Thermally Optimal Stop-Go Scheduling of Task Graphs with Real-Time Constraints

Pratyush Kumar Lothar Thiele

Computer Engineering and Networks Laboratory,
ETH Zürich, Switzerland
{pratyush.kumar,lothar.thiele}@tik.ee.ethz.ch

Abstract— Dynamic thermal management (DTM) techniques to manage the load on a system to avoid thermal hazards are soon becoming mainstream in today's systems. With the increasing percentage of leakage power, switching off the processors is becoming a viable alternative technique to speed scaling. For real-time applications, it is crucial that under such techniques the system still meets the performance constraints. In this paper we study stop-go scheduling to minimize peak temperature when scheduling an application, modeled as a task-graph, within a given makespan constraint. For a given static-ordering of execution of the tasks, we derive the optimal schedule referred to as the JUST schedule. We prove that for periodic task-graphs, the optimal temperature is independent of the chosen static-ordering when following the proposed JUST schedule. Simulation experiments validate the theoretical results.

I. Introduction

Power densities in today's electronic systems continue to rise due to (a) continued, though slower, feature size reduction, and (b) increasing functional complexity. Higher power densities directly translate to higher on-chip temperatures which can affect system reliability and even functional correctness [8]. Hardware cooling solutions, have not been able to keep pace with the rising on-chip temperatures [10]. To supplement such techniques, architectural-level software techniques, broadly referred to as Dynamic Thermal Management (DTM), have been widely studied [4, 6].

Based on the classification presented in [6], dynamic voltage scaling (DVS) and stop-go scheduling, also referred to as Dynamic Power Management (DPM) [2], are the two main classes of DTM techniques. In DVS, thermal hazards are avoided by adjusting, when necessary, the supply voltage and/or frequency of the system. On the other hand, stop-go scheduling puts the system in a low-power state to reduce on-chip temperature.

Naturally, either class of above DTM techniques directly impacts the real-time performance of applications. This has led to a large number of studies that aim to either maximize performance under thermal constraints, or minimize temperature under performance constraints. Bansal et al. [1] identify the maximum workload available in a time window without violating given thermal constraints. In [11], a reactive speed control scheme to minimize temperature and corresponding schedulability tests were proposed. The on-line energy reduction algorithm proposed by Yao et al. was adapted to reduce peak temperature [1]. Approximation algorithms were proposed in [13] to schedule a set of periodic tasks with each task running at a constant frequency. Chantem et al. [5] study the maximization of workload for DVS with discrete speeds and transition overhead.

Most of these studies consider DVS as the chosen DTM technique. This choice is influenced by the CMOS property that a linear reduction in the supply voltage results in a cubic reduction of the power consumption at the expense of a linear slow down in the processor frequency. However, as the leakage power becomes increasingly comparable to the dynamic power of a system [3], this is less true. Leakage power can, however, be reduced by stop-go techniques: by putting the system into a sleep state. Furthermore, for peripheral devices like I/O systems, memories and interconnects and for low-end systems, DVS features are not available, while temperature can still be managed using stop-go techniques.

Further, the above studies consider individual tasks with independent real-time deadlines. However, in reality, most applications are based on dataflow models of computation, such as task graphs, with a cumulative real-time performance constraint such as its makespan. The execution of a task depends on the execution of other tasks through precedence constraints. This broadens the scope of DTM techniques, as different orders of execution of tasks can potentially be used to reduce temperature, while still meeting the cumulative performance constraint.

In this paper, we study stop-go scheduling of a given task-graph under a makespan constraint while optimally minimizing the peak temperature. For non-preemptive static-order scheduling of the task-graph we derive the optimal stop-go schedule referred to as JUST schedule. When the static-ordering of tasks is unknown, we present an approximate iterative algorithm to determine the schedule that minimizes temperature by formulating it as a binary integer program. For periodic task graphs, we prove that the minimum peak temperature of the system is obtained when following the JUST policy and crucially is independent of the chosen static-ordering of tasks. With quantitative results on synthetic and real applications we substantiate the derived theoretical results.

The rest of the paper is organized as follows. We detail the system model and describe the problem in Section II. In Section III, we derive the main results of this paper by designing the thermally optimal scheduling of a task-graph. We extend this result to periodic task graphs in Section IV. Finally, in Section V, we present experimental results.

II. System model and problem statement

A. Hardware and thermal model

We consider the model of heat generation in the system based on the Fourier law of heating, which can be expressed in the following equation:

$$C\frac{dT}{dt} = P - G(T - T^0) \tag{1}$$

where T is the temperature of the system, C is the thermal capacity of the system, G is the conductance to the ambient, T^0 is the ambient temperature, and P is the generated power.

The leakage current on a VLSI system is a function of temperature itself: at higher temperature the leakage current is larger. Thus, the power consumption of the system is a function of temperature. This can be expressed as a linear model [7]

$$P = \alpha T + \beta \tag{2}$$

Including this dependence, the original differential equation (1) can be re-written as

$$C\frac{dT}{dt} = -(G - \alpha)T + (\beta + GT^0) \tag{3}$$

To simplify notation we shorten the above equation as

$$\frac{dT}{dt} = -aT + b, \text{with} \quad a = \frac{G - \alpha}{C}\ ,\ b = \frac{\beta + GT^0}{C} \tag{4}$$

The closed form solution to this differential equation is

$$T(t) = T^\infty + (T(t_0) - T^\infty) \cdot e^{-a(t - t_0)} \tag{5}$$

where $T^\infty = b/a$ is the steady state temperature.

When the system is executing tasks it is said to be in the *active* mode and when it is not executing tasks it is said to be in *idle* mode. The power consumption in these modes is different. We distinguish between the parameters of (2) corresponding to the active and idle modes as α^{act}, β^{act} and α^{idl}, β^{idl}, respectively. The corresponding a and b values (as shown in (4)) will also be referred to with the superscripts to denote the mode of the system. The steady state temperatures (T^∞ in (5)) will be denoted as T^{act} and T^{idl} for the active and idle modes, respectively. We introduce the following functions

$$f^{act}(t) = e^{-a^{act}t}, \qquad f^{idl}(t) = e^{-a^{idl}t} \tag{6}$$

The thermal properties of the system are completely specified by the tuple $\mathbf{T} = (C, G, a^{act}, b^{act}, a^{idl}, b^{idl})$. Instead of using the a and b parameters, we will use the derived values T^{act}, T^{idl} and the derived functions f^{act}, f^{idl}, in the rest of the paper.

B. Application model

The application is specified as a *task graph* $G = (V, E)$, where the vertices, V, denote tasks to be performed and directed edges E, denote precedence constraints. To allow for a deadlock-free execution, such a graph must be free of directed cycles, and is thus commonly referred to as a Directed Acyclic Graph (DAG). The function $\tau : V \rightarrow \Re^+$, identifies for every task $v \in V$, the time taken to execute it, on the considered system. It is required that the task-graph be executed within ω units of time, i.e. ω is the maximum allowed makespan. The application is thus fully specified by $\mathbf{A} = (G, \tau, \omega)$.

C. Scheduling

We consider a system where a switch from active to idle state cannot preempt a running task. Allowing for such preemption would necessitate hardware and software features to restore the context of the earlier running process. This is beyond the intended scope of this work. Thus, in our model, the system can move to idle state only after completing a task and before starting the next task. Scheduling the task graph on such a system involves a series of two choices: (a) the choice of task to be executed next, and (b) the choice of the amount of time (if any) the system must remain in idle state before the execution of that task. We refer to such a schedule as a *non-preemptive stop-and-go schedule*. Such a schedule, $\mathbf{S}$, is specified by an ordered list $\mathbf{S} = \left((\sigma_1, t_1^{idl}), \ldots, (\sigma_{|V|}, t_{|V|}^{idl})\right)$ where $\sigma_n \in V$ denotes the nth task to be executed in the system and $t_n^{idl} (\geq 0)$ denotes the time for which the system is in idle state *before* executing this task. The values of σ_j must satisfy the precedence constraints.

D. Problem statement

The problem that this paper solves is the following:
Given the thermal properties of a system $\mathbf{T}$*, the ambient temperature* T^0*, the initial temperature of the system* T_0*, and the properties of an application* $\mathbf{A}$*. Determine a non-preemptive stop-and-go schedule* $\mathbf{S}$ *such that (a) the finish-time of all tasks are within a given makespan* ω*, and (b) the peak temperature of the system is minimized.*

III. Optimal Stop-Go Schedule

We divide the solution into two parts. In Section A, we assume that an ordering of tasks σ is already given. Given such an order σ, we find the idle times, t_n^{idl}, which satisfy the makespan constraint and minimize the peak temperature. In Section B, we present a method to compute an ordering if none is given.

A. Choice of Optimal Idle Times for A Given Static-Ordering of Tasks

Similar, to the notation t_n^{idl}, let t_n^{act} be the execution time of the nth task for given static order σ:

$$t_n^{act} = \tau(\sigma_n), \quad n = 1, \ldots, |V| \tag{7}$$

Let T_j denote the temperature of the system at the *end* of execution of the jth task.

Theorem 1 *Suppose an arbitrary non-preemptive stop-go schedule with given static-ordering of tasks,* σ*. Then, any change of the idle times such that none of* $T_j,\ j \in \{1, \ldots, |V|\}$ *decreases, and at least one increases, decreases the makespan.*

Proof Writing (5) for the jth idle time and the execution of the jth task, and eliminating the temperature at the end of the idle time, we have

$$T_j = \left(T_{j-1} f^{idl}(t_j^{idl}) + T^{idl}(1 - f^{idl}(t_j^{idl}))\right) f^{act}(t_j^{act}) + T^{act}(1 - f^{act}(t_j^{act})) \tag{8}$$

$$\Rightarrow \quad f^{idl}(t_j^{idl}) = \frac{1}{f^{act}(t_j^{act})} \cdot \frac{T_j - T'_j}{T_{j-1} - T^{idl}}, \text{where} \tag{9}$$

$$T'_j = T^{idl} f^{act}(t_j^{act}) + T^{act}(1 - f^{act}(t_j^{act})) \tag{10}$$

Multiplying (9) for $j \in \{1, \ldots, |V|\}$, define F as

$$F = \prod_{j=1}^{|V|} f^{idl}(t_j^{idl})$$
$$= \frac{1}{\prod_{j=1}^{|V|} f^{act}(t_j^{act})} \frac{T_{|V|} - T'_{|V|}}{T_0 - T^{idl}} \prod_{j=1}^{|V|-1} \frac{T_j - T'_j}{T_j - T^{idl}} \quad (11)$$

From the definition of f^{idl} in (6) define c_1 as

$$c_1 = \prod_{j=1}^{|V|} f^{act}(t_j^{act}) = f^{act}\left(\sum_{v \in V} \tau(v)\right)$$

where c_1 is a constant for the specified application. Further, T_0 is a given quantity. Hence, we can define c_2 as

$$c_2 = \prod_{j=1}^{|V|} f^{act}(t_j^{act}) \cdot \left(T_0 - T^{idl}\right)$$

where c_2 is a constant for the given problem specification. Thus, (11) becomes,

$$F = c_2^{-1} \cdot \left(T_{|V|} - T'_{|V|}\right) \cdot \prod_{j=1}^{|V|-1} \frac{T_j - T'_j}{T_j - T^{idl}}$$

From the above it is clear that

$$\frac{\partial F}{\partial T_{|V|}} > 0 \quad (12)$$

Interpreting (10) using (5), we infer that that T'_j denotes the final temperature of the system when run in active mode for t_j^{act} amount of time starting from T^{idl} as initial temperature. Thus, we obtain

$$T'_j > T^{idl}, \quad \forall\, j \in \{1, \ldots, |V|\} \quad (13)$$

A function $h(x)$ defined as $h(x) = (x-a)/(x-b)$ is monotonically increasing iff $a > b$. Thus, we have

$$\frac{\partial F}{\partial T_j} > 0, \quad \forall\, j \in \{1, \ldots, |V| - 1\}. \quad (14)$$

Given the definition of f^{idl} in (6), we obtain

$$F = \prod_{j=1}^{|V|} f^{idl}(t_j^{idl}) = f^{idl}\left(\sum_{j=1}^{|V|} t_j^{idl}\right) = f^{idl}(\omega_{\mathbf{S}} - \tau_{tot}) \quad (15)$$

where $\omega_{\mathbf{S}}$ is the makespan when following the considered schedule and $\tau_{tot} = \sum_{v \in V} \tau(v)$. Since f^{idl} is a monotonically decreasing function, we have,

$$\frac{\partial F}{\partial \omega_{\mathbf{S}}} < 0 \quad (16)$$

From (12), (15) and (16), we have

$$\frac{\partial \omega_{\mathbf{S}}}{\partial T_j} < 0, \quad \forall\, j \in \{1, \ldots, |V|\}. \quad (17)$$

Changes in T_j can be brought about by changing the idle times. Thus, from (17), any change in idle times, such that none of $T_j, j \in \{1, \ldots, |V|\}$, decreases and at least one increases, would decrease the makespan. ∎

We use Theorem 1, to present an intuition for designing the optimal idle times. Let T^{opt} denote the minimum peak temperature under the given makespan constraint. Then, in the optimal schedule the temperature at the execution of some task must be T^{opt}. Let the $(p^{opt}+1)$th task in the static order σ, be the first such task. Theorem 1 tells us that in the optimal schedule the temperature at the end of execution of tasks, T_j, would be as close to T^{opt} as possible. Indeed, it is possible to have the temperatures at the end of all tasks after the $(p^{opt}+1)$th task exactly equal to T^{opt} by appropriately setting the idle times inserted before the tasks. Hence, the optimal schedule should have the temperatures $T_j = T^{opt}$, for all $j \in \{p^{opt}+1, \ldots |V|\}$, for some p^{opt}.

We now determine the value of p^{opt}. Clearly, the value of p^{opt} depends on the initial temperature T_0. For instance, if $T_0 = T^{opt}$, we would have $p^{opt} = 0$, as the very first task can end with temperature equal to T^{opt}. On the other hand, if we have a small value of a T_0 in comparison to T^{opt}, we may have a larger p^{opt}. Again, Theorem 1 tells us that p^{opt} must be as small as possible. The smallest p^{opt} is obtained when the processor runs the first p^{opt} tasks without any idle time inserted before these tasks. Define variable T_k^{cont} as the final temperature of the processor after running the first k tasks continuously, i.e. without any idle time before the tasks, having started from the given initial temperature T_0. Then we have

$$T_k^{cont} = T_0 f^{act}(\delta_k) + T^{act}(1 - f^{act}(\delta_k)). \quad (18)$$

The desired p^{opt} is now given by the following relation

$$T_{p^{opt}}^{cont} \leq T^{opt} \leq T_{p^{opt}+1}^{cont} \quad (19)$$

We can now impose the constraint that the makespan of the schedule is ω. Given that the first p^{opt} tasks run with no idle times and all subsequent tasks end with temperatures T^{opt}, we can derive the following equation similar in form to (11)

$$\frac{T^{opt} - T'_{p^{opt}+1}}{f^{act}(t_{p^{opt}+1}^{act})(T_{p^{opt}}^{cont} - T^{idl})} \cdot \prod_{j=p^{opt}+2}^{|V|} \frac{T^{opt} - T'_j}{f^{act}(t_j^{act})(T^{opt} - T^{idl})} = f^{idl}(\omega - \tau_{tot}). \quad (20)$$

We are now ready to define the optimal schedule for given ordering of tasks.

Definition 1 *A JUst Sufficient Throttling(σ) (JUST(σ)) is a non-preemptive stop-go schedule, which follows the static-order of tasks σ, and has*

- $t_j^{idl} = 0 \;\; \forall\, j \leq p^{opt}$,
- $T_j = T^{opt} \;\; \forall\, p^{opt} < j \leq |V|$.

where p^{opt} and T^{opt} are related by (19) and (20).

As discussed, a JUST schedule consists of two distinct phases. In the first phase the first p^{opt} tasks are run without any idle time before the tasks, and in the second phase all remaining tasks are executed by inserting idle times before them such that the temperatures at the end of executing tasks is T^{opt}. We now prove the optimality of the JUST schedule.

Theorem 2 *Among all stop-go schedules following a given static-ordering of tasks, σ, and satisfying a given makespan constraint ω, JUST(σ) has the smallest peak temperature.*

Proof We prove this by contradiction. Let there exist another stop-go schedule, $\mathbf{S}'$ with peak temperature less than T^{opt} and makespan not greater than ω.
Let T_j^{opt} and $T_j^{\mathbf{S}'}$ denote the temperature of the system at the end of the jth task, while executing according to the JUST(σ) and the $\mathbf{S}'$ schedules, respectively. As the system is not put to sleep at all for the first p^{opt} tasks in JUST(σ), we have

$$T_j^{\mathbf{S}'} \leq T_j^{opt}, \quad 1 \leq j \leq p^{opt} \tag{21}$$

For the remaining tasks, the JUST(σ) policy reaches the peak temperature. Thus,

$$T_j^{\mathbf{S}'} \leq T_j^{opt}, \quad p^{opt}+1 \leq j \leq |V| \tag{22}$$

Using (21) and (22), from Theorem 1, we have $\omega_{\mathbf{S}'} > \omega$. This contradicts the existence of schedule $\mathbf{S}'$. ∎

From Theorem 1, we know that increasing the maximum temperature of a schedule under the JUST policy would decrease the makespan. Thus, there is a unique value of T^{opt} for which (20) is tight. This value of T^{opt} can be found using binary search. However, instead of searching of T^{opt} we can equivalently search over p^{opt}, since p^{opt} and T^{opt} are monotonically related as given in (19). Binary search over p^{opt} is easier given the limited search space $\{0, \ldots, |V|\}$. Using this, for a given problem instance, we can numerically compute the optimal values T^{opt} and p^{opt} as shown in Algorithm 1.

Algorithm 1 Computing p^{opt} and T^{opt}

```
p^opt ← p^opt_initial
while true do
    Compute T^opt according to (20) for current p^opt
    if T^opt < T^cont_{p^opt} then
        Decrease p^opt
    else if T^opt > T^cont_{p^opt+1} then
        Increase p^opt
    else
        Return p^opt, T^opt
    end if
end while
```

B. Integrated choice of idle times and static-ordering

From (20), it is clear that, for any JUST(σ) schedule, a permutation of the tasks $\{\sigma_1, \ldots, \sigma_{p^{opt}}\}$ does not change either the makespan or the highest temperature. Similarly, a permutation of the tasks $\{\sigma_{p^{opt}+1}, \ldots, \sigma_{|V|}\}$ does not change either parameter. In particular, when p^{opt} is 0, all static-orders have the same makespan and peak temperature. Thus, searching on the space of orderings of tasks is equivalent to searching over possible ways of dividing the tasks, V, into two non-overlapping sets: V_1 and V_2, where tasks in V_1 are executed without any idle time and tasks in V_2 are executed with idle times based on the JUST policy. The precedence constraints of the task graph simplify to the following: no task in V_2 must be a predecessor of any task in V_1.

Finding the optimal V_1 and V_2 requires us to minimize T^{opt} across all valid sets V_1. However, the non-linear nature of (19) and (20) make this a computationally difficult problem. We start by solving a simpler problem: For a *given* target temperature T^t, determine an ordering σ^t, if it exists, that can satisfy the makespan constraint with maximum temperature $\leq T^t$. We can then conservatively approximate (20) as

$$\sum_{v \in V_2} \frac{T^t - T''_v}{f^{act}(t_j^{act})(T^t - T^{idl})} \geq f^{idl}(\omega - \tau_{tot}) \tag{23}$$

where $T''_v = T^{idl} f^{act}(\tau(v)) + T^{act}(1 - f^{act}(\tau(v)))$. Similarly, expressing (19), we have,

$$\sum_{v \in V_1} \tau(v) \geq (f^{act})^{-1}\left(\frac{T^{act} - T^t}{T^{act} - T_0}\right),$$

where $(f^{act})^{-1}(x) = -\ln(x)/a^{act}$, is the inverse of f^{act}. We can equivalently express the above constraint for the set V_2 as

$$\sum_{v \in V_2} \tau(v) \leq \tau_{tot} - (f^{act})^{-1}\left(\frac{T^{act} - T^t}{T^{act} - T_0}\right). \tag{24}$$

Consider a binary variable x_v for each $v \in V$, such that $x_v = 0$ denotes that $v \in V_1$ and $x_v = 1$ denotes that $v \in V_2$. Then the constraint that some $v1 \in V$ executes before some $v2 \in V$ can be expressed as $x_{v1} \leq x_{v2}$. Note that this simplicity in using only a binary program rather than an integer program is because in the JUST schedule the orderings of tasks within the sets V_1 and V_2 are not important. We can then cast (23) and (24) along with the precedence constraints of the task graph as an binary integer program (BIP) as shown below:

$$\begin{array}{lrcll} \text{Maximise} & 1 & & & \\ \text{subject to} & x_v & \in & \{0,1\} & \forall\, v \in V \\ & x_{v1} & \leq & x_{v2} & \forall\, (v1, v2) \in E \\ & A1^T x & \geq & b1 & \\ & A2^T x & \leq & b2 & \end{array}$$

where $A1_v = \dfrac{T^t - T''_v}{f^{act}(t_j^{act})(T^t - T^{idl})}$, $b1 = f^{idl}(\omega - \tau_{tot})$, $A2_v = \tau(v)$, $b2 = \tau^{tot} - (f^{act})^{-1}\left(\dfrac{T^{act} - T^t}{T^{act} - T_0}\right)$.
From the solution to the above BIP, if it exists, we can obtain the static-ordering of tasks, σ^t. We can then set the idle times based on JUST(σ^t). The peak temperature of the resultant schedule will be below the set target temperature T^t. To identify the smallest such T^t that leads to a valid JUST policy, we can perform a binary search on T^t, based on feasibility of the above BIP. We illustrate this in the experimental section.

IV. Periodically executed task graphs

In several practical scenarios, applications execute periodically, with the period equal to the maximum allowed makespan. Let us consider a task graph of such an application that is executed periodically with a period equal to ω. In each period we have an instance of the scheduling problem discussed in the previous section, with the same makespan constraints but with possibly different starting temperatures. Let $(T_0)_n$

denote the starting temperature of the system in the nth period. For each period we can use the JUST schedule corresponding to that initial temperature for some given or computed static ordering. For such a schedule, let $(T^{opt})_n$ denote the highest temperature of the system for the nth period. Let $\lim_{n\to\infty}(T^{opt})_n = (T^{opt})_\infty$, if the limit exists.

Theorem 3 *$(T^{opt})_\infty$ exists and it does not depend on the chosen static-ordering of tasks σ.*

Proof First we show that, $(T_0)_{n+1} > (T_0)_n$ implies that $(T^{opt})_{n+1} > (T^{opt})_n$. We show this by contradiction. Let $(T^{opt})_{n+1} \leq (T^{opt})_n$. Then, from (19) it follows that $(p^{opt})_{n+1} \leq (p^{opt})_n$. Then, with the above assumptions L.H.S. of (20) decreases, while it should be a constant. Hence, we have a contradiction. Similarly, one can show that $(T_0)_{n+1} < (T_0)_n$ implies that $(T^{opt})_{n+1} < (T^{opt})_n$.
Since, according to the JUST policy, the makespan equals ω, we have $(T_0)_{n+1} = (T^{opt})_n$. Thus, $(T^{opt})_n$ is a monotonic function of n. A fixed point is reached when $(T^0)_{n*} = (T^{opt})_{n*}$. Such a fixed point can be computed by substituting $p^{opt} = 0$ in (20). The corresponding equation defines $(T^{opt})_\infty$

$$\prod_{v\in V} \frac{(T^{opt})_\infty - T'_j}{((T^{opt})_\infty - T^{idl})} = f^{idl}(\omega - \tau_{tot}) \times f^{act}(\tau_{tot}) \quad (25)$$

As is clear from the equation above and given the fact that it is obtained for $p^{opt} = 0$, the value of $(T^{opt})_\infty$ is independent of the static-ordering of tasks σ. ∎

The above theorem implies that, if we use JUST policy in each period, the value of $(T^{opt})_n$ would monotonically either decrease or increase to the defined $(T^{opt})_\infty$. Indeed, if $(T^{opt})_\infty \leq T_0$, the highest temperature over all time would be T_0 and if $(T^{opt})_\infty > T_0$, the highest temperature over all time would be $(T^{opt})_\infty$. We have the following interesting result.

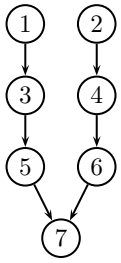
Corollary 4 *Under a JUST policy the highest temperature of a system running a periodic task graph is minimal and independent of the ordering of tasks.*

Thus, for a periodic task graph, as long as a JUST policy is used, the choice of the static-ordering of tasks is insignificant with respect to the peak temperature. The static ordering may be designed with consideration to other factors like required buffer capacity.

V. Experimental Results

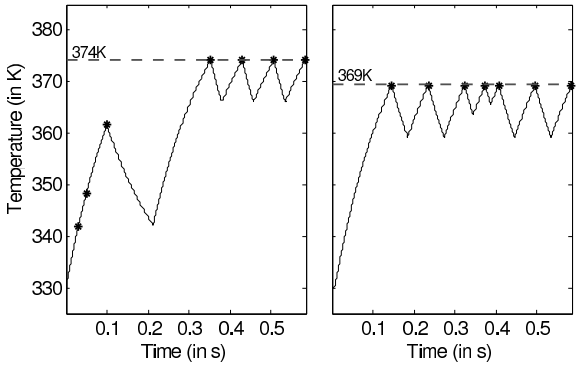
A. Thermal data

We source our thermal data from [12] for an ARM like processor. Typical numbers for such a processor are $C = 0.03J/K$ and $G = 0.3W/K$. Typical parameters of the power consumption in both modes are $\alpha^{act} = \alpha^{idl} = 0.1W/K$, $\beta^{act} = -11W, \beta^{idl} = -25W$. The ambient temperature, T^0 is assumed to be at room temperature of 300K. In all experiments, we assume that the initial temperature of the system is $T_0 = 330K$.



Task ID	Execution Time (in msec)
1	30
2	140
3	20
4	50
5	50
6	50
7	50

Fig. 1. Task graph and execution times of tasks of an example application



(a) T^t = 375 K (b) T^t = 370 K

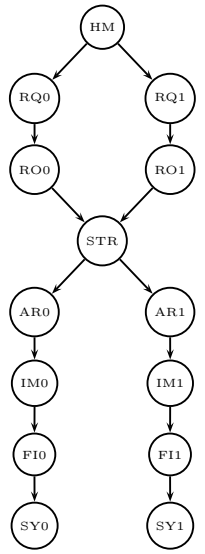
Fig. 2. Temperature variation with time for JUST schedules obtained with different values of T^t. Asterisks denote completion of tasks.

B. Integrated Choice of Static-Ordering and Idle Times for Synthetic Application

Consider the task graph shown in Fig. 1 with given execution time of tasks. The value of the makespan, $\omega = 1.5\tau_{tot} = 0.585s$. We demonstrate finding the optimal schedule for this application. For a target temperature of $T^t = 375$K, we attempt to find the defined sets V_1 and V_2. The corresponding x vector obtained from the BIP is [0 1 0 1 0 1 1]^T, where these values are ordered based on the node numberings. The only ordering of tasks for this x and satisfying the precedence constraints is $\sigma = (1, 3, 5, 2, 4, 6, 7)$. The temperature variation with time for JUST(σ) is shown in Fig. 2. Note the two said phases of the JUST policy: in the first phase $p^{opt} = 3$ tasks are executed with no idle times, and in the next phase, 4 tasks have temperatures at the end of their execution $T_j = T^{opt} = 374$K, which is indeed smaller than $T^t = 375$K.

We now perform a binary search on T^t based on the infeasibility of the BIP. For $T^t = 370$K, the BIP is feasible and the solution is x = [1 0 1 1 1 1 1]^T. One of the different orderings satisfying this x is $\sigma = (2, 1, 3, 5, 4, 6, 7)$. The temperature variation with time for JUST(σ) is shown in Fig. 2. Indeed, the peak temperature is less than 370K. Reducing T^t to 365K, we obtain an infeasible BIP formulation. Hence, we terminate the process. Note that, in all above BIP formulations, the error due to the approximation of (23) is less than 1K.

We execute the task graph periodically with a period of $\omega = 0.585s$, for either choice of σ discussed above. For the two cases, the series of values of $(T^{opt})_n$ are (374 K, 377 K, 378 K, 378.3 K, ...) and (369 K, 377 K, 378 K, 378.3 K, ...).



Task ID	Execution Time ($\times 10^{-7}$s)
HM	236070
RQ0	139325
RQ1	139325
RO0	69385
RO1	69385
STR	73618
AR0	13088
AR1	13088
IM0	711744
IM1	711744
FI0	157184
FI1	157184
SY0	1866138
SY1	1866138

Fig. 3. Task graph representation of a MP3 decoder application and execution times of the tasks on an ARM7TDMI processor

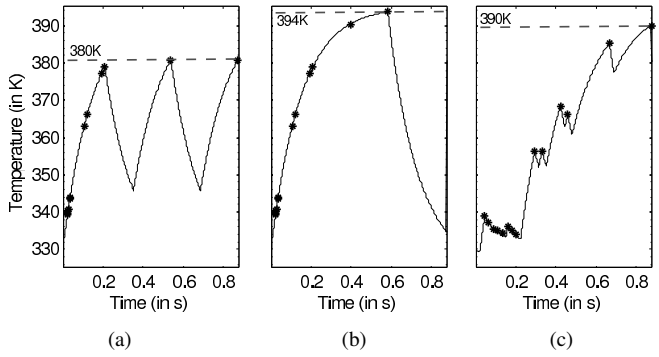


Fig. 4. Temperature variation with time for the MP3 decoder application for a given static-ordering of tasks with scheduling using (a) JUST policy, (b) work-conserving scheduler, and (c) equally distributed idle times.

This experimentally demonstrates Corollary 4. Moreover, it shows that the constant value of peak temperature, $(T^{opt})_\infty$, is reached within a small number of periods.

C. Comparison with naive techniques for a practical application

Consider an MP3 decoder application shown in Fig. 3. The execution times of the tasks on an ARM7TDMI processor are obtained from [9]. Let the maximum allowed makespan $\omega = 1.5 \times \tau_{tot} = 0.874s$.

Let σ be given by the ordering of the tasks in the table in Fig. 3. For this static-ordering of tasks, we compare JUST(σ) with two other naive policies: (a) work conserving scheduling, where idle times before all tasks are 0, (b) equal idle times scheduling, where idle time before each task is equal and makespan equals the bound. The temperature variation with time for each of the naive policies is shown in Fig. 4. Clearly the optimal JUST policy outperforms the others.

VI. Conclusions

Dynamic Thermal Management (DTM) techniques are increasingly becoming necessary in today's electronic systems to reduce on-chip temperatures. One such technique, dynamic voltage scaling (DVS) has been widely studied to manage on-chip temperatures while meeting performance constraints. However, with ever increasing leakage current, it is essential to reduce leakage power using alternate DTM techniques like stop-go scheduling. In this paper, we proposed the use of stop-go scheduling to schedule applications modeled as task-graphs, while satisfying a given makespan constraint minimizing the peak temperature of the system.

Given a static-ordering of the tasks, we proposed the JUst Sufficient Throttling (JUST) schedule, which we proved to be the stop-go schedule with the smallest peak temperature. When the static-ordering is unknown, we proposed a BIP-based approximate formulation to find one. We proved that for periodic task-graphs, where the period equals the maximum allowed makespan, the minimum peak temperature is obtained under a JUST policy. Interestingly, the peak temperature is independent of the chosen static-ordering of tasks, and is thus, free to be chosen based on other constraints. We experimentally validated the presented theoretical results on two sample applications executing on an ARM processor.

Acknowledgments

This work was supported by the PRO3D project financed by the European Community FP7 programme (ref. FP7-ICT-248776).

References

[1] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *FOCS*, 2004.

[2] L. Benini, A. Bogliolo, and G. D. Micheli. Dynamic power management of electronic systems. In *ICCAD*, 1998.

[3] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *DAC*, 2003.

[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, 2001.

[5] T. Chantem, X. S. Hu, and R. P. Dick. Online work maximization under a peak temperature constraint. In *ISLPED*, 2009.

[6] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *ISCA*, 2006.

[7] Y. Liu, R. P. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *DATE*, 2007.

[8] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *TACO*, 1(1), 2004.

[9] S. Stuijk, M. Geilen, and T. Basten. Sdf^3: Sdf for free. http://www.es.ele.tue.nl/sdf3/.

[10] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez. Reducing power in high-performance microprocessors. In *DAC*, 1998.

[11] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. *Real-Time Systems*, 39(1-3), 2008.

[12] C.-Y. Yang, J.-J. Chen, L. Thiele, and T.-W. Kuo. Energy-efficient real-time task scheduling with temperature-dependent leakage. In *DATE*, 2010.

[13] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *ICCAD*, 2007.