mathrm{S}$ at the start of hyperperiod
Output: Periodic schedule $\Pi(t)$
1: R(t): Set of ready task instances including idle task, at time t
2: $\omega_\mathrm{R} = \sum_{i\in\Gamma} \frac{\omega_{i,1}(\infty)L}{T_i} - \frac{1}{\beta}(\theta_\Pi - \theta_\mathrm{S})(1-e^{-\beta L})$
3: $\theta(t,i) = \theta(t-\epsilon)e^{-\beta\epsilon} + \frac{A_i}{\beta}(1-e^{-\beta\epsilon})$   ▷ temperature at time $t$ if task instance $i$ is executed in interval $[t-\epsilon, t)$
4: $\theta(0) = \theta_\mathrm{S}$, $\theta_\mathrm{Max} = \theta_\mathrm{S}$, $t \leftarrow \epsilon$, $j \leftarrow$ idle
5: while $t \leq L$ do
6:     Compute slack
7:     $\theta_\mathrm{Target} = \max\{\omega_\mathrm{R}/L,\ \theta_\mathrm{Max}\}$
8:     $j = \begin{cases} \underset{i\in R}{\text{argmin}}\{|\theta(t,i) - \theta_\mathrm{Target}|\} & \text{if} Slack > 0 \\ \underset{i\in\{R/idle\}}{\text{argmin}}\ \{D_i\}\} & \text{otherwise} \end{cases}$
9:     Add $j$ to schedule $\Pi$ in the interval $[t-\epsilon, t)$
10:    $\omega_\mathrm{R} = \omega_\mathrm{R} - \frac{\theta(t-\epsilon)-\theta(t,j)+A_j\epsilon}{\beta}$
11:    $t \leftarrow t + \epsilon$
12:    $\theta_\mathrm{Max} = \max\{\theta_\mathrm{max},\ \theta(t,j)\}$
13: end while
```

the following MILP formulation for this optimization problem: let L be the hyperperiod length for periodic task set Γ. Define the following variables:

- Y: Set of all ϵ length scheduling intervals in L. $Y = t_1, t_2 .... t_k$ where $k = L/\epsilon$
- $x_{i,j} = \begin{cases} 1 & \text{If task } i \text{ is executed in scheduling interval } t_j \\ 0 & \text{otherwise} \end{cases}$
- H_i: Heating constant for task i. $\mathrm{H}_i = \frac{A_i}{\beta}(1-e^{-\beta\epsilon})$.
- $\theta(j)$: Temperature at the end of scheduling interval t_j. $\theta(j) = \theta(j-1) \times \mathrm{e}^{-\beta\epsilon} + \sum_{i\in\Gamma} x(i,j) \times \mathrm{H}_i$
- ϕ: Maximum temperature. $\phi \geq \theta(j)\ \ \forall t_j \in Y$

The solution involves minimizing ϕ subject to:

$$\sum_{i\in\psi} x_{i,j} \leq 1 \qquad \forall t_j \in Y \tag{10}$$

$$\sum_{t\epsilon\leq R_i} x_{i,j} = 0 \qquad \forall i \in \psi \tag{11}$$

$$\sum_{t\epsilon\leq D_i} x_{i,j}\epsilon = C_i \qquad \forall i \in \psi \tag{12}$$

$$\theta(0) \geq \theta(L/\epsilon) \tag{13}$$

Equation 10 ensures that two or more task instances do not execute simultaneously. Equation 11 ensures that task instances are not executed before they are released. Equation 12 ensures that all task instances complete on or before their deadline. Equation 13 ensures that the temperature is minimized at *thermal steady state*.

Power Redistribution Algorithm: PRA attempts to minimize the maximum temperature within the hyperperiod by reducing the variation in temperature across the hyperperiod. If the temperature at the start of hyperperiod is θ_Π (Thermal steady state) and there is no variation in temperature across the hyperperiod, we achieve the lower bound on maximum temperature as specified in Theorem 3. Algorithm 1 takes the periodic task set Γ and processor temperature at the start of hyperperiod (θ_S) as input and gives the schedule ($\Pi(t)$) as output. θ_S can be a temperature reading from an on-chip thermal sensor. ω_R represents the remaining Thermal Impact in the hyperperiod. The while loop (lines 5-11) iterates through all the scheduling intervals, choosing to execute, either one of the ready periodic tasks, or *idle* task in the current scheduling interval. In the while loop, first the available slack is computed as explained in Section III-A (Line 6), by either ESES or SAS algorithms. Target Temperature (θ_Target) is then assigned a value in line 7. If slack is available, PRA selects the task which brings temperature closest to θ_Target. Tasks have different impact on system temperature because of differences in their *Activity*. We know from Theorem 3, that θ_Target is the minimum value of maximum temperature across the hyperperiod at thermal steady state. Therefore, we can *greedily* schedule periodic tasks such that temperature is as close to θ_Target as possible. If slack is not available, periodic task instance with minimum deadline is chosen to avoid any deadline violations. Line 9 of Algorithm 1 updates the value ω_R. Note that, reading temperature sensor once at the start of the hyperperiod is not a restriction of PRA algorithm. On the contrary, multiple sensor readings within the hyperperiod will result in lower temperature estimation errors; resulting in higher scheduling gains.

IV. Results

In this section we present the results for the proposed scheme and compare against the optimal solution given by the MILP method. We also provide complexity analysis by analyzing the time taken to solve the MILP (Optimal) problem compared to the the time taken by PRA.

1) Setup: Experiments are conducted with various periodic task sets with varying hyperpeiod lengths and number of tasks. The thermal parameters of the system are given at the end of Section II-B. Additionally, $\epsilon = 10$ msec and hyperperiod length varies between 1 sec to 40 sec. Therefore, the number of ϵ length scheduling intervals varies between 100 to 4000. More number of scheduling intervals are not simulated because the MILP solver reaches its resource limits for larger problem sizes and does not yield an optimal solution. The number of tasks in the periodic task set vary between 10-20. Following scheduling schemes are simulated for each periodic task set:

Optimal: Thermally optimal schedule (MILP solution).

PRA_ESES: PRA with Exact Slack Evaluation Scheme used to compute slack.

PRA_SAS: PRA with Slack Approximation Scheme used to compute slack.

EDF: Earliest Deadline First scheduling algorithm.

Figure 5 shows the histogram of the difference in maximum temperature for schedules given by PRA_ESES, PRA_SAS and EDF scheme compared to the thermally optimal schedule. As shown in Figure 5, PRA_ESES scheme has a high percentage of task sets where the difference in maximum temperature

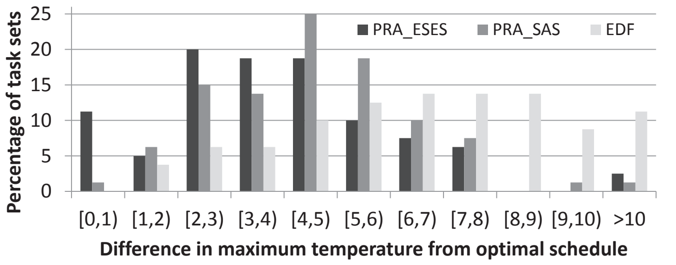

Fig. 5: Histogram of the difference in maximum temperature of schedules given by PRA_ESES, PRA_SAS and EDF compared to the thermally optimal schedules

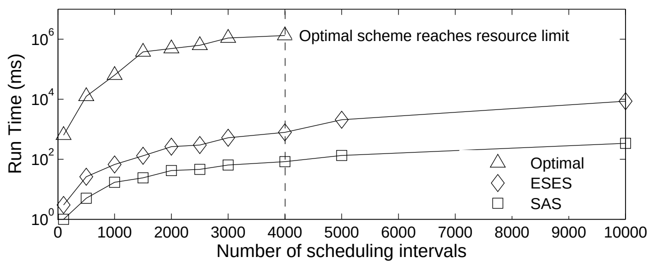

Fig. 6: Run times for Optimal, PRA_ESES and PRA_SAS schemes

from the thermally optimal schedule is 0-5. This scheme performs best out of the proposed schemes. PRA_SAS has lower performance, with high percentage of tasks distributed across the temperature difference range of 2-6. Although PRA_SAS uses the same algorithm for selecting tasks as PRA_ESES, it is conservative in the estimation of slack, which results in this difference in performance. EDF performs poorly, with a high percentage of tasks having difference in temperature of more than 5, from a thermally optimal schedule.

Another analysis of interest is the computation complexity of the proposed approaches (Optimal, PRA_ESES, PRA_SAS). We analyze this by recording the run-time each approach takes to arrive at its respective schedule. Figure 6 shows that the optimization problem does not scale well as the number of scheduling intervals increase. The run-time of solving the optimization problem is three orders of magnitude higher compared to the runtime of PRA_ESES. Furthermore, we were unable to solve optimization problems with number of scheduling intervals more than 4000. This is because the MILP solver reached its resource limit at higher problem size. Runtime of PRA_ESES is one order of magnitude higher compared to the runtime of PRA_SAS. This is because PRA_ESES constructs a new EDL schedule each time it has to evaluate slack as explained in Section III-A1. This is computationally more expensive compared to slack approximation scheme used by PRA_SAS (Order(n) comparison operations where n is the number of tasks). Therefore, PRA_SAS has the lowest computation complexity. However, this low computation complexity comes at the cost of higher system temperature as shown in Figure 5. Low computation complexity enables the use of PRA for dynamic reclaiming strategies which may be used to reduce system temperature further. Evaluation of dynamic reclaiming strategies and extension of the proposed schemes/concepts to multi-core processing environment is part of future work.

V. Conclusion

This work presents an algorithm (PRA) for scheduling periodic real time tasks such that the system temperature is reduced. In contrast to the existing schemes, PRA does not depend on speed scaling and uses a more realistic task power model. The presented algorithm also has low scheduling overhead allowing its application at run-time; and is shown to perform close to a thermally optimal schedule.

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] S. Wang, Bettati, and Riccardo, "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.

[6] V. Chaturvedi, H. Huang, and G. Quan, "Leakage aware scheduling on maximum temperature minimization for periodic hard real-time systems," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1802–1809.

[7] 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.

[8] S. Wang, J.-J. Chen, Z. Shi, and L. Thiele, "Energy-efficient speed scheduling for real-time tasks under thermal constraints," in *Embedded and Real-Time Computing Systems and Applications, 2009. RTCSA'09. 15th IEEE International Conference on*. IEEE, 2009, pp. 201–209.

[9] G. Quan, Y. Zhang, W. Wiles, and P. Pei, "Guaranteed scheduling for repetitive hard real-time tasks under the maximal temperature constraint," in *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*. ACM, 2008, pp. 267–272.

[10] L. Schor, H. Yang, I. Bacivarov, and L. Thiele, "Thermal-aware task assignment for real-time applications on multi-core systems," in *Formal Methods for Components and Objects*. Springer, 2013, pp. 294–313.

[11] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 4. IEEE, 2000, pp. 101–104.

[12] X. Feng, R. Ge, and K. Cameron, "Power and energy profiling of scientific applications on distributed systems," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, pp. 34–34.

[13] R. Ahmed, P. Ramanathan, and K. Saluja, "On thermal utilization of periodic task sets in uni-processor systems," in *Real-Time Computing Systems and Applications*. IEEE, 2013.

[14] Y. Liu, R. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *Design, Automation Test in Europe*, 2007, pp. 1–6.

[15] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proceedings of 30th Annual International Symposium on Computer Architecture*, Jun. 2003, pp. 2 – 13.

[16] E. Kursun, C. yong Cher, A. Buyuktosunoglu, and P. Bose, "Investigating the effects of task scheduling on thermal behavior," in *In Third Workshop on Temperature-Aware Computer Systems (TACS06*, 2006.

[17] H. Chetto and M. Chetto, "Some results of the earliest deadline scheduling algorithm," *IEEE Transactions on Software Engineering*, vol. 15, no. 10, pp. 1261–1269, Oct. 1989.