Temperature-aware DVFS for Hard Real-time Applications on Multi-core Processors

Vinay Hanumaiah, Student Member, IEEE and Sarma Vrudhula, Senior Member, IEEE

Abstract—This paper addresses the problem of determining the feasible speeds and voltages of multi-core processors with hard real-time and temperature constraints. This is an important problem, which has applications in time-critical execution of programs like audio and video encoding on application-specific embedded processors. Two problems are solved. The first is the computation of the optimal time-varying voltages and speeds of each core in a heterogeneous multi-core processor, that minimize the makespan—the latest completion time of all tasks, while satisfying timing and temperature constraints. The solution to the makespan minimization problem is then extended to the problem of determining the feasible speeds and voltages that satisfy task deadlines. The methods presented in this paper also provide a theoretical basis and analytical relations between speed, voltage, power and temperature, which provide greater insight into the early-phase design of processors and are also useful for online dynamic thermal management.

Index Terms—Multi-core, performance optimization, real-time, task deadlines, dynamic voltage and frequency scaling, thermal management, makespan minimization, leakage dependence on temperature, optimal control.

I. INTRODUCTION

Today, every major manufacturer of high performance microprocessors has shifted to multi-core processors as a solution to the problem of soaring power consumption of single core processors. As this trend continues, and processors with many hundreds of cores become available in a few years, the industry will once again face the problem of soaring power dissipation. In contrast to single core processors, multi-core processors will require far more sophisticated techniques to control the heat generated and maximize the performance.

As the die temperatures of the multi-core processors continue to rise, the packaging and the cooling solutions designed to accommodate the worst-case temperature gets prohibitively expensive [1]. Hence, for processors with tens of cores or higher, the package can only be designed to dissipate the average power dissipation. This necessitates the use of dynamic thermal management (DTM) techniques such as dynamic voltage and frequency scaling (DVFS), and task migration among the cores, to ensure that the peak temperature is maintained below the maximum specified limit.

In general, DTM is a complex, multi-dimensional, constrained optimization problem, with objective functions, control variables, and constraints being determined by the target market. In the case of a general purpose computer, a typical objective is to maximize throughput or minimize makespan of a set of tasks. For embedded or application specific systems, the objective would be to complete the maximum number of tasks, constrained by the task deadlines. The control variables are the frequency and/or the voltage of each core, and the allocation of tasks to cores over time. The global physical constraint would be to ensure that the die temperature never exceeds a specified upper bound. The die temperature is a time-varying function of task and physical parameters and has a circular dependence on the power consumption, due principally to the dependence of leakage power on temperature. Adding to these challenges will be the need to employ DTM much more frequently for multi-core processors and for it to be fast enough to be practically implementable.

A. Scope of this Paper

This paper presents a formulation and a solution to the problem of controlling the voltages and speeds of cores in a heterogeneous multi-core processor to guarantee execution of thermally constrained tasks with hard deadlines. A thermally constrained task is one which if executed on a core at the maximum possible speed, will result in the temperature of that core exceeding a given safe temperature limit. The formulation includes well established and accurate power and thermal models [2], [3], and accounts for (1) the differences in power consumption of tasks and cores, (2) the differences in the thermal characteristics of the functional blocks in the die and the various components of the package, and (3) the dependence of leakage power on temperature.

The solution to this problem is complicated by the fact that there is a circular dependency between the leakage power and the temperature, and the constraints on task deadlines. The leakage power dependence on temperature (LDT) is simplified by using a piece-wise linear approximation. In order to accommodate task deadlines, the problem is solved in two steps. In the first step, the deadline constraints are ignored and the optimal solution for minimizing the makespan is derived. In the second step, this solution is modified to satisfy the boundary conditions imposed by the task deadlines. The analytical results for the makespan minimization without task deadlines presented herein are useful both for online DVFS and also for early-phase design space exploration. The time to compute the deadline feasible solution increases...
loglinearly with the number of tasks. The proposed techniques for makespan minimization and deadline feasible solution are demonstrated by simulating benchmark programs on a multicore processor consisting of replicas of the Alpha 21264 core. Fig. 1 shows the organization of the rest of this paper.

B. Comparison with Existing Work

A significant body of work has been published on power and thermal-aware DVFS and task scheduling techniques for single and multi-core processors. Table I summarizes some of the recently published research that is most relevant to the work presented in this paper.

Until recently, most of the work on DTM has focused mainly on single-core processors [4]–[11]. The corresponding optimization problems in multicore processors are more challenging due to additional control variables such as voltages and speeds of each core, and the dynamic allocation of tasks to cores (task migration). Multicore DTM strategies such as those in [12]–[14] address power minimization, and such schemes may not be optimal under thermal constraints.

The difficulties raised by the inclusion of thermal constraints have led researchers to simplified formulations of the optimization problems. These include 1) only considering single-core processors [8], [10], [11], 2) examining only steady-state solutions without considering the transient behavior [7], [15], [16], 3) the use of lumped RC thermal models [3]–[5], [17], [18], 4) ignoring LDT [5], [10], [11], [19], [20], 5) ignoring voltage scaling [19]–[21] (voltage scaling provides cubic power reduction), and 6) ignoring task deadlines [10], [19], [22]. The aim of this work is to develop a more general optimization framework that removes the above limitations of existing work.

II. System Models

A. Task Model

Consider an n-core processor. Task \( t_{d,i} \) has a deadline \( t_{d,i} \) and runs on core \( i \). For notational convenience, it is assumed that \( t_{d,i} \leq t_{d,i+1} \). The speed \( s_i \) and the voltage \( v_i \) of core \( i \) are continuous functions of time, and normalized over [0,1].

Every core is assumed to execute a single task at a time (i.e. no simultaneous multi-threading) and the communication between tasks is neglected. The last two assumptions are certainly valid for a (large) class of stream processors, where each core is designed to be simple enough to run one task at a time and requires every thread to have independent execution [36]. Each task is assumed to run for at least the die thermal time constant (few milliseconds), as tasks durations shorter than that do not have a significant thermal impact, and hence would not benefit from the proposed optimization.

B. Power and Thermal models

The well established thermal model HotSpot [2] is used here. HotSpot uses an electro-thermal analogy, where power sources are modeled as current sources, and heat spreading and storage capacity are modeled through resistors and capacitors respectively. The power models are adopted from [3]. Each core is divided into \( m \) functional blocks on the die and the thermal interface material (TIM) layers. The spreader and the heat sink layers below the TIM have 5 and 9 blocks respectively. Together the total number of thermal blocks is \( N = 2nm + 14 \). For a four core processor with 20 blocks per core, the total number of blocks is 174.

Owing to the large number of parameters, this high order system presents significant difficulties for further analysis. To simplify the model, we examined the Alpha 21264 RC thermal model, and made several observations that allowed us to make useful and practical simplifications [9], [15]. The observations and the corresponding simplifications are as follows.

1) The thermal resistance and the capacitance of the cooling system are much larger than that of other components of the package. Since the thermal constraint is specified for the entire chip, and not for any specific component of the package, the package can be lumped into a single node.

2) The lateral resistance between the functional blocks on the die and the TIM blocks are four times higher than the corresponding vertical resistances. Thus most of the heat spreads through vertical resistances and hence lateral resistances can be ignored. Note that this does not eliminate thermal hotspots as they are created due...
to differences in the power densities among functional blocks.

3) The thermal time constant of the package is three orders of magnitude higher than the thermal time constant of the die. The representative tasks considered in this paper have execution times that are much larger than the die thermal time constant, and are comparable to the package thermal time constant. Consequently, this saturates the die thermal capacitances, and hence these capacitances can be ignored.

It is to be noted that these assumptions are critical in deriving an analytical equation that allows us to estimate the maximum number of instructions that can be executed in a given interval. As it will be shown later in Section V-A this estimation of the number of instructions is required to determine the optimal core speeds and voltages that satisfy the task deadlines. Moreover, it was shown in [37] that the temperature predictions with these assumptions result in an error less than 6%. Hence these assumptions do not result in significant loss in accuracy of the solution.

Fig. 2 shows the high-level thermal model based on the above simplifications. The power of a functional block within core $i$ is modeled with a current source $P_{i,b}$, which is the sum of dynamic power $P_{d,i,b}$ and leakage power $P_{l,i,b}$. The dynamic power varies linearly with speed $s_i$ and quadratically with voltage $v_i$, while the leakage power depends exponentially on the temperature $T_{i,b}$ and the voltage $v_i$ [3]. The package resistance and capacitance are denoted by $R_p$ and $C_p$ respectively. The resistances connecting the functional blocks in the die to the package are represented by $R_{i,j}$, which is the sum of the vertical resistances in the die and the TIM.

C. Voltage-speed-temperature Relationship

It is known that mobility degradation with temperature affects the selection of speed and voltage, which are chosen such that they satisfy the delay constraints of circuits. The empirical relationship relating a core’s maximum speed, its voltage and its peak temperature in 65 nm technology, is given by [3]

$$s_{i,max}(t) = k_s^i \left( \frac{v_i(t) - v_{th}}{v_i(t) \max(T_{i,b}(t))^{\gamma_i}} \right)^{1.2},$$  

(1)

$k_s^i$ is the constant of proportionality and $v_{th}$ is the threshold voltage. The $\max$ in (1) is taken over all the blocks of core $i$.

III. PROBLEM FORMULATION

A. Problem Statement

The notations used in the paper are summarized in Table II. Given an $n$-core processor executing $n$ tasks, let $N_{tot,i}$ be the number of instructions to be executed and $IPC_i$ be the corresponding instruction per cycle of task $i$. Let $N_i(t)$ denotes the number of instructions of task $i$ completed by time $t$. The problem is to determine the speeds and voltages of cores such that all tasks complete their execution within their deadlines $d_{i}$, while satisfying the thermal constraints.

Further, among the set of deadline feasible solutions, a solution with the minimum makespan is sought. Note that this is a natural extension to the problem under consideration as cores need to execute as fast as possible in order to satisfy the deadlines, especially when constrained by a maximum temperature. This problem is formulated as follows.

$$\min_{s(t), v(t)} \quad t_f = \max_{1 \leq i \leq n} t_{f,i},$$  

(2)

s.t.  

$$\frac{dN_i(t)}{dt} = IPC_i(t)s_i(t), \quad \forall i, t,$$  

(3)

$$N_i(0) = 0, \quad N_i(t_{d,i}) = N_{tot,i}, \quad \forall i,$$  

(4)

$$T_{i,b}(t) = f(P(T_i(t), s(t), v(t))), \quad \forall i, t,$$  

(5)

$$T_{i,b}(0) = T_{i0}, \quad \forall i, b,$$  

(6)

$$T_{i,h}(t) = \max(T_{i,b}(t)) \leq T_{max}, \quad \forall i, t,$$  

(7)

$$s_i(t) \leq k_s^i \left( \frac{v_i(t) - v_{th}}{v_i(t)T_{i,h}(t)^{\gamma_i}} \right)^{1.2},$$  

(8)

$$0 \leq v(t) \leq v_{min}, \quad \forall t.$$  

(9)

In the above formulation, $t_f$ represents the final completion time or the makespan of all tasks, whose completion times
are given by $t_{f,i}$. Each task starts at time $0$ and should finish its execution by its deadline $t_{d,i}$ as stated in (4). The rate of execution is related to the speed of a core through (3). Each core has to be operated at a speed such that the temperature of its hottest block $T_{i,b}$ is less than the maximum temperature $T_{max}$. In this work, $T_{max}$ is the maximum specified junction temperature $TJ_{max}$ of a chip, which is the temperature limit for safe and reliable operation of a processor as set by the manufacturer [38].

Equation (5) shows a key difficulty in the temperature computation of each block in a core. $f$ is some yet unknown function. It shows the cyclic dependency of temperature with leakage power of a core. We simplify this by approximating the relation between the leakage power and the temperature with a piece-wise linear function as discussed in the next section.

### B. Piecewise-linear (PWL) Model of LDT

Leakage power increases exponentially with temperature and voltage of a core [3]. One necessary and useful simplification is to split the range of temperatures and voltages into several regions and then compute a piecewise linear approximation for each region. This is explained with the aid of Fig. 3, where the surface of leakage power is triangulated w.r.t. the temperature of a thermal block and the voltage of a core. The corresponding equation for leakage power of a triangulated region of the PWL model is given below.

$$ P_{i,b} = P_{0,i,b} + k_{T,i,b}T_{i,b} + k_{v,i,b}v_{i}. $$

$k_{T,i,b}$ and $k_{v,i,b}$ are the slopes of leakage power versus the temperature and the voltage of block $b$ in core $i$ respectively for the given triangulated section.

### C. Decoupling of Leakage Power and On-chip Temperature

Here the results from [39] are used to decouple the LDT. The key observation in this derivation is the fact that the thermal time constants of the die are three orders of magnitude smaller than the thermal time constant of the package. Thus for thermal transients on the die, the package temperature appears constant. With this assumption, it can be shown that the temperature of a block $b$ within core $i$ is given by

$$ T_{i,b}(t) = \zeta_{i,b}T_{p}(t) + R_{i,b}P'_{i,b}(s_i, v_i, t), $$

where

$$ \zeta_{i,b} = (1 - k_{T,i,b}^2R_{i,b})^{-1}, \quad \text{and} \quad P'_{i,b}(s_i, v_i, t) = \zeta_{i,b}[P_{0,i,b} + k_{T,i,b}^2T_{i,b} + k_{v,i,b}v_{i}(t)v_{i}^2(t)P_{d,i,b}^{max}(t)]. $$

$P'_{i,b}(s_i, v_i, t)$ is called the apparent power for the reason that it has the same form as the power $P_{i,b}$ and used in its place in subsequent equations. Note that $P_{d,i,b}^{max}(t)$ corresponds to the maximum power dissipation at the maximum frequency and the maximum voltage. $P_{d,i,b}^{max}(t)$ is a function of the access rate of block $b$ of core $i$ and it varies with time.

Using the above equation, the circular dependency between the leakage power and the die temperature is removed and the block power $P_{i,b}$ is made to depend only on the package temperature $T_{p}$.

$$ P_{i,b}(s_i, v_i, t) = P_{0,i,b}(v_i, t) + P_{d,i,b}(s_i, v_i, t) = P'_{i,b}(s_i, v_i, t) + (\zeta_{i,b} - 1)T_{p}(t)/R_{i,b}. \quad (14) $$

In the absence of LDT, $k_{T,i,b} = 0$, and $\zeta_{i,b} = 1$, and $P'_{i,b} = P_{i,b}$. Even with the decoupling of the leakage power and the die temperature, the temperature of each core is still affected by the activity of other cores. However, this dependence is through the package temperature.

### D. Computation of Package Temperature

With the above PWL approximation to the LDT, the package temperature is computed from the high-level thermal model described in Section II-B. Let

$$ P'(s, v, t) \triangleq \sum_{i=1}^{n} \sum_{b=1}^{m} \zeta_{i,b}P'_{i,b}(s_i, v_i, t) \quad \text{and} \quad G \triangleq \sum_{i=1}^{n} \sum_{b=1}^{m} (\zeta_{i,b} - 1)/R_{i,b}. $$

From Fig. 2 the package temperature is computed as

$$ \frac{dT_{p}(t)}{dt} = -\frac{T_{p}(t)}{R_{p}C_{p}} + \frac{1}{C_{p}} \sum_{i=1}^{n} \sum_{b=1}^{m} P_{i,b}(s_i, v_i, t). \quad (15) $$

Substituting for $P_{i,b}$ from (14),

$$ \frac{dT_{p}(t)}{dt} = -\frac{T_{p}(t)}{R_{p}C_{p}} + \frac{1}{C_{p}} \sum_{i=1}^{n} \sum_{b=1}^{m} \left[ \zeta_{i,b}P'_{i,b}(s_i, v_i, t) + \left( \zeta_{i,b} - 1 \right)T_{p}(t) \right]/R_{i,b} $$

$$ = -\frac{T_{p}(t)}{R_{p}C_{p}} + \frac{P'(s, v, t) + GT_{p}(t)}{C_{p}} \quad (16) $$

where $R_{p}^{' \triangleq} R_{p}(1 - GR_{p})$. $T_{p}(t)$ can be obtained by solving the above first order ODE.
Having decoupled the LDT, knowing how to compute the package temperature, the original optimization problem is re-formulated as follows:

\[
\min \int_0^{t_f} \left[ \sum_{i=1}^{n} s_i(t) + \sum_{i=1}^{m} v_i(t) \right] dt,
\]

subject to:

\[
\frac{dN_i(t)}{dt} = IPC_i(s_i(t), \forall t, i), 
\]

\[
N_i(0) = 0, \ N_i(t_{d, i}) = N_{tot, i}, \forall i,
\]

\[
T_{i,h}(t) = \max_b \{T_{i,b}(t)\}, \forall t, i,
\]

\[
\zeta_{i,h} R_{i,h} \left[ T_p(t) + P'_{i,h}(s_i, v_i, t) \right] \leq T_{max}, \forall t, i,
\]

\[
P'_{i,b}(s_i, v_i, t) = P_{0,i,b} + k_i^v v_i(t) + s_i(t) v_i^2(t) P_{d,i,b}^{max}, \forall t, i, b,
\]

\[
\frac{dT_p(t)}{dt} = -\frac{T_p(t)}{R_p C_p} + \frac{P'(s, v, t)}{C_p}, \forall t,
\]

\[
P'(s, v, t) \triangleq \sum_{i=1}^{n} \sum_{b=1}^{m} \zeta_{i,b} P'_{i,b}(s_i, v_i, t),
\]

\[
T_p(0) = T_{p0},
\]

\[
s_i(t) \leq k_i^v \left( \frac{v_i(t) - v_{ih}}{v_i(t) T_{i,h}(t)} \right)^{1.2}, \forall i,
\]

\[
o_0 \times 1 \leq v_i(t) \leq 1_n \times 1, \forall t.
\]

\(R_{i,h}, P'_{i,h}\) and \(\zeta_{i,h}\) correspond to the hottest block \(h\) of core \(i\) as determined in (20).

The die temperature (21) is the same as (11), but for the hottest block \(h\) and replaces (7). Equation (23) represents the fact that the die temperature is expressed in terms of the package temperature. The initial conditions on the die temperatures (6) are changed in terms of the initial condition on the package temperature (25).

The above formulation falls under minimum-time problems in optimal control theory [40]. \(N\) and \(T_p\) are state variables with fixed boundary conditions. The mixed control-state pointwise inequality (21) complicates the derivation of the optimal solution. The solution is obtained in two steps. In the first step, task deadlines are ignored, and the optimal speeds and voltages that minimize the makespan subject to a maximum temperature constraint are derived (see Section IV). Next in Section V a solution that meets the deadlines is constructed which achieves the minimum makespan among all deadline feasible solutions.

**IV. OPTIMAL SOLUTION FOR MINIMUM MAKESPAN WITHOUT DEADLINES**

If the deadlines are ignored then the constraint on completion times in (19) changes to \(N_i(t_f) = N_{tot, i} \forall i\). The optimal solution is obtained through the use of direct adjoining approach [41]. The optimal speed of core \(i\) is given by

\[
s_i^*(t) = \begin{cases} 
1, & T_{i,h}(t) < T_{max}, \\
0, & T_{i,h}(t) > T_{max}, \\
\text{smax}_{i,h}(t), & T_{i,h}(t) = T_{max}.
\end{cases}
\]

The proof of (28) is given in Appendix A. The quantity \(\text{smax}_{i,h}\) is the speed of core \(i\) which maintains the hottest block of core \(i\) at the maximum temperature. Equation (28) expresses the optimal speed policy and is explained with the aid of Fig. 4.

Initially the temperature of the hottest block is less than the maximum, hence the speed of the core is set to the maximum. Once the temperature of the hottest block reaches the maximum at time \(t_{m,i}\), the speed is decreased according to \(s_{\text{max}, i}\) to maintain the temperature of the hottest block at the maximum. Finally the task completes at time \(t_{c,i}\). This policy can be also be stated as follows: for the minimum makespan, there should not be slack in both the speed of a core and the temperature of its hottest block at the same time during the execution of a task. This is referred to as the zero-slack policy and the corresponding solution will be referred to as the minimum makespan solution (MMS).

The optimal voltage \(v_i^*\) corresponding to \(s_i^*\) is obtained by solving the following equation.

\[
s_i^*(t) = k_i^v \left( \frac{v_i^*(t) - v_{ih}}{v_i^*(t) T_{i,h}(t)} \right)^{1.2}.
\]

To compute \(s_i^*\), several quantities need to be determined. These are: (1) \(T_p\), (2) \(h\) – the identity of the hottest block in core \(i\) and its temperature \(T_{i,h}\), (3) \(t_{m,i}\) – the time at which the temperature of core \(i\) reaches \(T_{max}\), and (4) \(s_{\text{max}, i}\). These quantities are all inter-dependent. Consequently, they must be computed iteratively at discrete points in time. The time interval between two successive computations, denoted by \(t_s\), is called the scheduling interval, which is on the order of the die thermal constant (i.e. a few milliseconds).

Let \(t_k = k t_s\) denote the \(k\)th time point at which the speed \(s_i\) and voltage \(v_i\) are to be updated. The initial package temperature \(T_p(0)\) is known. The following is the sequence of steps involved in computing (28).

1) Using (16), compute \(T_p(t_k) = T_p(t_{k-1}) + \frac{dT_p(t_{k-1})}{dt} t_s\).
2) Identify the hottest block \(h = h_i(t_k)\) in core \(i\), and compute \(T_{i,h}(t_k)\). Note that the identity of the hottest block changes with time as does its temperature. However, at a fixed \(t\), the identity does not change with speed and voltage because all blocks are affected in the same way by a core’s speed and voltage. Therefore, \(P'_{i,h}(s_i = 1, v_i = 1, t_k)\), \(\forall b\) is computed using (13). This is then substituted into (11) to compute \(T_{i,b}(t_k)\), \(\forall b\).

Then \(T_{i,b}(t_k) = \max_b \{T_{i,b}(t_k)\}\).
3) If $T_{i,h}(t_k) < T_{max}$ then set $s_i^*(t_k) = 1$, and compute the corresponding $v_i^*(t_k)$ using (29). Otherwise, if $T_{i,h}(t_k) \geq T_{max}$ and $T_{i,h}(t_{k-1}) < T_{max}$, set $t_{m,i} = t_k$, and $s_i(t_k) = s_{max,i}(t_k)$. The computation of $s_{max,i}(t_k)$ is described next.

A. Determination of $s_{max,i}$

$s_{max,i}$ is defined as the speed necessary to maintain the temperature of the hottest block at the maximum temperature $T_{max}$. Equating $T_{i,h}$ to $T_{max}$ for the hottest block $h$ in (11), the package temperature is given by

$$T_p(t) = [T_{max} - P'_{i,h}(s_{max,i}, v_{max,i}, t)R_{i,h}] / \zeta_{i,h}, \forall i$$

(30)

$P'_{i,h}$ is the apparent power dissipation of block $h$ in core $i$ that corresponds to the speed $s_{max,i}$ and the voltage $v_{max,i}$.

Substituting the above equation in (16) $P'_{i,h}$ is expressed as,

$$\frac{dP'_{i,h}(s_{max,i}, v_{max,i}, t)}{dt} = \gamma_i - \alpha_i P'_{i,h}(s_{max,i}, v_{max,i}, t) - \sum_{c \neq i} \beta_{c,b} P'_{i,h}(s_{max,i}, v_{max,i}, t),$$

(31)

where

$$\alpha_i \equiv R_{i,h} + \zeta_{i,h} P'_{i,h} / R_{i,h}$$

$$\beta_{c,b} \equiv \frac{\zeta_{c,b}}{R_{c,b}}$$

and $\gamma_i \equiv \frac{T_{max}}{R'_{p,C}C_R}.$

The solution to (31) is

$$P'_{i,h}(s_{max,i}, v_{max,i}, t) = P'_{i,h,0}e^{-\frac{t}{\tau_i}} + P'_{i,h,ss}(1 - e^{-\frac{t}{\tau_i}}), \forall t > t_{m,i},$$

(32)

where $P'_{i,h,0}$ is the apparent initial power consumption and the apparent steady state power consumption of the hottest block of core $i$, and $\tau_i$ is the time constant of the power curve of the hottest block. These quantities are computed as follows.

$$P'_{i,h,0} = \frac{[T_{max} - \zeta_{i,h}T_p]}{R_{i,h}},$$

(33)

$$\tau_i = \left(\alpha_i + \sum_{c \neq i} \beta_{c}\right)^{-1},$$

(34)

$$P'_{i,h,ss} = \gamma_i \tau_i.$$

(35)

In (34), $n_a$ is the set of active cores, whose operating frequencies are greater than zero.

All the terms on the R.H.S of (32) are known. The L.H.S of (32) is an expression in the unknowns $s_{max,i}$ and $v_{max,i}$. The expression is obtained by substituting $h$ for $b$, $s_{max,i}$ for $s_i$ and $v_{max,i}$ for $v_i$ in (13). This results in one equation in the unknowns $s_{max,i}$ and $v_{max,i}$. The second equation that relates these two quantities is (29). Solving these two simultaneously yields the values of $s_{max,i}$ and $v_{max,i}$.

V. MINIMUM MAKESPAN WITH DEADLINES

Minimizing makespan does not guarantee that task deadlines will be satisfied. Fig. 5 shows a hypothetical example of a two-core processor executing two tasks with deadlines. Fig. 5(a) shows the minimum makespan solution, that violates the task deadlines. Fig. 5(b) shows a desired execution of the same tasks that respects both the task deadlines and the thermal constraints. Since minimum makespan is a global objective, over some interval it may execute a task at slower speed than the minimum speed required to meet the task deadline.

This section addresses the problem of determining the speed $s_i(t)$ and the voltage $v_i(t)$ of each core so that the given task deadlines are satisfied.

The proposed method constructs a deadline feasible solution, if one exists. Moreover, such a solution will have the minimum makespan.

Without loss of generality it is assumed that the $n$ tasks are numbered and ordered by their deadlines such that $d_{1,2} < \cdots < d_{n,n}$.

If the MMS satisfies the deadlines, then it is the solution sought. Therefore it is assumed that the MMS has at least one deadline violation.

In general, in a MMS, if the completion time $t_{e,n}$ of the last task, exceeds its deadline $t_{d,n}$, there can be no deadline feasible solution. This is because, by the definition of a MMS, $[0, t_{e,n}]$ is the shortest interval within which all instructions of all tasks are completed, and hence, $t_{e,n}$ cannot be reduced. Therefore it is assumed that task $i$, for some $i \leq n-1$, is the first task that violates its deadline. Such a task is referred to as a critical task. The intervals in which the speed of task $i$ can be modified are denoted by $I_1 = [0, t_{e,1}], I_2 = [t_{e,1}, t_{e,2}], \ldots, I_i = [t_{e,i-1}, t_{d,i}]$ (see Fig. 6(a)).

Speeding up task $i$ to reduce its completion time must be done so that $t_{e,i} = t_{d,i}$. If task $i$ is sped up so that $t_{e,i} < t_{d,i}$, then this could lead to 1) deadline violations of earlier tasks, or 2) as a consequence of the zero-slack policy, reduction in throughput which cannot be regained, and can cause deadline violations of later tasks. In other words a deadline feasible solution with $t_{e,i} < t_{d,i}$ would have a makespan no smaller than one with $t_{e,i} = t_{d,i}$.
A. Critical Task Voltage-speed Determination

The method to adjust the speed profile of a critical task is explained with an example. Fig. 6(a) shows a MMS with four tasks, in which Task 3 is the critical task. Fig. 6(b) shows the desired deadline feasible solution. The modification of the speed of Task 3 is done by successively examining the intervals $I_3 = [t_{e,2}, t_{d,3}]$, $I_2 \cup I_3 = I(t_{e,1}, t_{d,3})$ and $I_1 \cup I_2 \cup I_3 = [0, t_{d,3}]$.

Consider the interval $I_3 = [t_{e,2}, t_{d,3}]$. The objective is to determine $s_3(t)$ and $v_3(t)$ in $I_3$. The initial package temperature in $I_3$ is $T_p(t_{e,2})$, and is retained from the original makespan solution. The final package temperature in $I_3$, viz. $T_p(t_{d,3})$ needs to be reduced to increase $s_3(t)$.

Recall that in a MMS, the speed $s_3(t) = 1$ until the time $t_{m,3}$, which is the time when $T_{3,h} = T_{max}$. Therefore if $s_3(t_{e,2}) < 1$, then $t_{m,3} < t_{e,2}$. If $s_3(t_{e,2}) = 1$, then $t_{m,3} \in [t_{e,2}, t_{d,3}]$.

Suppose $s_3(t_{e,2}) < 1$. Then $s_3(t)$ and $v_3(t)$ for $t \in [t_{e,2}, t_{d,3}]$ are the same as $s_{max,h}(t)$ and $v_{max,h}(t)$. They are determined as follows.

1) Since $t_{e,2}$ is the left endpoint of the interval, the apparent initial power $P'_{3,h,0}$ in (33) refers to $P'_{3,h}(t_{e,2})$. Therefore $P'_{3,h}(t_{e,2}) = [T_{max} - \zeta_{3,h} T_p(t_{e,2})]/R_{3,h}$.

2) Let $P'_{3,h,lb}(t_{d,3})$ and $P'_{3,h,ub}(t_{d,3})$ be the lower and the upper bounds on $P'_{3,h}(t_{d,3})$ corresponding to speeds $s_3(t_{d,3}) = 0$ and $s_3(t_{d,3}) = 1$ respectively. Note that the corresponding $v_3(t_{d,3})$ is derived from (29) with $T_{3,h}(t_{d,3}) = T_{max}$. Let $T_p(t_{d,3})$ and $T_p(t_{d,3})$ be the corresponding lower and upper bounds on the package temperature $T_p(t_{d,3})$ obtained by substituting $P'_{3,h,lb}(t_{d,3})$ and $P'_{3,h,ub}(t_{d,3})$ respectively for $t = t_{d,3}$ in (30).

3) Perform a binary search on $T_p(t_{d,3}) \in [T_p(t_{d,3}), T_p(t_{d,3})]$ and find the corresponding $P'_{3,h}(t_{d,3}) = [T_{max} - \zeta_{3,h} T_p(t_{d,3})]/R_{3,h}$. Substituting $P'_{3,h}(t_{d,3})$ in (32) for $t = t_{d,3}$ and using (34), compute $P'_{3,h,0}$ and $\tau_{3,h}$ numerically.

4) Now the R.H.S of (32) is known for all $t \in [t_{e,2}, t_{d,3}]$. This is equated to the R.H.S of (13) for $P'_{3,h}$, which yields one equation in $s_{max,h}$ and $v_{max,h}$. These are solved numerically along with (29).

5) Let $N_{rem,3} = N_{tot,3} - \int_0^{t_{d,3}} IPC_3(t) s_3(t) dt$, be the remaining number of unexecuted instructions in $[0, t_{d,3}]$.

If $N_{rem,3} \neq 0$, Steps 3, 4 and 5 are repeated until $N_{rem,3} = 0$. This is done by repeatedly choosing $T_p(t_{d,3}) \in [T_p(t_{d,3}), T_p(t_{d,3})]$. If $s_3(t_{e,2}) = 1$, then $t_{m,3} \in [t_{e,2}, t_{d,3}]$. $t_{m,i}$ is determined through a binary search. For every binary search on $t_{m,i}$, the above steps 1 – 5 are repeated, replacing $t_{e,2}$ by $t_{m,3}$.

If the above steps do not yield a deadline feasible solution for Task 3, then the search interval is expanded to $I_2 \cup I_3 = [t_{e,1}, t_{d,3}]$ by setting $T_p(t_{d,3}) = T_p(t_{d,3})$. The above steps are repeated to search the appropriate package temperature at $t_{e,2}$, i.e. $T_p(t_{e,2})$ (instead of $T_p(t_{d,3})$), such that $N_{rem,3} = N_{tot,3} - \int_0^{t_{d,3}} IPC_3(t) s_3(t) dt = 0$.

B. Speeds of non-critical tasks

Let $[t_{sec}, t_{ec}]$ be the interval, where the critical task speed was altered from its original MMS. Therefore, the speeds of other non-critical tasks need to be determined only in this interval. First, the package temperature $T_p(t), t \in [t_{sec}, t_{ec}]$ is obtained that corresponds to the power profile of the hottest block of the critical task $P'_{3,h}$ from (30). Then $\frac{dT_p(t_k)}{dt}$ is computed for every scheduling interval $t_s$ as

$$\frac{dT_p(t_k)}{dt} \approx \frac{T_p(t_k) - T_p(t_k-1)}{t_s}, \quad (36)$$

Since both $\frac{dT_p(t_k)}{dt}$ and $T_p(t_k)$ are known, the total apparent power consumption for all cores $P(t_k)$ in the $k^{th}$ scheduling interval can be determined from (16). Let $P(t_k) = P'(t_k)$ (critical task is $i$) be the remaining total apparent power budget that has to be allocated to the remaining tasks at time $t_k$. This allocation needs to ensure

1) that the previous tasks $1, \ldots, i-1$ satisfy their deadlines, and
2) the number of deadline violations in future tasks $i+1, \ldots, n$ are minimized.

Since ensuring that the previous tasks satisfy their deadlines is important, they get higher priority in the order of their deadlines, while allocating the power budget. Note that the previous tasks are not allocated any more power than it is required to satisfy their deadlines and if any of the deadline constraints is not satisfied, then there is no deadline feasible solution to satisfy deadlines of all tasks. To determine the priority of allocation of power budget to the rest of the tasks $i+1, \ldots, n$, the following procedure is followed.

1) Run the zero-slack policy for the interval $[t_{d,i}, t_{d,n}]$, for tasks $i+1, \ldots, n$, with $N'_{tot,c} = N_{tot,c} - \int_0^{t_{d,n}} IPC_c(t) s_c(t) dt$, with instructions.

2) Let $D$ denote the temporal order of tasks whose deadlines have been violated under the zero-slack execution.
Let $L = 0$; Run the zero-slack policy for all cores (Section IV);

while any task violates deadlines do

Let task $i$ first violate (critical task);
Let $R = i$, $t_{sc} = t_{e,R-1}$ and $t_{ec} = t_{d,i}$;

while task $i$ violates deadline do

Determine $(s_i, v_i)$ profile for task $i$ for interval $[t_{sc}, t_{ec}]$ (Section V-A);
if $\int_0^t IPC_i(t)s_i(t)dt \neq N_{tot,i}$ then

if $R = L$ then

No deadline feasible solution exists;
return;

else

$L = R - 1$, $t_{sc} = t_{e,R-1}$ and $t_{ec} = t_{e,R}$;

non_critical($s_i$, $[t_{sc}, t_{ec}]$, $i$);
$L = i$;
Run zero-slack policy for cores $i + 1, \ldots, n$;

end

end

end

Algorithm 2: Procedure for feasible speed determination of tasks with deadlines under thermal constraints.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>basicmath</th>
<th>qsort</th>
<th>IPv4</th>
<th>GSM dec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst. (billions)</td>
<td>116</td>
<td>82</td>
<td>111</td>
<td>140</td>
</tr>
<tr>
<td>IPC</td>
<td>1.8</td>
<td>1.85</td>
<td>1.95</td>
<td>2.45</td>
</tr>
<tr>
<td>Avg. dyn. power (W)</td>
<td>60.13</td>
<td>56.75</td>
<td>55.41</td>
<td>57.54</td>
</tr>
<tr>
<td>Deadlines (seconds)</td>
<td>60</td>
<td>25</td>
<td>40</td>
<td>75</td>
</tr>
</tbody>
</table>

TABLE III

INSTRUCTION LENGTH, IPC, AVERAGE DYNAMIC POWER AND DEADLINES OF THE TASKS USED IN THE EXPERIMENT.

The above steps are necessary to determine the number of instructions that are critical to avoid future deadline violations. Hence those tasks that have deadline violations are given higher priority. During the allocation of the power budget in the interval $[t_{sc}, t_{ec}]$, if any of the tasks from $D$ execute $N'_{rem}$, then those tasks are removed from $D$ and are assigned normal priority as they are no more critical. Note that during the allocation of power, the maximum power that can be allocated to a task in a scheduling interval is fixed. The power consumption of the hottest block is bounded by $T_p$ and $T_{max}$ as seen from (11). This in turns limits the maximum speed of that core and its total power consumption.

The details of the voltage-speed determination of the non-critical tasks is summarized in Algorithm 1 and the overall procedure of deadline feasible speed determination in Algorithm 2.

The above procedure is applicable even in the case of multiple tasks sharing same deadline. One among the multiple tasks is identified as the critical task and rest of the tasks, including tasks that shared the critical task deadline, are grouped as non-critical tasks.

The time-complexity of the deadline-feasible speed determination algorithm is $O(n (\log n) t_{d,n}/t_s)$, where $t_{d,n}/t_s$ is the total number of scheduling intervals, and $\log n$ is due to the binary search on $T_p$, to determine the speeds of critical tasks.

VI. EXPERIMENTAL RESULTS

A. Experimental Setup

For the simulation experiments, we created a hypothetical multi-core version of Alpha 21264 processor by replicating single Alpha cores and scaling the cores to fit the size of a single core Alpha processor. The HotSpot thermal model [2] was used to model the thermal behavior of the processor. The convectional thermal resistance was set at $0.35 \, ^\circ C/W$. The power characteristics were obtained using PTSemulator [3] on MiBench benchmarks [42]. The maximum temperature was set at $110 ^\circ C$. The dynamic power was limited to $230 \, W$, while the leakage power was set at $60 \, W$. The maximum frequency of cores were set at $2 \, GHz$. The speeds were scheduled at an interval of 10 ms.

B. Makespan Minimization for Tasks without Deadlines

Fig. 7 shows the plot of speeds, voltages and temperatures of the hottest blocks of cores under the optimal MMS for a four
core processor executing four tasks. The details of the tasks along with their instruction length are shown in Table III. Since the instruction lengths of the original benchmarks were very small to observe appreciable influence of temperature on the execution of tasks, we repeatedly executed the benchmarks for the duration of the package time constant. The corresponding number of instructions are noted in Table III. The deadlines were chosen such that an individual benchmark when executed alone on a multi-core processor at the maximum speed should complete its execution by its deadline, but not when executed with other tasks at the maximum feasible speeds. When multiple tasks execute on a processor, the power dissipation from all the tasks increase the package temperature, thus reducing the operational speed of cores.

As mentioned in Section IV, the MMS determines the core speeds and voltages such that the temperature of the hottest block is maintained at the maximum temperature as seen in Fig. 7. Note that the fluctuations in the core speeds and voltages are due to the time-varying power profiles.

For the purpose of demonstrating the MMS, we ignore the deadlines in this section. Note that the completion of a task allows other tasks to execute at a higher speed due to reduced power dissipation. This we observe after the completion of task *qsort* in Fig. 7.

Since we did not find equivalent policies that determines time-varying DVFS considering time-varying power profiles for multi-core processors, we could not provide a comparison of our policy with existing work.

---

### Table IV

<table>
<thead>
<tr>
<th>Tasks</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMS (s)</td>
<td>36.4</td>
<td>45.3</td>
<td>32.9</td>
<td>48.8</td>
<td>47.6</td>
</tr>
<tr>
<td>Modified algo. (s)</td>
<td>73.1</td>
<td>86.5</td>
<td>77.3</td>
<td>88.9</td>
<td>86.4</td>
</tr>
<tr>
<td># violations</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Comp. time/core (min)</td>
<td>2.1</td>
<td>2.15</td>
<td>2.03</td>
<td>1.97</td>
<td>2.18</td>
</tr>
</tbody>
</table>

#### C. Makespan Minimization for Tasks with Deadlines

We add the deadline constraints from Table III to demonstrate that the optimal MMS only ensures achieving the minimum makespan but does not guarantee satisfying the deadlines. The deadline constraints are taken into consideration by our modified algorithm as demonstrated in this section.

Fig. 8 shows the plot of speeds, voltages and temperatures of cores when scheduled according to the modified makespan minimization algorithm which satisfies the deadlines. Unlike the MMS which violated the deadlines (Fig. 7), the modified algorithm resulted in a higher makespan (73.1 s) to ensure that both the thermal and the deadline constraints are satisfied for all tasks as seen from Fig. 8.

Annotated portion ‘A’ in both Figs. 7 and 8 illustrate the differences in the speed control between the optimal MMS and the modified algorithm. Task *qsort* could not meet its deadline (25 s) when scheduled according to the makespan minimization procedure. This is because, the speeds of all cores were kept at their thermally maximum speeds to mini-
mize the overall makespan. This resulted in constraining the speed of task qsort, which had the earliest deadline. On the other hand, the modified procedure ensured that the deadline constraint of task qsort is met by increasing its speed and constraining the speeds of other non-critical tasks optimally. Among the non-critical tasks, IFFT had the earliest deadline at 40 s. Hence it was allocated the maximum power out of the remaining total power budget. This ensured that it executed at a higher speed compared to other non-critical tasks as seen in Fig. 8. Note that the speeds of tasks basicmath and GSM dec. are zero in the first interval [0–25] as the remaining power budget was not sufficient to allocate for these tasks.

Table IV shows the results of additional experiments with the number of tasks (or cores) ranging from 4 to 64. The number of instructions and the corresponding deadlines for the workload specified in Table III. Fig. 9 shows the approximate discrete voltage-speed implementation of the modified minimum makespan solution satisfying deadlines for the workload specified in Table III with ten discrete voltage-speed states (Intel Core i processors support up to ten voltage-speed pairs). The discretization is done by selecting the highest discrete voltage and speed that is feasible for the corresponding temperature at every scheduling interval. Note that there is an increase in the completion times of tasks due to the approximation of speeds to the corresponding discrete states and thereby degrades the performance. This implementation demonstrates the possibility of a practical implementation of our policies in processors with discrete voltage-speed states.

D. Discrete Voltage-speed Implementation

Temperature-aware scheduling continues to play an important role in the design of real-time systems. Power-aware scheduling techniques are inadequate in addressing thermal-aware design issues and similarly, single-core optimal solutions are no longer optimal when adapted to multi-core processors. In this paper, a novel DVFS technique is proposed to determine a feasible set of core speeds and voltages that guarantee completion of tasks on a multi-core processor while meeting the deadlines of tasks and also ensuring that the thermal constraints are satisfied. The proposed solution makes use of accurate power and thermal models, including leakage dependence on temperature. The solution is shown to be of polynomial-time-complexity and is demonstrated through experiments that the proposed solution achieves the minimum makespan and also meets the deadlines of all tasks while satisfying the temperature constraints.

VII. CONCLUSION

Temperature-aware scheduling continues to play an important role in the design of real-time systems. Power-aware scheduling techniques are inadequate in addressing thermal-aware design issues and similarly, single-core optimal solutions are no longer optimal when adapted to multi-core processors. In this paper, a novel DVFS technique is proposed to determine a feasible set of core speeds and voltages that guarantee completion of tasks on a multi-core processor while meeting the deadlines of tasks and also ensuring that the thermal constraints are satisfied. The proposed solution makes use of accurate power and thermal models, including leakage dependence on temperature. The solution is shown to be of polynomial-time-complexity and is demonstrated through experiments that the proposed solution achieves the minimum makespan and also meets the deadlines of all tasks while satisfying the temperature constraints.

REFERENCES

K. Stavrou and P. Trancoso, “Thermal-aware Scheduling for Future Chip...

R. Rao and S. Vrudhula, “Efficient Online Computation of Core Speeds...

S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D.

V. Hanumaiah, S. Vrudhula, and K. S. Chatha, “Performance Optimal...

R. M. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge,