GOP-Level Dynamic Thermal Management in MPEG-2 Decoding

Wonbok Lee, *Student Member, IEEE*, Kimish Patel, *Student Member, IEEE* and Massoud Pedram, *Fellow, IEEE*

***Abstract*—In this paper, we present a dynamic thermal management (DTM) algorithm based on (i) accurate estimation of the workload of frames in a group of pictures (GOP) in an MPEG-2 video stream and (ii) slack borrowing across the GOP frames in order to achieve a thermally safe state of operation in microprocessors during the video decoding process. The proposed DTM algorithm employs dynamic voltage and frequency scaling (DVFS) while considering the frame-rate-dependent GOP deadline, variance of the frame decoding times within the GOP, and a maximum chip temperature constraint. If it becomes necessary to sacrifice video quality or violate the GOP deadline due a low temperature bound, then the (intra-frame) spatial quality degradation and the (inter-frame) temporal quality degradation will be applied to the GOP. Experimental results demonstrate the competence and efficiency of the proposed online DTM algorithm.**

***Index Terms*—Dynamic thermal management, dynamic voltage and frequency scaling, GOP, MPEG-2 decoding, quality degradation.**

I. Introduction

PEAK power dissipation and resulting temperature rise have become the dominant limiting factors to processor performance and constitute a significant component of its cost. Expensive packaging and heat removal solutions are needed to achieve acceptable substrate and interconnect temperatures in high-performance microprocessors. The heat flux in state-of-the-art microprocessors chips is currently in the range of 10-20 W/cm^2, which is already exceeding the confines of air cooling.

Current thermal solutions are designed to limit the peak processor power dissipation to ensure its reliable operation under worst-case scenarios. However, the peak processor power and ensuing peak temperature are hardly ever observed. Dynamic thermal management (DTM) has been proposed as a class of micro-architectural solutions and software strategies to achieve the highest processor performance under a peak temperature limit. Furthermore, it is known that power density across the chip is non-uniform, resulting in localized hot spots. DTM solutions must address this phenomenon as much as they tackle system-wide temperature violations. When the chip approaches the thermal limit, a DTM controller initiates hardware reconfiguration, slow-down, or shutdown to lower the chip temperature.

Due to the relationship between power density and temperature, thermal issues have received increasing attention from low power designers. The industry standard interfaces for operating system (OS)-directed power management on computers, i.e., Advanced Configuration and Power Interface (ACPI), points to the fact that the OS can play an important role in the thermal management of a system while maintaining a target level of performance for the platform.

Traditionally, thermal issues within a chip have been handled at the package level. Chip manufacturers have devised sophisticated, albeit expensive, packaging and cooling assemblies, i.e., heat sinks and micro-fluidic conduits, to the processor chips so as to efficiently transfer heat generated within a chip to the ambient environment. However, packaging and cooling systems without the knowledge about the resource utilization and power dissipation demands of a software program running on a microprocessor chip have some major limitations. As such, micro-architecture level solutions can result in changes to the dynamic temperature profile of a chip so as to avoid the worst-case power density and temperature conditions.

The thermal behavior of a microprocessor during a program run can be accurately observed only at runtime. As a consequence, Dynamic thermal management (DTM) has been proposed as a class of micro-architectural thermal solutions at runtime. Some examples of DTM methods include fetch-toggling (instruction fetching is stalled for next N cycles) and instruction cache throttling (throttle the instruction forwarding from the instruction cache to the instruction buffer), activity migration (dispatching computations to different locations on the die) and dynamic voltage and frequency scaling (DVFS). A common feature of most proposed DTM methods is their reactive nature, which relies on two pre-defined levels of thermal limits: a *trigger temperature*, $T_{trigger}$, and a *critical temperature,* $T_{critical}$. $T_{trigger}$ is a thermal limit above which the DTM method will be employed whereas $T_{critical}$ is a thermal limit above which the temperature could damage the microprocessor chip, and hence, it must not be exceeded. In the remainder of this paper, we set $T_{trigger}$=$T_{critical}$.

Manuscript submitted, February 8, 2007.
Wonbok Lee is with Dept. of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA (e-mail: wonbokle@usc.edu).
Kimish Patel is with Dept. of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA (e-mail: kimishpa@usc.edu).
Massoud Pedram is with Dept. of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA, (e-mail: pedram@usc.edu).

Unfortunately, thermal safety achieved by these DTM methods typically comes at the expense of appreciable (speed) performance degradation, which is sometimes intolerable. A good example is the MPEG decoding task where maintaining a fixed throughput, i.e. frame rate, is crucial. In such applications, quality may be sacrificed in order to achieve the desired performance. The present paper introduces a DTM algorithm which applies DVFS and possibly video quality (spatial/temporal) degradations in order to meet both performance and maximum temperature constraints for MPEG-2 decoding.

II. Prior Work

Many of the requisite performance features of microprocessors such as real-time processing and the mean-time-between-failure (MTBF) are significantly affected by the power density and resulting temperature [1]. In fact, the power dissipation and heat density of microprocessors have increased super linearly to the point that they affect the microprocessors' performance, aging, reliability, and the total system cost including the heat removal solutions [2].

DTM methods can be classified as: 1) local vs. global, 2) aggressive vs. conservative, 3) HW-based vs. SW-based, and 4) spatial vs. temporal. For example, spatial methods, e.g., AM, distribute excess heat to different locations within a microprocessor whereas the temporal methods, e.g., DVFS, distribute the excess heat over time.

In [3], Brooks et al. explore a variety of hardware and software DTM methods, which include clock frequency scaling, DVFS, decode throttling, speculation control, and instruction cache fetch toggling. In [4], Heo et al. present an activity migration (AM) technique which is effective for temperature control in the register file, albeit this method was accompanied by large area overhead. In [5], Srinivasan et al. propose instruction window resizing and switching-off the active functional units, which in turn results in both issue width reduction and register file port deactivation. The authors combine dynamic voltage scaling (DVS), instruction window resizing, and functional unit adaptation in multi-media applications. In [6], Kumar et al. propose a hybrid-DTM, combining the reactive hardware techniques, i.e., clock gating, and proactive software techniques, i.e., process scheduling. In [7], Skadron et al. incorporate fetch gating during low thermal stress times and DVS during high thermal stress times.

A practical DTM solution uses thermal sensors that can generate processor-wide thermal profile. However, this type of approach raises delicate issues such as, optimal sensor deployment, sensor's response time, and chip-temperature increase by the sensor(s). To make it feasible, some researches have utilized the built-in performance monitoring feature (PMU) [8] in off-the-shelf processors. In [9][10], Frank et al. use the PMU in the Linux operated Intel Pentium IV based platform and extract the power/energy approximation on-the-fly. In [11], Lee et al. propose another PMU based DTM solution which interfaces with HotSpot [12][13]. In [14][15], Chung et al. extend the aforesaid work by adopting the off-line regression analysis to find the relationship between the event information and temperature.

Some research efforts have tried to find the thermal solutions in an early design phase, i.e., placement/floor-planning [16]-[19]. For example, in [17], Han et al. show that temperature-aware floor-planning can reduce the maximum temperature of an Alpha processor by 21℃ whereas in [18], Healy et al. present a multi-objective floor-planning algorithm considering not only layout area and wire length, but also performance (IPC) and temperature.

Several DTM works have taken advantage of extra resources in microprocessors. In [4], Heo et al. propose to replicate the functional units. In [20], Lim et al. attach a relatively simple secondary pipeline featured by single issue and in-order execution, to the primary pipeline featured by multiple issues and out-of-order execution. In [21], Kursun et al. propose to use helper engine to alleviate the switching overhead between duplicated cores.

Skadron et al. propose a formal feedback control theory which invokes DTM method in response to the localized hotspot rather than chip-wide temperature in [22]. In [23]-[24], the same authors present a compact thermal RC model, which is based on the well-known duality concept between the heat transfer and the current flow. This model has become the basis of the most widely used thermal simulator called HotSpot.

Most aforesaid DTM methods have achieved thermally safe state at the expense of performance. However, some real time applications cannot tolerate performance degradation. For example in MPEG, each frame decoding deadline, i.e., frame rate, should be met at all times/costs. There has been a number of research works that address power management in MPEG decoding, but very little has been done with respect to temperature management in MPEG. Son et al. [25] propose two DVS algorithms for MPEG decoding. One is DVS with delay and drop rate minimizing algorithm where voltage is determined based on previous workload only. Another algorithm scales the supply voltage according to the predicted MPEG decoding time and previous workload. Choi et al. [26][27] present an off-chip latency driven DVFS. Their DVFS strategy relies on frame dependent (FD) versus frame independent (FI) parts of the workload and the on-chip versus off-chip (i.e., CPU versus memory) workloads within the frame decoding task. To the best of our knowledge, none of the previous DTM works have considered paying quality degradation in order to meet performance requirement.

In this paper, we present a DTM method based upon the (per-frame decoding) slack time estimation and its distribution in the GOP to achieve thermally safe state of operation in microprocessors during MPEG-2 decoding. The proposed DTM method will incorporate DVFS with the consideration of per-frame deadline and temperature constraints in the GOP. When DVFS is predicted to fail to meet the performance constraints, then we do the (intra-frame) spatial quality

degradation and the (inter-frame) temporal quality degradation in order to make up for the loss of performance.

III. Dynamic Thermal Management

A. On Efficacy of DTM Methods

We have carried out detailed analysis of the temperature zones in terms of thermal gradients and classified these zones into two classes (cf. Fig. 1(a)): i) Fast Temperature Rise (FTR) zone - The rising thermal gradient is higher than the falling thermal gradient i.e., the temperature rises faster than it falls (when the chip is allowed to cool off). ii) Fast Temperature Fall (FTF) zone - The falling thermal gradient is equal to or higher than the rising thermal gradient i.e., the temperature drops faster than it rises. Note that the DTM methods are most effective in the FTF zone. Based on our simulations, the FTF zone is above the FTR zone. This is fortunate because the temperature profile of a microprocessor chip is such that DTM techniques become more effective as the chip temperature rises.

Depending on the type of packaging and cooling solutions, $T_{critical}$ may lie in one or the other of these two zones. In the absence of DTM techniques, an application program running on a microprocessor chip will result in a *steady-state temperature*, T_{ss}, depending on the program's characteristics e.g., its CPI. (Strictly speaking, T_{ss} corresponds to the steady-state temperature of the hottest functional block.) The goal of DTM techniques is to ensure that T_{ss} remains below $T_{critical}$ while minimizing power dissipation, maximizing performance, or meeting a combination of power-performance constraints. If the original T_{ss} is below $T_{critical}$, no DTM technique is needed. Otherwise, if $T_{critical}$ lies in the FTF zone (cf. Fig. 1(b), then the DTM methods tend to work very well and the new T_{ss} of the chip will be lowered below $T_{critical}$ with little performance penalty. Otherwise (cf. Fig. 1(c), the DTM techniques are expected to be less effective i.e., the new T_{ss} of the chip will be brought below $T_{critical}$ only with a significant performance penalty.

To mathematically support the idea, we use a thermal model developed by Skadron et al. in [22]. Based on this model, the temperature increase in the processor is represented by:

$$\Delta T = \left(\frac{P}{C_{th}} - \frac{T_{old}}{R_{th} \cdot C_{th}}\right) \cdot \Delta t \qquad (1)$$

where Δt is a time interval, P is the average power dissipated in an interval, R_{th} is a thermal resistance, C_{th} is a thermal capacitance and T_{old} is the initial temperature of a time period, respectively. After a time interval, the new temperature becomes:

$$T_{new} = T_{old} + \Delta T \qquad (2)$$

Let $t_{initial}$ and t_{final} denote two instances of time (and their difference be denoted by Δt), respectively. Then, the rising thermal gradient with respect to time is represented as:

$$\frac{\Delta T_r}{\Delta t} = \left(\frac{P}{C_{th}} - \frac{T_{old}}{R_{th} \cdot C_{th}}\right) \qquad (3)$$

In contrast, when the processor is stalled (it is put in the standby mode) with invocation of any DTM method, the chip power dissipation is negligible compared to its active power dissipation i.e., P=0. In that case while processor is stalled, the falling thermal gradient is calculated as:

$$\frac{\Delta T_f}{\Delta t} = \left(-\frac{T_{old}}{R_{th} \cdot C_{th}}\right) \qquad (4)$$

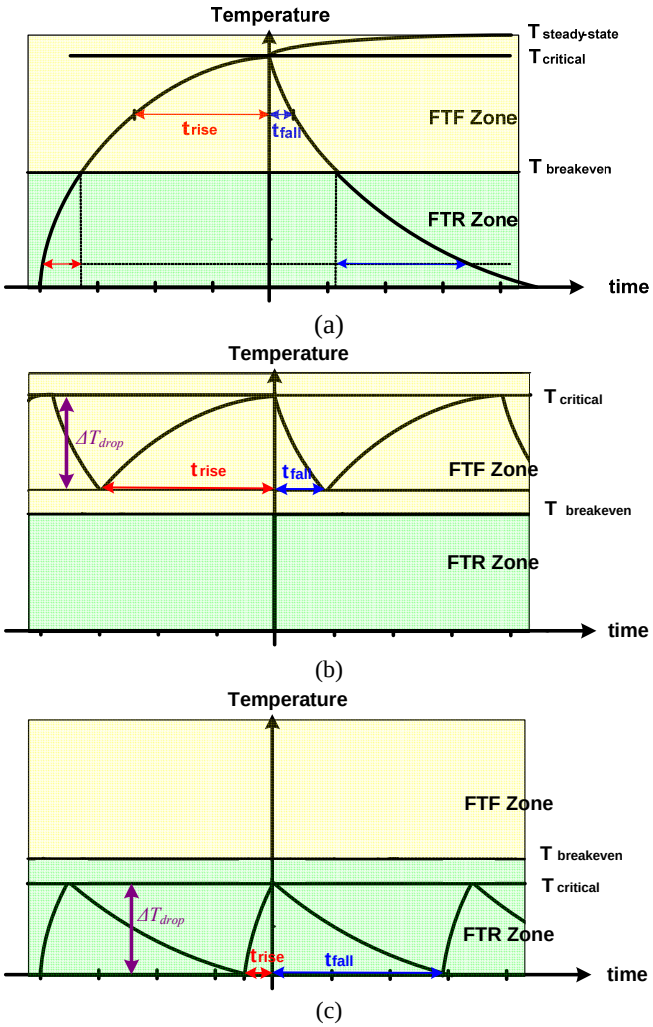

Fig. 1. (a) Thermal gradients at different temperature levels. (b) $T_{critical}$ is in the FTF zone; (c) $T_{critical}$ is in the FTR zone. $T_{breakeven}$ is a temperature at which rising and falling thermal gradients are equal. ΔT_{drop} is the minimum temperature drop before we deactivate the DTM i.e., go back to normal mode of operation. Notice that ΔT_{drop} is fixed and known in advance.

B. Thermal Behavior of Software Programs

1) General Applications

Fig. 2 depicts examples of thermal behavior of a number of application programs with different workloads. Prior to a program run, the microprocessor chip is at a certain *initial temperature*, which we denote as $T_{initial}$. When the program starts to run, the temperature rises and finally reaches a steady-state temperature, T_{ss}. Note that T_{ss} is program (workload) dependent but is not dependent on $T_{initial}$.

Eventually, every program reaches its own T_{ss}, albeit in different amount of (settling) time, which is denoted by t_{settle}. The T_{ss} values for different programs can be estimated based on the program behavior, e.g., its CPI-dependent power value, which is denoted by P_i.

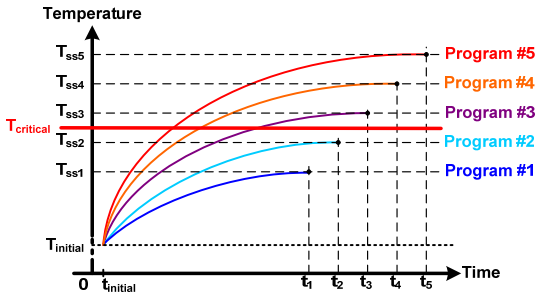

Fig. 2. Steady-state temperatures and the settling times of different application programs.

The thermal gradient, $\Delta T/\Delta t$, goes to zero as a program's thermal curve approaches T_{ss}. Unfortunately, T_{ss} is known only after a program's thermal curve reaches its steady state level. If we are able to predict T_{ss} of a program from its P_i, then we can use this predicted value of T_{ss} to select the best-effort voltage and frequency at any time during the program run. (In this context, the best-effort solution means the highest voltage and frequency that guarantees a steady state temperature below $T_{critical}$ while incurring minimum performance degradation.) For the level of $T_{critical}$ shown in Fig. 2, the difference between T_{ss} of the program (in this case programs #3, #4, and #5) and $T_{critical}$ can guide the degree of DVFS that guarantees thermal safety while incurring minimum performance degradation.

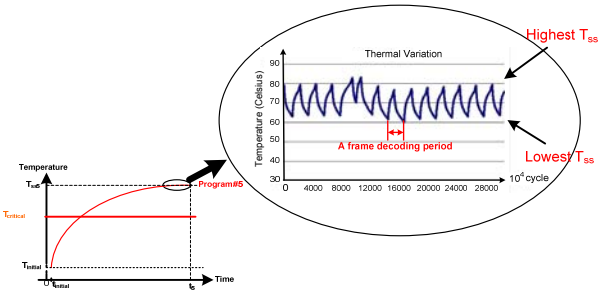

Fig. 3. Steady-state temperatures and the settling times of MPEG-2.

2) *Multi-media Applications*

The thermal curves of the multi-media applications are different from those of the general application programs. Fig. 3 shows one such example corresponding to an MPEG-2 decoding program [28]. As shown, the major difference is the existence of regular fluctuations in T_{ss}: When the frame decoding is finished earlier than the given deadline, microprocessor becomes idle (waiting to display the frame) until the next frame decoding starts. This pattern gives rise to a high temperature level (i.e., the temperature when the frame decoding is completed) and a low temperature (i.e., the temperature at the end of the idle time) on the thermal curve. Only if the frame decoding workload fully utilizes the microprocessor without any waiting (residual time), there is no significant fluctuation in T_{ss}.

To eliminate this frame-based T_{ss} fluctuation, we need to scale down the voltage and frequency such that no waiting period exists in the frame decoding. In contrast to the processor supported maximum voltage and frequency, which we denote as (V_{MAX}, f_{MAX}), we refer to such voltage and frequency by ($V_{max}^{no\ wait}$, $f_{max}^{no\ wait}$). In this case, the original T_{ss} will typically be lowered to a new temperature level denoted as $T_{ss}^{no\ wait}$ which is supposed to be lower than the original T_{ss}. Note that $T_{ss}^{no\ wait}$ is not always guaranteed to lie below $T_{critical}$ i.e., we may still have to further scale down the voltage and frequency to meet the chip's thermal constraints. Note that T_{ss} is reduced in both cases; however, the ($V_{max}^{no\ wait}$, $f_{max}^{no\ wait}$) combination does not incur any performance penalty. Hence the performance degradation is experienced only in the latter voltage and frequency scaling when $f < f_{max}^{no\ wait}$.

C. *Steady-State Temperature Calculation*

As mentioned earlier, we assume that the large-scale P_i's in a program run are more or less constant. Then, the constant power term and the current temperature (i.e., T_{old}) in equation (1) will determine ΔT and T_{new} at the end of the time period Δt. The T_{new} will be used as new T_{old}, and the same calculation repeats for the next time period of Δt. Eventually after certain number of such time periods, Δt, T_{new} will arrive at the temperature, defined as T_{ss}, when the thermal gradient satisfies $\Delta T/\Delta t \approx 0$, hence the calculation of equation (1) repeats until this condition is satisfied at which we arrived at T_{ss}. This flow is shown in equation (5):

$$T_{ss} = T_{initial} + \sum_{k=1}^{n^*} \left(\frac{P}{C_{th}} - \frac{T_k}{R_{th} \cdot C_{th}} \right) \cdot \Delta t \qquad (5)$$

where n^* is the minimum index where $\Delta T_{n^*} / \Delta t \approx 0$

The number of iterations, n^*, determines t_{settle}, which is $n^* \cdot \Delta t$. In fact, there is a more direct way of calculating T_{ss} as follows:

$$\frac{\Delta T}{\Delta t} = 0 \Rightarrow \frac{P}{C_{th}} - \frac{T_{ss}}{R_{th} \cdot C_{th}} = 0 \ \Rightarrow \ T_{ss} = P \cdot R_{th} \qquad (6)$$

If the calculated T_{ss} is below $T_{critical}$, program is thermally safe and no DTM method is needed. If the calculated T_{ss} is above $T_{critical}$, the temperature gap between T_{ss} and $T_{critical}$ will set the degree of DTM that is needed. Notice that R_{th} is constant for the given microprocessor chip and the only parameter that we can control is power term P in order to control T_{ss}.

D. *Determination of Voltage and Frequency Levels*

Once the T_{ss} of a program is obtained corresponding to the program workload and maximum power value and is found to

be above $T_{critical}$ then we must scale down the voltage and frequency in order to reduce the power and in turn reduce T_{ss}. We start from a simple equation for power dissipation per instruction execution in a CPU:

$$P = C \cdot V^2 \cdot f \cdot \alpha \qquad (7)$$

where C is the total capacitance of the CPU obtained by summing all the gate input capacitances, V is the operating voltage, f is the clock frequency, and α is an switching activity factor defined as the expected number of transitions per gate per instruction. Notice that α is dependent on the clock per instruction count of the target application program. The above power equation is only utilized in the remainder of this paper to show that as long as the application program running on a CPU is unchanged, the power dissipation is only dependent on the product of V^2 and f. Clearly there are more accurate CPU power macro models that can capture the effect of program behavior (inter-instruction dependencies, cache access patterns, etc.) but we make this assumption for simplicity.

The previously used P_i is calculated under the processor-given maximum voltage and frequency. Consequently, the resultant thermal curve reaches to T_{ss}. By employing DVFS, the different voltage and frequency levels will generate different power dissipation value smaller than P_i. Plugging the power terms into equation (5), we obtain different T_{ss} values corresponding to each of these power terms. Among these different voltage and frequency levels generating different T_{ss} values, the highest level that satisfies $T_{ss} < T_{critical}$ is the best voltage and frequency since it results in the minimum performance degradation.

$$\begin{aligned}
&\text{Available frequency levels: } f_{\min} = f_1 < f_2 < \ldots f_{n-1} < f_n = f_{\max} \\
&\text{Available voltage levels: } \quad V_{\min} = V_1 < V_2 < \ldots V_{n-1} < V_n = V_{\max} \\
&\forall i \in 1,\ldots,n \quad P_i = C \cdot V_i^2 \cdot f_i \cdot \alpha \Rightarrow \\
&\qquad T_{ss}^{f_i} = P_i \cdot R_{th} = b \cdot V_i^2 \cdot f_i \text{ where } b = C \cdot \alpha \cdot R_{th}
\end{aligned} \qquad (8)$$

Due to workload difference among different application programs, T_{ss} of each of these programs corresponding to (V_{max} ,f_{max}) is different. Therefore each of these programs requires different level of voltage and frequency to bring their T_{ss} below $T_{critical}$. Fig. 4 provides the pictorial example of this point. We reuse the thermal curves of program #3 and #5 from Fig. 2. As shown in bold lines, frequency levels f_1 and f_5 are the best selections for programs #5 and #3, respectively.

Notice from the plot that the higher is the program's workload, the more aggressive voltage and frequency scaling have to be in order to be thermally safe. We define the competitiveness of a DTM strategy, denoted by the *c-factor*, as the degree by which the DTM employs DVFS to control temperature rise/fall on the CPU. A higher c-factor for a DTM strategy means more aggressive thermal management. In particular, we set a c-factor of 1 when minimum voltage and frequency setting must be applied to bring the temperature below $T_{critical}$ (making CPU thermally safe) and a c-factor is zero when no voltage or frequency scaling has to be applied to keep the CPU thermally safe due to low workload.

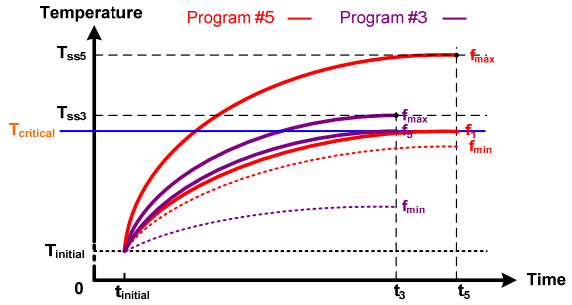

Fig. 4. Example showing DTM c-factor' dependence on steady-state temperatures.

E. Observations about MPEG-2 Decoding

As is widely known, MPEG has 3 different frame types: I, P, and B frames. An I (intra)-frame, which is encoded independent of other types of frames, has a nearly fixed workload and thus takes an almost constant decoding time. In contrast, P (predictive) and B (bi-directional) frames have variable workloads due to the temporal motion variation in a frame sequence resulting in different degrees of compression; hence, they take variable decoding time.

Another descriptor in MPEG is a Group of Pictures (GOP) which is defined as a repeated pattern of I, P, and B frames. Basically, GOP breaks the inter-dependencies of frame sequences in the sense that a frame in a given GOP has no dependency on any frame in other GOPs. Furthermore, although the I-P-B pattern in GOPs in an encoded MPEG stream is not necessarily fixed, in practice this pattern has a fixed format [29], which is determined at the encoding time [30]. In our view, this GOP is at the appropriate level of granularity for the DTM task because i) In an MPEG stream, although the decoding workloads for P and B frame types varies widely (cf. Fig. 5), the workload variation across different GOPs is rather small (cf. Fig. 6); ii) When applying DVFS to control chip temperature, the reliance on a GOP minimizes the change in voltage and frequency settings (which has some overhead) and at the same time allows us to perform slack time borrowing across different frames in a GOP so as to minimize the inter-frame jitter that would arise from greedy and hurried DVFS in response to workload of the current frame.

Fig. 5 reports the B-frame's decoding time variation in a video stream decoded with MediaBench MPEG-2 decoder program [28]. We run the decoder on an Intel Pentium IV 2.8GHz platform (OS is Linux-2.6.15) and report the percentage change in each B-frame's decoding workload with respect to its previous B-frame. The figure depicts data for a sequence of six GOPs selected from an MPEG stream that has a 720x416 resolution. (The simulation methodology will be explained later.) As seen, the B-frame's decoding time fluctuates by a large amount over the frame sequence, which in turn hinders accurate workload prediction on a per-frame basis.

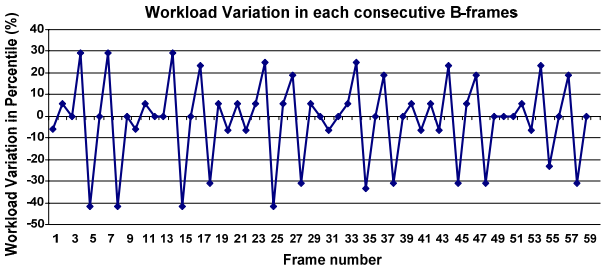

Fig. 5. Workload variation in each consecutive B-frames.

In contrast, Fig. 6 shows the per-frame decoding time of each frame within the same six GOPs. The x-axis, which depicts the invariant GOP structure in this MPEG stream, has a frame sequence of 'IBBPBBPBBPBBPBB'. As shown, per-frame decoding times in each GOP vary; however frames that appear in the same position in all of the GOPs require nearly constant decoding times. We can thus conclude that although per-frame decoding times in a GOP can vary significantly, the decoding times for each frame in a GOP can be precisely estimated after the first few GOP's are analyzed and the pattern and workload of I, P and B frames for the GOP's are recorded. Therefore, GOP-aware workload prediction for frames outperforms the conventional workload prediction for frames (which is typically based on regression estimation of workload for each frame type independent of the GOP boundaries). Once the DVFS policy for frames in a GOP is determined, the same policy will remain in effect for all consecutive GOP's until the workload changes significantly and at that time we will run our DTM again to derive and set a new GOP-level DVFS policy.

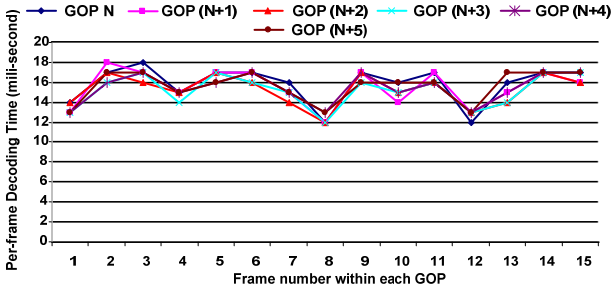

Fig. 6. Decoding time in each consecutive GOPs.

F. Quality Degradation

In soft real-time applications, one simply finds the "best" voltage and frequency that satisfies the thermal constraint while minimizing the total execution time. The possible increase in the task execution time (which is due to the application of DVFS to meet the thermal constraint) is tolerated by the system. In contrast, in hard real-time multi-media applications, there are hard deadlines that must be met by each frame and/or by each GOP. For this reason, some degree of image quality degradation may become necessary in order to meet both the thermal and deadline constraints. To make this statement more precise, we define two types of frame image quality degradations: spatial quality degradation and temporal quality degradation.

Definition: The spatial quality degradation is a measure of how a modified frame differs from the original frame.

Definition: The temporal quality degradation is a measure of how the modified video stream differs from the original video stream. The number of skipped/dropped frames is used as the metric here.

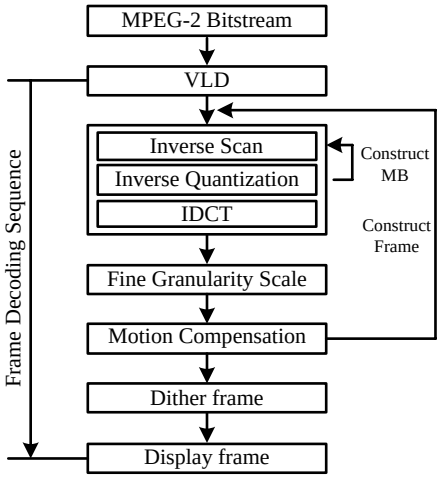

Fig. 7. Typical MPEG-2 decoding steps.

Fig. 7 shows the typical MPEG-2 decoding steps: Variable Length Decoding (VLD), Inverse Quantization (IQ), Motion Compensation (MC), Inverse Discrete Cosine Transformation (IDCT), dithering, display, etc. Among these steps, we look for the ones that minimally distort the frame image quality while maximally reduce the per-frame workload. To this end, we analyzed the relationship between the per-frame workload reduction and a number of candidate quality enhancement methods and ended up with selecting two MPEG-2 features: SNR scalability and the saturation control. SNR scalability enhances the video quality by means of a two-layer structure: a base layer and an enhancement layer. Base layer includes the coarse level DCT coefficients while the enhancement layer includes the finer DCT coefficients. The saturation control clips the results of the IQ. These two fine granularity scalability (FGS) techniques [31] in MPEG-2 were primarily introduced to adjust to the time-varying bandwidth for smooth image quality degradation. From the experiments, we observe that these two steps consume approximately 10% of the total frame decoding times. We use them to perform the spatial quality degradation in our DTM framework. Notice that the execution time reduction due to spatial quality degradation is smaller than the time that is saved due to temporal quality degradation (i.e., dropping a whole frame).

G. DTM Policy

We first define some terms.

N: Number of frames per GOP.

W: Workload of a frame in terms of the number of CPU cycles needed to finish the frame decoding.

***f*:** Operating frequency, chosen from a discrete set of frequencies in the range f_{MIN} to f_{MAX}.

***t*:** Per-frame decoding time which is calculated as follow:

$$t = \frac{W}{f} \tag{9}$$

***D*:** Per-frame decoding deadline (which is determined by the frame rate).

***Δ t*:** Per-frame slack time which is calculated as follows:

$$\Delta t = D - t \tag{10}$$

t_{slack}**:** Per-GOP slack time which is the summation of the *Δ t*'s within a GOP, i.e.,

$$t_{slack} = \sum_{1}^{N} \Delta t \tag{11}$$

***δ*:** Pre-determined time saving by applying spatial quality degradation.

T_{limit}**:** User-defined temperature limit guiding our DTM policy.

1) Problem Formulation and Offline Solution

The problem of assigning frequencies to each frame while meeting both deadline and temperature constraints may be formulated as follows:

$$\text{Minimize } \sum_{i=1}^{N}\left(D - \frac{W_i}{f_i}\right)^2 \quad \text{such that } T_{ss}^{f_i} < T_{limit} \text{ and } \sum_{i=1}^{N}\frac{W_i}{f_i} \le N \cdot D \tag{12}$$

The objective function represents the frame decoding time variance which is known as '*jitter*' in multi-media programs. First, we must ensure that the steady-state temperature under frequency assignment is less than T_{limit}. By assigning different voltage and frequency settings to every frame in a GOP, we try to complete the whole frame decoding workload in the GOP within the given deadline, $D_{GOP} = N \cdot D$.

Unfortunately, only a discrete and finite number of frequencies are available in practice. Moreover, it is possible that severe thermal constraint prohibits any frequency from satisfying the deadline constraint in the GOP. As a consequence, it is not always possible to find the frequency assignments that meet both of the abovementioned constraints. In that case, we have to choose frequencies that violate the deadline (performance) constraint but still meet the temperature constraint, and subsequently compensate for the performance loss by using quality degradation. Due to this limitation and the computational complexity (by using the ILP solver) of solving problem formulation (10) online, we have developed an online heuristic algorithm as described below.

2) Online DTM Algorithm

The DTM policy presented in this section performs energy efficient frequency (and subsequently voltage) assignment while meeting the thermal constraints. If the performance constraint cannot be met by the frequency scaling, then we resort to quality degradation. Before explaining the proposed DTM solution, we examine the relationship between frequency-dependent deadline and the thermal constraint for a frame in Fig. 8. Notice that although we actually scale both voltage and frequency, we only show the frequency term in this figure. When the available frequency range is given as f_{MIN} to f_{MAX}, $f_{deadline}^{min}$ is the minimum frequency that meets the per-frame decoding the deadline constraint while $f_{thermal}^{max}$ is the maximum frequency that does not violate the thermal constraint, i.e., $T_{ss} < T_{limit}$. The dotted lines in Fig. 8 imply that these two limits are different in every frame within the GOP. Basically, the proposed DTM policy decides the degree of thermal management depending on the relative positions of these two parameters in each frame. The possible frame types categorized based on these two parameters are as follows:

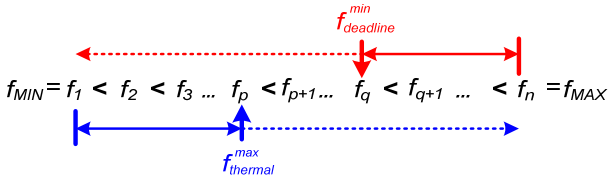

Fig. 8. Deadline and thermal constraints in frame decoding.

Frame type I (Non-negative slack generator: $f_{deadline}^{min} \le f_{thermal}^{max}$): In this type of frame, multiple frequencies meet both temperature and performance constraints for the frame decoding. Among the multiple frequencies, we initially select $f_{deadline}^{min}$ but we have the freedom to pick any frequency up to (and including) $f_{thermal}^{max}$ and we use these choices in case we need to compensate for any performance degradation due to frequency assignment to other frames in the GOP. The initial selection of $f_{deadline}^{min}$ and the following frequency increase is energy efficient since we always try to use minimum frequency (and corresponding voltage) while meeting the performance constraint. In the flow chart, the '***AFS (Available Frequency Set)***' corresponds to the set of frequencies equal to or higher than $f_{deadline}^{min}$ but less than or equal to $f_{thermal}^{max}$ for this type of frame. Since each frame has different workload, different ranges of '***AFS***' exist for each frame.

Frame type II (Negative slack generator: $f_{thermal}^{max} < f_{deadline}^{min}$): In this type of frame, none of the frequencies meet both constraints in the frame. From the flow chart, we initially select $f_{deadline}^{min}$ for this type of frame; however, we will eventually switch to $f_{thermal}^{max}$ to meet the thermal constraint. So this type of frame will result in a frame deadline violation, but not necessarily a GOP deadline violation. If the latter occurs (because for example when we have many type II frames in a GOP and few type I ones), then we must sacrifice video quality through spatial/temporal quality degradation methods.

These two types of frames co-exist in each GOP and together generate a t_{slack} for the whole GOP, which in turn determines the aggressiveness factor of DTM, and the degree of spatial and temporal quality degradation in each GOP.

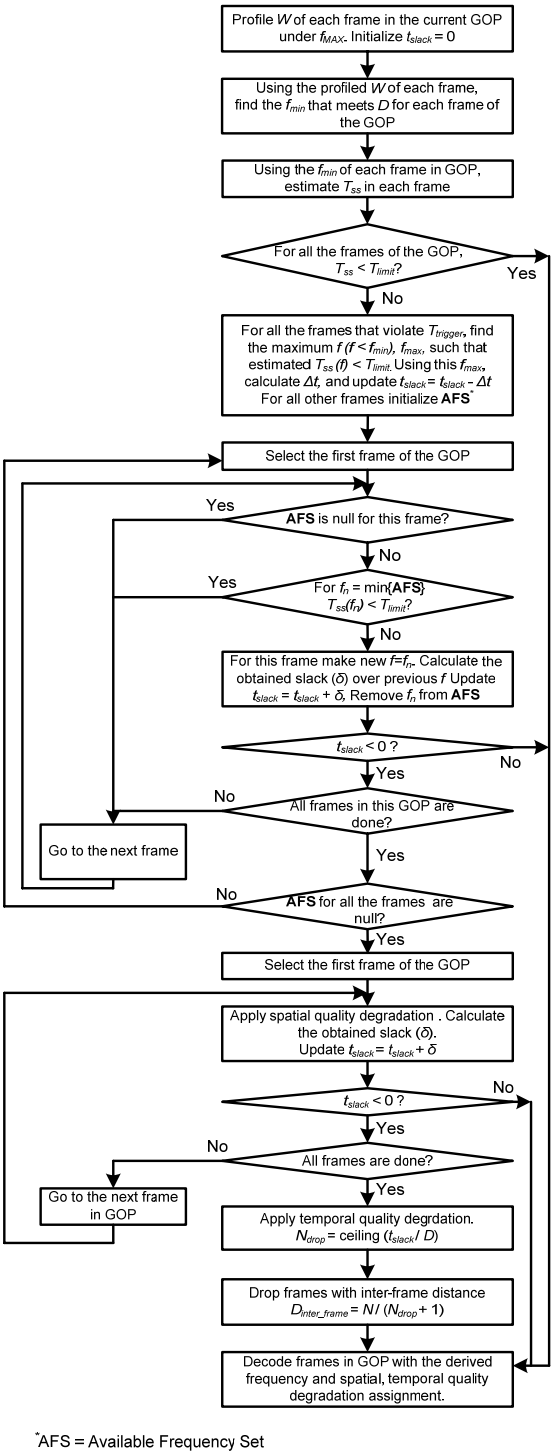

Fig. 9. Online algorithm for computing GOP-level DTM policy.

Fig. 9 shows the proposed GOP-level DTM policy. As shown in this flow chart, we employ DVFS, spatial quality degradation, and temporal quality degradation in that order. Note that the policy depicted in Fig. 9 takes the exact workload of each frame in a GOP as an input and outputs the frequency/voltage and spatial/temporal quality degradation assignments for each frame in the GOP. As long as the workload of the GOP remains relatively unchanged, the same voltage/frequency assignments are in effect. However, if the GOP changes (more precisely, if the GOP workload changes by more than 5% with respect to the workload of the last GOP based upon which the GOP-level policy was determined), then we will re-compute the voltage / frequency assignments by running the GOP-level policy derivation algorithm of Fig. 9.

IV. Experimental Results

A. Methodology

For the experiments, we integrated Simplescalar [32], Wattch [33], and HotSpot into one package. The Simplescalar serves as the architectural simulator, Wattch generates power figures for each functional block, while HotSpot generates temperature information with its thermal model parameters. The simulated micro-processor model is based on Alpha 21364, which has a minimum feature size of 0.18µ, nominal supply voltage of 1.6V, and a nominal clock speed of 1GHz. The Wattch power model does not account for the leakage power. The simulated supply voltages have a range from 0.8V to 1.8V in steps of 0.2V, and clock frequencies have a range from 600MHz to 1.2GHz in steps of 100MHz. We developed a simulation environment similar to that in [23]. The ambient and the initial temperatures were set 40.0℃ while we experimented with different maximally allowed temperature limits (denoted as T_{limit}) in order to show the effectiveness of the proposed DTM method. The combined thermal simulator generates temperature results for each functional block every 10K cycles. Our experimental results are based on temperatures in the hottest functional block.

TABLE 1 summarizes the architectural parameters that were used in our simulations.

TABLE 1
BASLINE CONFIGURATOIN OF SIMULATED PROCESSOR

Memory Latency	100 cycles/10 cycles
L1 I/D Cache	64KB 2-way 32Byte block 1 cycle hit latency
I/D-TLB	Fully associate, 128 entries 30 cycles miss latency
Branch Predictor	4K Bimodal
Functional Units	4 INT ALU,1 INT MULT/DIV, 2 FP ALU, 1 FP MULT/DIV
RUU/LSQ size	64/32
Inst. Fetch Queue	8
Issue Width	6 instruction per cycles

For the application programs, we use the MPEG-2 decoder program of MediaBench benchmark suite [28] while considering a frame rate of 60 frames per second. TABLE 2 summarizes the MPEG-2 files used in the experiments, which are obtained from [34]. Moreover, several custom-made MPEG-2 files are used.

B. Simulation Results

TABLE 3 shows the temperature reduction and the resulting

spatial/temporal quality degradation with the proposed online DTM algorithm. We set different T_{limit}, namely 65℃, 60℃, 55℃ and 50℃ and show how the proposed DTM policy works for the each of MPEG files. Clearly, spatial/temporal quality degradation increases as T_{limit} decreases. Similarly the degradation is higher for video files that have higher workload, i.e., those with higher resolution. Files that have relatively small workload are not affected by the proposed DTM at all.

TABLE 2
MPEG-2 FILES USED IN THE EXPERIMENT

Files	Frames	Frame resolution	GOP Structure
mei60f.m2v	48	704 x 480	$IB^2PB^2PB^2PB^2$
hhilong.m2v	36	720 x 576	$IB^3PB^3PB^3PB^3$
Jurassic Park	105	720 x 416	$IB^2PB^2PB^2PB^2PB^2$
time_015.m2v	36	704 x 480	$IB^2PB^2PB^2PB^2$
Dead poet's Society	435	640 x 352	$IB^2PB^2PB^2PB^2PB^2$
Finding Nemo	420	560 x 352	$IB^2PB^2PB^2PB^2PB^2$
soccer_015.m2v	60	640 x 480	$IB^2PB^2PB^2PB^2PB^2$
Monsters Inc.	75	576 x 320	$IB^2PB^2PB^2PB^2PB^2$
Back to the Future	300	352 x 256	$IB^2PB^2PB^2PB^2PB^2$
cact_015.m2v	240	352 x 192	$IB^2PB^2PB^2PB^2$
tens_015.m2v	168	352 x 192	$IB^2PB^2PB^2PB^2$
Incredible	390	352 x 240	$IB^2PB^2PB^2PB^2PB^2$

Fig. 10 shows the relationship between the average per-frame decoding time (workload) as a percentage of the given frame decoding deadline D (cf. y-axis on the left), and the degree of average quality degradation per-frame (cf. y-axis on the right). The bar graph corresponds to the workload in percentile and the line graphs correspond to the combined quality degradation. The combined quality degradation is generated by applying different weighting factors between the spatial and the temporal quality degradations. This figure basically discloses a number of facts: i) across the programs, the degree of quality degradation becomes more severe as the program's workload becomes higher. ii) As T_{limit} gets lower in the programs, higher degrees of quality degradation are needed to meet both deadline and thermal constraints. iii) For programs whose workload is above 100%, e.g., *mei60f*, the combined quality degradation will never reach 0%. iv) For programs whose workload is less than 100%, e.g., *time_015*, (which implies that they naturally meet the performance constraint) the combined quality degradation may be higher than 0% since the thermal constraints can become a bottleneck.

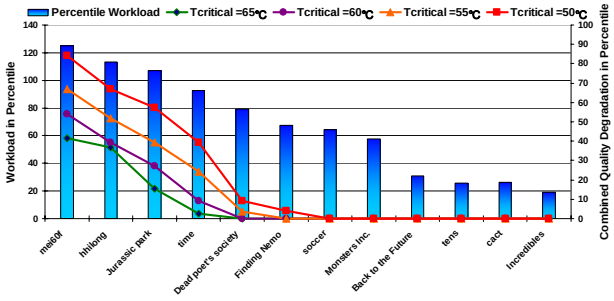

Fig. 10. Relation between the workload and the quality degradation.

Fig. 11 shows the transient thermal behavior during a part of the simulation for three MPEG-2 files: "*Jurassic park*", "*Dead poet's society*", and "*Back to the future*". The X-axis is the frame number within GOP while the Y-axis is the temperature in Celsius. Temperature measurements are performed every 10K cycles. For each MPEG-2 file, we report two simulation curves: 1) without and 2) with the proposed DTM method under a T_{limit} value of 65℃. We chose the three video files to capture the range of possible workloads: *Jurassic park* (red curve) for heavy workload, *Dead poet's society* (green curve) for medium workload, and *Back to the future* (light blue curve) for light workload. For each MPEG-2 file, our DTM method is shown in dark blue, orange, and violet curves (cf. the quasi flat

TABLE 3
ACHIEVED TEMPERATURES AND THE CORRESPONDING QUALITY DEGRADATION (IN PERCENTILE) IN DIFFERENT T_{limit}

Input file	T_{limit} = 65℃			T_{limit} = 60℃			T_{limit} = 55℃			T_{limit} = 50℃		
	T_{ss}	Spatial	Temporal	T_{ss}	Spatial	Temporal	T_{ss}	Spatial	Temporal	T_{ss}	Spatial	Temporal
mei60f.m2v	63.26	**91.6**	**36.1**	56.84	**91.6**	**50.0**	51.36	**91.6**	**63.8**	47.21	**91.6**	**83.3**
hhilong.m2v	61.81	**91.6**	**30.5**	56.76	**91.6**	**33.3**	51.29	**91.6**	**47.2**	47.17	**91.6**	**63.8**
Jurassic Park	63.37	**93.3**	**5.7**	56.84	**93.3**	**17.1**	51.34	**93.3**	**28.6**	47.21	**93.3**	**45.7**
time_015.m2v	63.26	**16.7**	**0**	56.85	**61.1**	**0**	51.35	**61.1**	**11.1**	47.21	**61.1**	**22.2**
Dead poet's Society	58.45	0	0	55.53	0	0	51.34	**20.2**	**0**	47.21	**93.1**	**0**
Finding Nemo	52.41	0	0	52.41	0	0	50.34	0	0	47.21	**44.3**	**0**
soccer_015.m2v	49.78	0	0	49.78	0	0	49.61	0	0	47.02	0	0
Monsters Inc.	47.41	0	0	47.41	0	0	47.41	0	0	46.51	0	0
Back to the Future	43.71	0	0	43.71	0	0	43.71	0	0	43.71	0	0
cact_015.m2v	43.71	0	0	43.71	0	0	43.71	0	0	43.71	0	0
tens_015.m2v	43.71	0	0	43.71	0	0	43.71	0	0	43.71	0	0
Incredible	43.71	0	0	43.71	0	0	43.71	0	0	43.71	0	0

curves at 63℃, 58℃ and 48℃), respectively. This figure, together with the quality degradation results in TABLE 3, reveal the following: i) Files with heavy workload barely meet the thermal constraint but the quality degradation is high. ii) Files with medium workload meet the thermal constraint with a good margin and exhibit little or no quality degradation. In this case, thermal curve far below T_{limit} implies that a non-maximum frequency in ***AFS*** was selected to minimize energy dissipation after meeting the thermal constraint. iii) Files with light workload have no concern about the thermal issues, yet DVFS shifts the original thermal curve down.

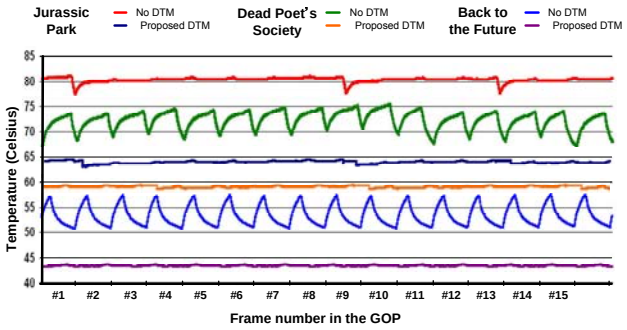

Fig. 11. Comparison of thermal curves in three MPEG-2 files.

V. Conclusion

We presented a GOP-level dynamic thermal management (DTM) algorithm based on accurate estimation of per-frame workload within a GOP and slack time borrowing across the frames to achieve a thermally safe state of operation in microprocessors during MPEG-2 decoding. To this end, the proposed DTM method achieves thermally safe state of operation in microprocessors by applying DVFS and allows spatial and temporal quality degradation in order to compensate for the loss of performance. The degree by which the proposed DTM employs DVFS to control temperature rise/fall in the microprocessor is determined by per-frame based GOP workload and temperature constraints. While meeting the temperature and performance constraints, voltage and frequency assignments in our DTM policy guarantee the minimal energy consumption in the microprocessor. Our experimental results demonstrated the effectiveness of the proposed online DTM algorithm.

References

[1] M. Pedram, and S. Nazarian, "Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods," *Proc. of the IEEE*, vol. 94, no. 8, Aug. 2006.

[2] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, "Managing the Impact of Increasing Microprocessor Power Consumption," In *Intel Technology Journal*, 2001.

[3] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," *Proc. of the Int'l Symposium on High-Performance Computer Architecture*, pp. 171, Jan. 2001.

[4] S-M. Heo, K. Barr, and K. Asanovic, "Reducing Power Density through Activity Migration," *Proc. of the Int'l Symposium on Low Power Electronics and Design*, pp. 217-222, Aug. 2003.

[5] J. Srinivasan and S. V. Adve, "Predictive Dynamic Thermal Management for Multimedia Applications," *Proc. of the Int'l Conf. on Supercomputing*, pp. 109-120, 2003.

[6] A. Kumar, L. Shang, L-S. Peh, and N. K. Jha, "HybDTM: A Coordinated Hardware-Software Approach for Dynamic Thermal Management," *Proc. of the Design Automation Conf.*, pp. 548-553, Jul. 2006.

[7] K. Skadron, "Hybrid Architectural Dynamic Thermal Management," *Proc. of the Design Automation and Test in Europe*, pp. 10-15, Feb. 2004.

[8] B. Sprunt, "Pentium 4 Performance-Monitoring Features," *Proc. of the Int'l Symposium on Microarchitecture*, vol. 22, no. 4, pp. 72-82, Jul./Aug. 2002.

[9] F. Bellosa, S. Kellner, M. Waitz, and A. Weissel, "Event-Driven Energy Accounting for Dynamic Thermal Management," In the *Workshop on Compilers and Operating Systems for Low Power*, Sep. 2003.

[10] A. Weissel and F. Bellosa, "Dynamic Thermal Management for Distributed Systems," In the *Workshop on Temperature-Aware Computer Systems*, 2004.

[11] K-J. Lee and K. Skadron, "Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors," In the *Workshop on High-Performance, Power-Aware Computing*, pp. 232, Apr. 2005.

[12] M. R. Stan, K. Skadron, M. Barcella, W. Huang, K. Sankaranarayanan, and S. Velusamy, "HotSpot: A Dynamic Compact Thermal Model at the Processor-Architecture Level," *Proc. of the Microelectronics Journal: Circuits and Systems*, vol. 34, no. 12, pp. 1153-1165, Dec. 2003.

[13] W. Huang, Ghosh. S., Velusamy S., Sankaranarayanan K., K. Skadron, and M. R. Stan, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 14, no. 5, pp. 501-513, May. 2006.

[14] S-W. Chung and K. Skadron, "Using On-chip Event Counters for High Resolution, Real-time Temperature Measurement," *Proc. of the 10th Intersociety Conf. on thermal and Thermo-mechanical Phenomena in Electronics Systems*, pp. 114-120, Jun. 2006.

[15] S-W. Chung, K. Skadron, "A Novel Software Solution for Localized Thermal Problems," University of Virginia *Tech Report:* CS-2006-10, Apr. 2006.

[16] K. Sankaranarayanan, S. Velusamy, M. R. Stan, and K. Skadron, "A Case for Thermal-Aware Floor-planning at the Micro-architectural Level," *Proc. of the Journal of Instruction Level Parallelism*, Jun. 2005.

[17] Y Han, I. Koren, and C. A. Moritz, "Temperature Aware Floor-planning," In the *Workshop on Temperature-Aware Computer Systems*, 2005.

[18] M. Healy, M. Vittes, M. Ekpanyapong, C. Ballapuram, S. K. Lim, H. Hsin S. Lee, and G. H. Loh, "Micro-architectural Floor-planning under Performance and Thermal Tradeoff," *Proc. of the Design Automation and Test in Europe*, pp. 1288-1293, Mar. 2006.

[19] V. Nookala, D. J. Lilja, and S. S. Sapatnekar, "Temperature-Aware Floor-planning of Microarchitecture Blocks with IPC-Power Dependence Modeling and Transient Analysis," *Proc. of the Int'l Symposium on Low Power Electronics and Design*, pp. 298-303, Oct. 2006.

[20] C. H. Lim, W. R. Daasch, and G. Cai, "A Thermal-Aware Superscalar Microprocessor," *Proc. of the Int'l Symposium on Quality Electronic Design*, pp. 517-522, 2002.

[21] E. Carson, G. Reinman, S. Sair, A. Shayesteh, and T. Sherwood, "Low-Overhead Core Swapping for Thermal Management," In the *Workshop on Power Aware Computer Systems*, Dec. 2004.

[22] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," *Proc. of the Int'l Symposium on High-Performance Computer Architecture*, pp. 17-28, Feb. 2002.

[23] K. Skadron, M. R. Stan, W. Huang, and S. Velusamy, "Temperature-Aware Micro-architecture," *Proc. of the Int'l Symposium on Computer Architecture*, pp. 2, Jun. 2003.

[24] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-Aware Micro-architecture: Modeling and Implementation," *Proc. of the ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94-125, Mar. 2004.

[25] D. Son, C. Yu, and H. Kim, "Dynamic Voltage Scaling on MPEG Decoding," *Proc. of the Int'l Conf. on Parallel and Distributed Systems*, pp. 633-640, 2001.

[26] K-H. Choi, K. Dantu, W-C. Cheng, and M. Pedram, "Frame Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder," *Proc. of the Int'l Conf. on Computer Aided Design*, pp. 732-737, Nov. 2002.
[27] K-H. Choi, R. Soma, and M. Pedram, "Off-chip Latency Driven Dynamic Voltage and Frequency Scaling for and MPEG Decoding," *Proc. of the Design Automation Conf*., pp. 544-549, Jun. 2004.
[28] Mediabench at: http://euler.slu.edu/~fritts/mediabench
[29] Y. Yokoyama, "Adaptive GOP Structure Selection for Real-time MPEG-2 Video Encoding," *Proc. of the Int'l Conf. on Image Processing*, pp. 832-835, 2000.
[30] H. Wu, M. Claypool, and R. Kinicki, "Guidelines for Selecting Practical MPEG Group of Pictures," *Proc. of the IASTED Int'l Conf. on Internet and Multimedia Systems and Applications*, pp. 61-66, 2006.
[31] MPEG-2 Standard: International Organization for Standardization/International Electro-technical Commission (ISO/IEC) 13818-2.
[32] Simplescalar tutorial at http://www.simplescalar.com
[33] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proc. of the Int'l Symposium on Computer Architecture*, pp. 83-94, Jun. 2000.
[34] MPEG-2 Streams at http://www.mpeg2.de/video/streams