Online Work Maximization under a Peak Temperature Constraint

Thidapat Chantem
Department of CSE
University of Notre Dame
Notre Dame, IN 46556
tchantem@nd.edu

X. Sharon Hu
Department of CSE
University of Notre Dame
Notre Dame, IN 46556
shu@nd.edu

Robert P. Dick
Department of EECS
University of Michigan
Ann Arbor, MI 48109
dickrp@eecs.umich.edu

ABSTRACT

Increasing power densities and the high cost of low thermal resistance packages and cooling solutions make it impractical to design processors for worst-case temperature scenarios. As a result, packages and cooling solutions are designed for less than worst-case power densities and dynamic voltage and frequency scaling (DVFS) is used to prevent dangerous on-chip temperatures at run time. Unfortunately, DVFS can cause unpredicted drops in performance (e.g., long response times). We propose and optimally solve the problem of thermally-constrained online work maximization for general-purpose computing systems on uniprocessors with discrete speed levels and non-negligible transition overheads. Simulation results show that our approach completes 47.7% on average and up to 68.0% more cycles than a naïve policy.

Categories and Subject Descriptors

B.8 [**Hardware**]: Performance and Reliability

General Terms

Algorithms, Design, Performance, Theory

1. INTRODUCTION & CONTRIBUTIONS

In response to the increasing computing demands made by applications, system designers have been delivering processors with higher performance at the expense of increasing power densities and temperatures. High chip temperature impacts reliability, performance, cost, and power consumption; microprocessor failure rate depends exponentially upon operating temperature [11]. To handle unsafe temperatures, packages and cooling solutions can be designed to handle worst-case temperature profiles. However, this solution is prohibitively expensive, since the cost of cooling solutions increases super-linearly in power consumption [5].

This work was supported in part by NSF under grant numbers CNS-0834180, CNS07-20457, CCF-0702761, and CNS-0347941 and in part by SRC under grant number 2007-HJ-1593.

ISLPED'09, August 19–21, 2009, San Francisco, California, USA.

Another, less expensive, solution to the temperature problem is to use processor throttling at run time: when the chip temperature exceeds some threshold, the processor power consumption and performance are temporarily reduced by hardware or the operating system. Unfortunately, throttling can cause significant and difficult-to-predict performance loss such as increase in response times.

In this work, we attempt to minimize task response times by maximizing the work completed via an online control policy that requires no prior knowledge of the workload. Dynamic Voltage and Frequency Scaling (DVFS) is used to keep the chip temperature within a temperature constraint. Maximizing the work completed can also be useful in soft real-time systems where the objective is to meet as many deadlines as possible, since it can reduce the number of deadline misses. This claim is substantiated in Section 6.

While there exists work in literature that maximizes work completed online using DVFS under a peak temperature constraint [3,1,4], most solutions assume processors can continuously adjust their speeds. While some authors discuss ways to adapt their solutions for processors with discrete speed levels (albeit without any analysis), only Wang and Bettati placed an emphasis on processors with discrete speed levels [12]. In that work, the processor runs at the highest speed until the threshold temperature is reached. The *equilibrium* speed will then be used to keep the temperature just below its constraint. The equilibrium speed is determined by task power consumption, processor thermal resistance, and temperature constraint and does not necessarily coincide with one of the available speed levels.

To the best of our knowledge, there exists no work that proposes a DVFS control policy for maximizing the work completed for processors with discrete speed levels and non-negligible transition overheads. The Intel chips have two thermal management policies [6]. Once the chip temperature reaches the threshold temperature, the first mechanism (known as *Thermal Monitor 1*), which is also the default mechanism, reduces the duty cycle of the clock (i.e., the proportion of time the clock is active) until the chip temperature drops below the maximum temperature and a timer has expired. The second mechanism (*Thermal Monitor 2*), which is user-configurable, uses throttling to reduce power consumption. While these mechanisms seem reasonable, it is unclear how the user may select the appropriate speed levels and the associated time durations to maximize the amount of work completed. Finally, most existing industry thermal management solutions have operated under the assumption that thermal emergencies are rare events, for which reactive

techniques are sufficient. Now and in the future, due to the cost of high-performance cooling solutions, processors will often operate near their threshold temperatures, requiring proactive techniques to maintain good performance.

In this paper, we tackle the problem of determining speed schedules that maximizes the work completed under a maximum temperature constraint. We propose an optimal DVFS control policy for processors with discrete speed levels and non-negligible transition overheads. Our policy is applicable to any uniprocessor architecture and requires only two speed levels to maximize the work completed. The two speed levels alternate in a periodic manner (Section 3) with some high speed being applied until the chip temperature reaches the threshold temperature. Simulation results show that our DVFS control policy completes 47.7% on average and up to 68.0% more cycles than a naïve policy.

2. PRELIMINARIES

We consider a DVFS-enabled processor with a temperature threshold T_{max}. When the processor temperature reach this threshold, the processor starts throttling, i.e., switching from some high speed to some lower speed to reduce power consumption and performance.

We adopt the lumped RC thermal model similar to that used by Zhang and Chatha [14]. The die temperature above the ambient temperature after t time units is

$$T = \hat{T} + (T_0 - \hat{T}) \cdot e^{-\frac{t}{\tau}}, \quad (1)$$

where $\hat{T}$ is the die's steady-state temperature and $\hat{T} = P_{dyn} \cdot R$, with P_{dyn} being the dynamic power consumption of the die and R its resistance. In addition, τ is the chip time constant, and T_0 is the initial die temperature. Eq. 1 was obtained by solving the following differential equation for T:

$$RC\frac{dT}{dt} + T - RP = 0. \quad (2)$$

Although we will use Eq. 1 for the rest of the paper, all of our derivations in this paper hold for any exponential temperature equation of the same form. For instance, we can replace Eq. 1 with the temperature equation obtained by Rao and Vrudhula where the die and package are modeled separately [10]. We can also extend Eq. 1 to account for leakage power by noting that a piecewise-linear function can be used to estimate leakage power in the operating temperature ranges with roughly 5% error [8]. That is, the modified Eq. 1 can be obtained by solving the following:

$$RC\frac{dT}{dt} + T - R(P_{dyn} + P_{leak}) = 0, \quad (3)$$

where $P_{leak} = \alpha T + \beta$ for some constants α and β [8].

For each speed level k of the processor, we define an associated tuple (V_k, S_k, P_k), where V_k, S_k, and P_k are the required voltage, speed, and power consumption of the processor when it executes at speed level k, respectively. For speed level k, Eq. 1 can be written as

$$T = \hat{T}(S_k) + (T_0 - \hat{T}(S_k)) \cdot e^{-\frac{t}{\tau}}, \quad (4)$$

where $\hat{T}(S_k)$ is the steady-state temperature when the processor executes at speed level k and $\hat{T}(S_k) = S_k^3 P_k R$.

Problem 1: Given a processor that is kept busy with work to be completed, determine a speed schedule such that the peak temperature constraint is met and total work completed is maximized.

3. POLICY FOR PROCESSORS WITH NEGLIGIBLE TRANSITION OVERHEADS

We describe a policy for maximizing the work completed over a schedule length based on some crucial observations. Namely, we determine (i) the speed levels needed, (ii) the length of time the processor should spend in each speed level, and (iii) the temporal sequence of speed levels at which the processor should execute.

For now, we assume that the processors under consideration have negligible speed transition overheads (Section 4 will generalize). This simplifying assumption allows us to identify some important characteristics of our policy.

Our objective is to develop an optimal DVFS control policy for use once the chip reaches its threshold temperature, and not a pre-throttling policy. In many systems, the time from startup to reaching the temperature constraint is a negligible percentage of total time. In addition, it is important to note that our DVFS control policy does not perform task scheduling, though it can be used in conjunction with any existing task scheduling algorithm.

Consider a high speed level S_H where $\hat{T}(S_H) \geq T_{max}$. During throttling, if the processor execute tasks using S_H for long enough, the chip peak temperature will eventually reach T_{max}. Our first question is whether such a high speed should be used until the chip temperature reaches T_{max} or be used for a shorter amount of time. The following lemma answers this question. In addition, the "sufficiently large time interval" requirement is there to ensure that the time interval under consideration is long enough for the chip temperature to reach T_{max} at least once.

LEMMA 1. *Given a sufficiently large time interval $[t_a, t_b]$, consider the speed schedules that consecutively apply S_{L1}, S_H, and S_{L2} where $\hat{T}(S_{L1}) < T_{max}$, $\hat{T}(S_{L2}) < T_{max}$ and $\hat{T}(S_H) \geq T_{max}$. In addition, S_{L1} and S_{L2} may be identical but need not be. Let the transition overheads be negligible. Given some initial temperature $T_a \geq \min\{\hat{T}(S_{L1}), \hat{T}(S_{L2})\}$ and end temperature T_b. A schedule that completes the maximum amount of work must allow the chip temperature to reach T_{max} at the end of the application of S_H.*

We give a sketch of the proof (details can be found in our technical report [2]). The work completed is a function of $S_H, S_{L1}, S_{L2}, t_a, t_b$, as well as t_1 and t_2, which are the durations of the application of S_{L1} and S_H, respectively. Accordingly, we can write three temperature equations for interval $[t_a, t_b]$. Solving the system of equations, we express the work completed as a function of T_2, the end temperature of the high speed execution. By differentiating the total work completed with respect to T_2 and setting the resulting function to 0, we can solve for T_2 and show that to maximize the work completed, $T_2 = T_{max}$.

In Lemma 1, we referred to the high (S_H) and low (S_{L1} and S_{L2}) speeds with the requirements that $\hat{T}(S_{L1}), \hat{T}(S_{L2}) < T_{max}$ and $\hat{T}(S_H) \geq T_{max}$. Since modern processors often have several speed levels, we use the following theorem to determine which maximize the work completed.

THEOREM 1. *Given a sufficiently large time interval $[t_a, t_b]$, consider the speed schedules that consecutively apply S_{L1}, S_H, and S_{L2} an arbitrary number of times and $\hat{T}(S_{L1}) < T_{max}$, $\hat{T}(S_{L2}) < T_{max}$, and $\hat{T}(S_H) \geq T_{max}$. Let the transition overheads be negligible. Given some initial temperature*

$T_a \geq \min\{\hat{T}(S_{L1}), \hat{T}(S_{L2})\}$ *and end temperature* T_b *and assume that the duration of the application of* S_{L1} *or* S_{L2} *is non-zero, a schedule that completes the maximum amount of work must satisfy* $S_{L1} = S_{L2} = \max\{s|\hat{T}(s) < T_{max}\}$ *and* $S_H = \min\{s|\hat{T}(s) \geq T_{max}\}$.

PROOF. We first prove the theorem for the schedules that consecutively apply S_{L1}, S_H, and S_{L2} once. Let t_1 and t_2 denote the durations of the application of S_{L1} and S_{L2}, respectively. Let t_3 denotes the duration of the application of S_H and $t_3 = t_b - t_a - t_1 - t_2$. In addition, let T_1 and T_3 be the temperatures at the end of the application of S_{L1} and S_H, respectively. We can write three associated temperature equations as follows:

$$T_1 = \hat{T}(S_{L1}) + (T_a - \hat{T}(S_{L1}))e^{-\frac{t_1}{\tau}}, \tag{5}$$

$$T_3 = \hat{T}(S_H) + (T(t_1) - \hat{T}(S_H))e^{-\frac{t_b - t_a - t_1 - t_2}{\tau}}, \text{ and} \tag{6}$$

$$T_b = \hat{T}(S_{L2}) + (T(t_3) - \hat{T}(S_{L2}))e^{-\frac{t_2}{\tau}}. \tag{7}$$

From Lemma 1, we know that $T_3 = T_{max}$. Letting $A = e^{-\frac{t_b - t_a - t_2}{\tau}}$ and combining Eq. 5 with Eq. 6 yields

$$t_1 = \tau \ln\left(\frac{\hat{T}(S_H) - T_{max} + (T_a - \hat{T}(S_{L1}))A}{(\hat{T}(S_H) - \hat{T}(S_{L1}))A}\right). \tag{8}$$

By definition, the total work completed during $[t_a, t_b]$ is $W = (S_{L1} - S_H) \cdot t_1 + (S_{L2} - S_H) \cdot t_2 + S_H \cdot (t_b - t_a)$. To determine the appropriate value of S_{L1} to maximize W, we take the partial derivative of W with respect to S_{L1}, observing that neither A nor t_2 depends on S_{L1}. We obtain:

$$\frac{\partial W}{\partial S_{L1}} = \tau \ln\left(\frac{\hat{T}(S_H) - T_{max} + (T_a - \hat{T}(S_{L1}))A}{(\hat{T}(S_H) - \hat{T}(S_{L1}))A}\right) + \frac{\tau(S_{L1} - S_H)\hat{T}'(S_{L1})}{\hat{T}(S_H) - \hat{T}(S_{L1})} \cdot \left(\frac{\hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_H))}{\hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_{L1}))}\right). \tag{9}$$

From Eq. 9, $\frac{\partial W}{\partial S_{L1}} = 0$ if $T_a = A^{-1}(T_{max} - \hat{T}(S_H) + \hat{T}(S_H)A)$. However, $T_{max} < \hat{T}(S_H) - A(\hat{T}(S_H) - T_a)$ and therefore $T_a > A^{-1}(T_{max} - \hat{T}(S_H) + \hat{T}(S_H)A)$. (This can be proved by combining Eq. 5 with Eq. 6, writing the resulting equation as a function of T_{max}, and directly comparing T_{max} to $\hat{T}(S_H) - A(\hat{T}(S_H) - T_a)$). Now, we want to determine how $\frac{\partial W}{\partial S_{L1}}$ changes as T_a increases. To do so, we take the second partial derivative of W with respect to T_a. We have

$$\frac{\partial^2 W}{\partial T_a \partial S_{L1}} = \tau \frac{A}{\hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_{L1}))} \cdot \left(1 + \frac{A(S_{L1} - S_H)\hat{T}'(S_{L1})}{\hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_{L1}))}\right). \tag{10}$$

Since $\hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_H)) > 0$, $\hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_{L1})) > 0$. Now, $\frac{\partial^2 W}{\partial T_a \partial S_{L1}} \geq 0$ if the expression inside the main parentheses of Eq. 10 is greater than 0. In other words, we wish to show that

$$\frac{A(S_{L1} - S_H)\hat{T}'(S_{L1})}{\hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_{L1}))} \geq -1. \tag{11}$$

Since $\hat{T}(S_{L1}) < \frac{\hat{T}(S_H) - \hat{T}(S_{L1})}{S_H - S_{L1}}$, we have

$$A(\hat{T}(S_H) - \hat{T}(S_{L1})) \leq \hat{T}(S_H) - T_{max} + A(T_a - \hat{T}(S_{L1}))$$
$$T_{max} \leq \hat{T}(S_H) - A(\hat{T}(S_H) - T_a), \tag{12}$$

which holds, as was explained between Eq. 9 and Eq. 10. Therefore, $\frac{\partial^2 W}{\partial T_a \partial S_{L1}} \geq 0$ and as T_a increases, $\frac{\partial W}{\partial S_{L1}}$ also increases. In other words, $\frac{\partial W}{\partial S_{L1}} \geq 0$ for all valid values of T_a and $S_{L1} = \max\{s|\hat{T}(s) < T_{max}\}$.

The same technique can be used to prove that $S_H = \min\{s|\hat{T}(s) \geq T_{max}\}$ and $S_{L2} = \max\{s|\hat{T}(s) < T_{max}\}$. Furthermore, using induction, we can prove the theorem for the schedules that consecutively apply S_{L1}, S_H, and S_{L2} an arbitrary number of times. □

As a direct consequence of Theorem 1, a DVFS control policy that maximizes the work completed only needs to use two speed levels: $S_H = \min\{s|\hat{T} \geq T_{max}\}$ and $S_L = \max\{s|\hat{T} < T_{max}\}$. Incorporating these results, the following theorem generalizes the observation from Lemma 1 to an arbitrary number of high speed intervals. For the rest of the section, the proofs can be found in our technical report [2]).

THEOREM 2. *Given a sufficiently large time interval* $[t_a, t_b]$, S_L *and* S_H *be alternately applied and* $\hat{T}(S_L) < T_{max}$ *and* $\hat{T}(S_H) \geq T_{max}$. *Let the transition overheads be negligible. Given an initial temperature* $T_a = \hat{T}(S_L)$ *and end temperature* T_b, *and let the total number of speed transitions be fixed, a schedule that completes the maximum amount of work must allow the chip temperature to reach* T_{max} *at the end of every application of* S_H.

Theorems 1 and 2 provide a theoretical foundation for any work-maximizing, DVFS control policy. That is, the theorems specify the speed levels needed and indicate that it is advantageous in terms of maximizing the work completed to alternate between the high and low speeds while allowing the chip to reach the maximum temperature at the end of every high speed application interval. We now need to determine how long the low speed level should be applied and whether each low speed level interval should have the same duration. To answer these questions, we begin by defining a periodic speed schedule then showing that such a schedule is part of the optimal DVFS control policy.

Definition 1. A periodic speed schedule is a speed schedule that alternately applies S_L and S_H (where $\hat{T}(S_L) < T_{max}$ and $\hat{T}(S_H) \geq T_{max}$) in a time interval such that the durations of all applications of S_H are the same and the durations of all applications of S_L are the same.

LEMMA 2. *Given a sufficiently large time interval* $[t_a, t_b]$ *with the initial chip temperature of* T_{max}, *let* S_L *and* S_H *be alternately applied with* S_H *being applied until the chip temperature reaches* T_{max} *and* $\hat{T}(S_L) < T_{max}$ *and* $\hat{T}(S_H) \geq T_{max}$. *Let the transition overheads be negligible and let the total number of speed transitions be fixed. A schedule that completes the maximum amount of work must be a periodic speed schedule.*

Though Lemma 2 describes a desired property of a work-maximizing speed schedule, it does not specify the length of the S_L intervals. The time duration in which the processor applies S_L determines the number of speed transitions in a given time interval. For processors with negligible transition overheads, more transitions would lead to more work completed (i.e., the duration of every application of S_L should be minimized), as shown by the following theorem.

THEOREM 3. *Given a sufficiently large time interval $[t_a, t_b]$, Let S_L and S_H be alternately applied and satisfying $\hat{T}(S_L) < T_{max}$ and $\hat{T}(S_H) \geq T_{max}$. Let the transition overheads be negligible. A schedule with m speed transitions completes more work than a schedule with n speed transitions if $m > n$.*

We now summarize our **optimal DVFS control policy**. To maximize the work completed, the processor should periodically alternate between the low speed $S_L = \max\{s|\hat{T} < T_{max}\}$ and the high speed $S_H = \min\{s|\hat{T} \geq T_{max}\}$. In addition, the processor should run at the high speed S_H until the chip temperature reaches T_{max}. With negligible transition overhead, the processor should minimize the time it spends running at the low speed (i.e., the throttling time) by switching to the high speed as soon as possible.

4. EXTENDING THE OPTIMAL POLICY TO NON-IDEAL CASES

Each speed transition imposes some overhead, reducing the amount of time spent on computation. Figure 1 illustrates the typical trajectories for voltage and speed levels during two transitions. When transitioning from a lower speed to a higher speed, the voltage is gradually increased until it reaches the required value (we have simplified the voltage curve to a straight line when in reality it is a staircase curve). Once this happens, the processor switches to the higher speed. During this transition, there is a small time interval α during which the processor clock is halted and no work is completed. The process of transitioning from a higher speed to a lower speed is similar, except that the processor switches to the new speed immediately and gradually decreases the voltage. Once again, the processor clock is halted for a short duration β. Typical values for α and β are on the order of tens of microseconds. The voltage changing times, a and b, are on the order of hundreds of microseconds.

Compared to the scenario where there is no transition overhead (denoted as "ideal" in Figure 1), there is no work loss during b. During α and β, the number of cycles lost is $S_H \cdot \alpha$ and $S_L \cdot \beta$, respectively. Finally, during a, the number of cycles lost is $(S_H - S_L) \cdot a$. To find the optimal value of the time the processor spends at the low speed level t_l, we find the maximum value of the net work completed function that accounts for transition overheads.

Given a schedule length L, the net work completed W^* is

$$W^* = [(t_l - \beta + a) \cdot S_L + (t_h - \alpha - a) \cdot S_H] \cdot \frac{L}{t_l + t_h}, \quad (13)$$

where t_h is the duration of the high speed level application.

The following theorem identifies the optimal value for t_l.

THEOREM 4. *Given a schedule length L, let S_L and S_H be two speed levels satisfying $\hat{T}(S_L) < T_{max}$ and $\hat{T}(S_H) \geq T_{max}$ Let t_l and t_h be the time durations the processor spends at the low and high speeds, respectively, and let $\lambda = \beta \cdot S_L + \alpha \cdot S_H + (S_H - S_L) \cdot a$, where α, β, and a are constants associated with transition overheads as defined previously. Further, assume that the processor uses the DVFS control policy presented in Section 3. A speed schedule that maximizes the net work completed over L must have t_l^* that satisfies*

$$(S_H - S_L) \cdot (t_h - t_l^* \cdot t_h') - \lambda \cdot (1 + t_h') = 0, \quad (14)$$

where t_h is expressed as a function of t_l^* and $t_h' = \frac{\partial t_h}{\partial t_l^*}$.

As a result, we can use our previously proposed policy to maximize the work completed for processors with non-negligible transition overheads if the throttling time is found by solving Eq. 14 using a nonlinear equation solver.

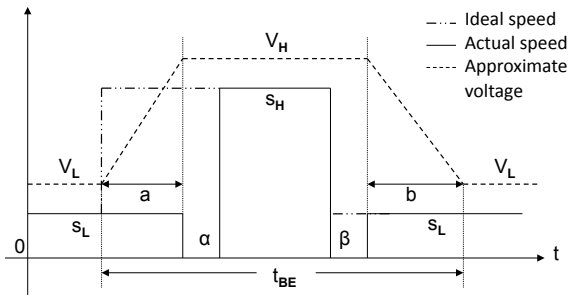


Figure 1: Waveforms of speed S and voltage V levels for two transitions.

5. WORKLOADS WITH DIFFERENT POWER CONSUMPTIONS

So far, we have assumed that processor power consumption is fixed over time for a given speed level. In reality, the required power consumption may depend on the details of operation and hardware, resulting in steady-state temperatures differing among workloads even for the same processor speed level. Since our DVFS control policy relies on the steady-state temperature of different speed levels to determine S_H and S_L, some modifications are needed. Consider a system that must execute different workloads with different power consumption over time. For the duration of a workload, which may consist of a number of applications, we can determine S_H and S_L as follows: $S_H = \min\{s|s^3 P_i R \geq T_{max}\}$ and $S_L = \max\{s|s^3 P_i R < T_{max}\}$, where P_i is the maximum power consumption when executing workload WL_i. As long as the power consumption of the workload in a given time interval is known at run-time, e.g., by using performance counter based power models [7], our proposed DVFS control policy will maximize the work completed. The more complex case where tasks in a workload require different power consumptions is left as future work.

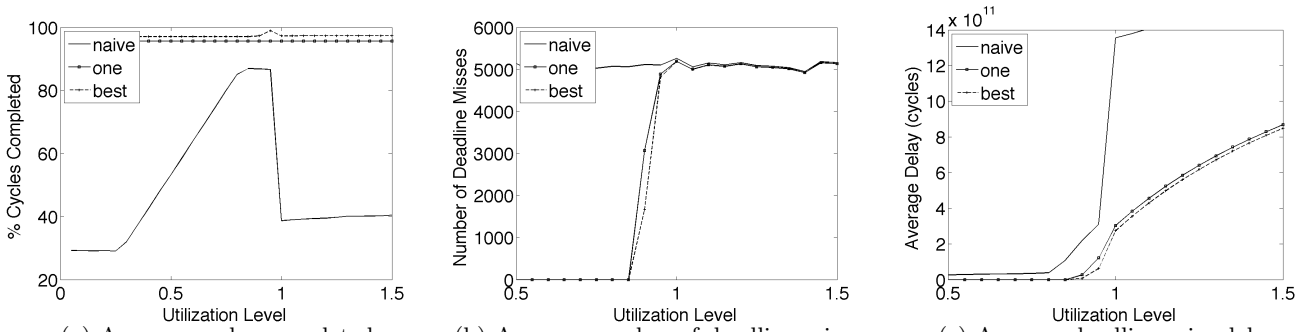
6. SIMULATION RESULTS

In this section, we use simulation results to demonstrate the effectiveness of our optimal DVFS control policy.

6.1 Simulation Setup

Using a Java simulator, we modeled our processor based on the Alpha 21264 processor, which consumes 120 W of power when running at the highest frequency of 4 GHz with the maximum temperature of 110 °C [10]. The silicon die and copper package have the dimensions of 16 mm × 16 mm × 0.5 mm and 24 mm × 24 mm × 2 mm, respectively. The threshold temperature is set to 90 °C. To compute the time constant for Eq. 1, we obtained temperature data via simulations in ISAC [13] using the default settings for all thermal-related parameters (e.g., heat capacity).

Since we did not have the data on the available voltage and frequency pairs of the Alpha processor, we assumed that it can switch to the same speed levels as the Intel Core Duo [6]. That is, we used the speed levels of the Intel Core Duo but calculated the corresponding power consumption and frequency. For our system, the available speed levels are: 0.462, 0.615, 0.692, 0.769, 0.846, 0.923, and 1.



(a) Average cycles completed (b) Average number of deadline misses (c) Average deadline miss delay

Figure 2: Results for different speed selection policies with throttling time of 10 s.

As our policy does not perform scheduling, it is insensitive to the type of applications that may be running. We used a periodic soft real-time system as an example application. Each simulation consisted of 100 randomly generated task sets of 20 tasks each for 30 different utilization levels ($U_{level} = 0.05, 0.1, \cdots, 1.5$), for a total of 3,000 task sets. The utilization level signifies how loaded the processor is; a utilization level of 1 or greater means that the system is overloaded. Task periods ranged from 1 s to 10 s, with task execution times from 0.2 to 0.8 of the periods. For underloaded systems, a background job was also added. Each task set was simulated for a duration of 1,000 s.

For each run, we recorded the following data: number of cycles completed, number of deadline missed, average delays for jobs that missed their deadlines, and associated transition overhead, if applicable. Note that while we use a soft real-time system as an example here, metrics such as the number of cycles completed are relevant for general-purpose computing systems as well. In addition, while maximizing the work completed is not the same as maximizing the number of deadlines met, our results show that completing more work often leads to meeting more job deadlines.

6.2 Negligible Transition Overheads Case

We now discuss the simulation results for the different speed selection policies: (i) the naïve approach where the highest and lowest speed levels are both used (corresponding to the speeds of 1 and 0.462, respectively), (ii) the one speed approach where the highest of the low speed levels is used the entire time (corresponding to the speed of 0.846), and (iii) the best approach where the lowest of the high and highest of the low speed levels are selected (corresponding to the speeds of 0.923 and 0.846, respectively). Note here that the naïve approach could represent the policy used by *Thermal Monitor 2* described in Section 2. We did not specifically compare with *Thermal Monitor 1* since it is non-configurable. The initial chip temperature is set to T_{max} since we are interested in the performance of the system once the processor starts to throttle. The throttling time used for this set of simulations is 10 s, which is similar to the default throttling time for the 2.13 GHz Pentium M-770 CPU [9].

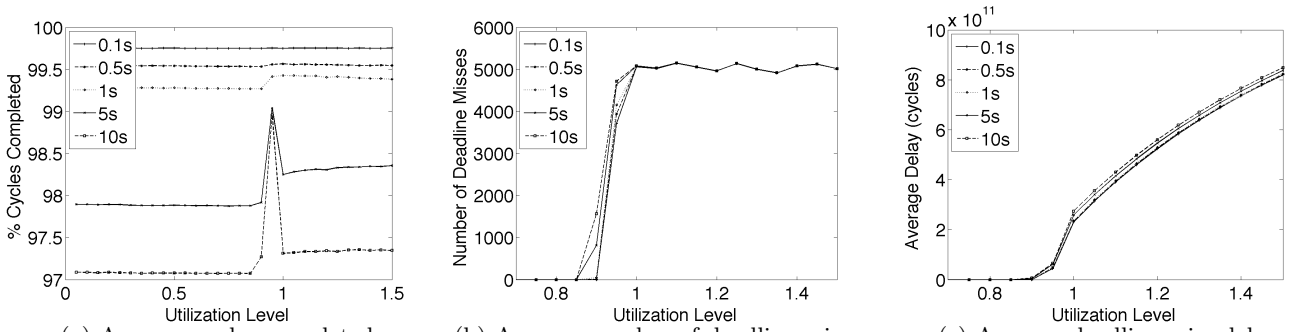
The results are shown in Figure 2(a)–(c), which compare the average number of cycles completed, average number of deadline misses, and average delays for jobs that missed their deadlines, respectively. The average number of cycles completed is compared with the number of cycles completed when using the equilibrium speed described in Section 2. We can see that the proposed speed selection policy consistently outperforms the naïve policy in terms of all performance metrics. A comparison between the naïve and best speed selection policies reveals that our approach completes 47.65% on average and up to 67.99% more cycles than the naïve approach. Our policy also improves the number of cycles completed by the one speed policy by 1.60% on average and up to 3.29%. When compared to the number of cycles completed using the equilibrium speed, our approach deviates on average by only 2.76% and up to 2.93%. In addition, our policy reduces deadline misses by 59.38% on average and up to 100% compared to the naïve policy. Our policy also reduces deadline misses compared to the one speed approach by 3.65% on average and up to 45.74%. Note that the best policy would yield more substantial performance improvements over the one-speed policy for systems with fewer speed levels and/or when the lowest of the high speed differs significantly from the equilibrium speed.

Figure 3(d)–(f) show the effect of different throttling times (i.e., the times the processor spends at the low speed) on system performance when using our policy. We used the throttling times of 0.1 s, 0.5 s, 1 s, 5 s, and 10 s. Recall that the 2.13 GHz Pentium M-770 CPU uses a throttling time of about 10 s [9]. With a throttling time of 0.1 s, our DVFS control policy completes between 0.20% and 2.51% on average and up to between 0.22% and 2.68% more cycles than the throttling times of 0.5 s, 1 s, 5 s, and 10 s, respectively. When compared to the policy that uses the equilibrium speed, our policy completes 0.25% fewer cycles (using the throttling time of 0.1 s). Last but not least, the smallest throttling time (0.1 s) reduces deadline misses by between 8.14% and 9.42% on average and up to 100%.

6.3 Non-Negligible Transition Overheads Case

As described in Section 4, the constants associated with transition overheads that we need to consider are α, β, and a, which we set to 10 us, 5 us, and 100 us, respectively. Since these constants are much smaller than the die thermal time constants, it is reasonable to assume that the die temperature does not change during speed transitions.

Figure 4 plots the net number of cycles completed (i.e., with transition overheads considered) as a function of the throttling times using our DVFS control policy. By solving Eq. 14, we know that the optimal throttling time is 43.23 ms and this is confirmed by the results of the simulation. As expected, when transition overheads are non-negligible, switching from the low to high speeds more often



(a) Average cycles completed (b) Average number of deadline misses (c) Average deadline miss delay

Figure 3: Results for different throttling times for best speed selection policy.

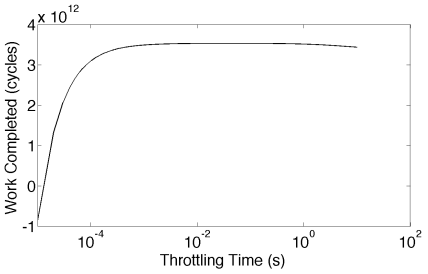


Figure 4: Average number of cycles completed as a function of throttling times when transition overheads are considered.

than optimal decreases the rate of computation due to the increasing proportion of time spend idle during transitions. On the other hand, switching very infrequently can reduce the computation rate because a lower percentage of time can be spent at the higher speed.

7. CONCLUSIONS & FUTURE WORK

We proposed an optimal online DVFS control policy to maximize instruction cycles subject to a peak temperature constraint. Our solution is applicable to any processor with discrete speed levels and non-negligible transition overheads. Our policy completed 47.7% on average and up to 68.0% more cycles when compared to the naïve policy.

We plan on extending our proposed policy to consider the time interval before the system reaches the threshold temperature for the first time. To handle workloads where tasks have different power consumption, modifications to the proposed policy are needed. Finally, we would like to extend our policy to consider multiprocessor architectures.

8. REFERENCES

[1] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. of the ACM*, 54(1):1–39, Mar. 2007.

[2] T. Chantem, X. S. Hu, and R. P. Dick. Online work maximization under a peak temperature constraint. Technical report, University of Notre Dame, June 2009.

[3] J.-J. Chen, C.-M. Hung, and T.-W. Kuo. On the minimization of the instantaneous temperature for periodic real-time tasks. In *Proc. Real-Time and Embedded Technology and Applications Symp.*, pages 236–248, Apr. 2007.

[4] J.-J. Chen, S. Wang, and L. Thiele. Proactive speed scheduling for real-time tasks under thermal constraints. In *Proc. Real-Time and Embedded Technology and Applications Symp.*, pages 141–150, Apr. 2009.

[5] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, 5(1):1–9, Feb. 2001.

[6] Intel Core Duo processor and Intel Core Solo processor on 65 nm process datasheet. http://www.intel.com/design/mobile/datashts/309221.htm.

[7] T. Karkhanis and J. E. Smith. Automated design of application specific superscalar processors: An analytical approach. In *Proc. Int. Symp. Computer Architecture*, pages 402–411, 2007.

[8] Y. Liu, R. P. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *Proc. Design, Automation & Test in Europe Conf.*, pages 204–209, Mar. 2007.

[9] A. Mallik, J. Cosgrove, R. P. Dick, G. Memik, and P. Dinda. PICSEL: Measuring user-perceived performance to control dynamic frequency scaling. In *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems*, pages 70–79, Mar. 2008.

[10] R. Rao and S. Vrudhula. Performance optimal processor throttling under thermal constraints. In *Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 257–266, Oct. 2007.

[11] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *Proc. Int. Symp. Computer Architecture*, pages 520–531, June 2005.

[12] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. In *Proc. Euromicro Conf. Real-Time Systems*, pages 161–170, July 2006.

[13] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang. ISAC: Integrated space and time adaptive chip-package thermal analysis. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pages 86–99, Jan. 2007.

[14] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *Proc. Int. Conf. Computer-Aided Design*, pages 281–288, Nov. 2007.