Effective Dynamic Thermal Management for MPEG-4 Decoding

Inchoon Yeo, Heung Ki Lee, Eun Jung Kim
Department of Computer Science
Texas A&M University
College Station, TX 77840
{ryanyeo, hklee, ejkim}@cs.tamu.edu

Ki Hwan Yum
Department of Computer Science
University of Texas at San Antonio, TX 78249
yum@cs.utsa.edu

Abstract

This paper proposes Dynamic Thermal Management (DTM) based on a dynamic voltage and frequency scaling (DVFS) technique for MPEG-4 decoding to guarantee thermal safety while maintaining a quality of service (QoS) constraint. Although many low-power and low-temperature multimedia playback techniques have been proposed, most of them are impractical in real-time and have several restricting assumptions. Multimedia data consists of several frames requiring different decoding efforts. Since both temperature and performance of a multimedia system are affected by the complexity of scenes, our main idea is to use the information on scene complexity to find an appropriate frequency. In order to predict the complexity of the current scene, we extract information from the previous group of pictures (GOP) using feedback control with a display buffer. Experimental results with twelve movies show that our DTM scheme guarantees the threshold of temperature (70°C) while maintaining 0% frame miss ratio. Also, our DTM scheme decreases the average temperature by up to 13% without any additional hardware and playback latency.

1 Introduction

Thermal issues are becoming critical in multimedia systems to achieve high reliability. Although the speed of a modern microprocessor supports processing of the multimedia data in real-time, a multimedia system consumes lots of power for computation and cooling. General purpose computer systems consume over 25% of the power for energy management such as air conditioning, backup cooling and power delivery systems [3]. However, portable battery-operated devices cannot afford such high cooling power. Without sufficient cooling, embedded systems suffer from long-time overheating and eventually cause the system to crash. However, reducing the voltage level causes the overall performance to slow down. Therefore, the best solution to reduce energy dissipation with dynamic voltage and frequency scaling (DVFS) techniques is to dynamically adjust the voltage scales while maintaining the minimum required circuitry to accommodate workloads within appropriate computation time and throughput constraints [4]. Multimedia data consist of different frames with different deadlines to be displayed. MPEG frames are classified into three different coding types including intra (I), predictive (P), and bi-directional (B) that consume various power/energy, which leads to raise a different amount of temperature during decoding frames. In addition, a wide variety of frame sizes make it difficult to predict power consumption and to control temperature. Furthermore, since DVFS reduces the overall computation speed, it is likely to have some frames missing their deadlines. Therefore, it is challenging to find a right speed to control system temperature without quality degradation. In [8, 10], authors suggested a feedback control from a display buffer to find an adequate speed without quality degradation and to reduce the power consumption of MPEG decoding. However, they did not concern thermal problems and used only the buffer occupancy information that is not sufficient to control the speed for both performance and temperature. We observe that the decoding time required depends on the complexity of scenes that can be measured with the number of frames in a group of pictures (GOP), and that frames in a GOP require similar computation time for decoding. Therefore, we suggest using the previous GOP information to predict the computation power of a current frame. We propose an efficient Dynamic Thermal Management (DTM) scheme for a multimedia system to find an appropriate frequency for the available decoding and display buffer based on an advanced feedback control. Our scheme uses the information of the previous GOP considering the trade-offs between the quality of data and thermal safety using a frequency efficient factor.

1-4244-1258-7/07/$25.00 ©2007 IEEE

2 Related Work

Several schemes using architecture adaptation provided DTM solutions [2, 5]. Brooks, *et al.* suggested the fetch toggling to avoid thermal limit using the stall of instruction fetching [2]. Heo, *et al.* transformed the fetched computation into other duplicated unit during cooling down the overheated unit [5]. However, these schemes cannot satisfy the deadline of workload in real-time. Especially, the missed deadline results in low performance in the multimedia system.

Skadron, *et al.* suggested several schemes for the thermal management including temperature-tracking [11], hybrid scheme [9], and feedback control [10, 8]. In [11], temperature-tracking scheme manages temperature based on frequency scaling, localized toggling, and computation migration. In [9], they proposed a hybrid scheme between fetch gating and DVFS. Also, a feedback control configures the temperature based on feedback information [10, 8]. Since the method suggested by Skadron, *et al.* does not take into account the complexity of scenes for multimedia, it cannot avoid the degradation of performance in multimedia applications with radical picture changes.

In [12], Mircea, *et al.* designed the thermal model using the thermal behavior, thermal resistances, and capacitances within functional blocks at the architectural level. Also their model is implemented and widely used in the temperature research. Even though the temperature behavior can be detected in real-time, the specific hardware like the built-in performance monitoring unit (PMU) is required[7].

3 OVERVIEW OF A FEEDBACK CONTROL

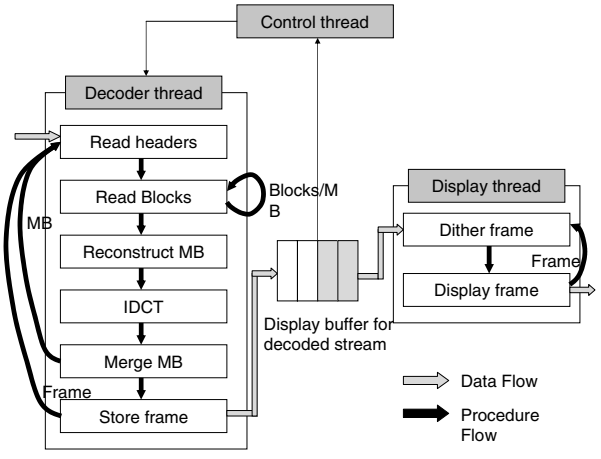


Figure 1. MPEG Process with Display Buffers.

Figure 1 shows the details of decoding procedures used in [10, 8]. After reading a stream in 'Read headers' and 'Read Blocks' steps, a decoder thread decodes macro blocks at 'Reconstruct MD', 'IDCT' and 'Merge MB.' Finally, the decoder thread puts the decoded frame into a buffer for display. Then a display thread performs the steps of 'Dither' and 'Display' with the frames in the display buffer. The decoder thread executes processor-intensive operations, while the display thread just displays the decoded frames. To obtain an adequate frequency for the decoding stream, a control thread checks the state of the display buffer. With high occupancy in the display buffer, the control thread decreases the frequency, since the decoding elapsed time with the current frequency is too much faster than the display elapsed time. On the contrary, low occupancy lets the control thread increase the frequency to meet the deadline of each frame. Therefore, the performance (i.e., the deadline of frames) should be considered in low occupancy, while energy efficiency and thermal safety also should be considered in the high occupancy of the display buffer. To fulfill these two considerations, the control thread has to determine an optimal frequency without QoS degradation. Frames should be decoded sequentially and displayed on the display device with a constant playback interval denoted by $t_{interval}$. Although each frame can be decoded at a different elapsed time due to computation variations, each decoded frame in the display buffer are displayed at a uniform speed. To support the QoS requirement, the adjusted frequency should satisfy the following Eq. (1)[8]:

$$\frac{\sum_{k=1}^{i} D_k}{i} \leq t_{interval}, 1 \leq i \leq n, \quad (1)$$

where D_k is the decoding time for frame k. i means the number of frames in the display buffer and n is the size of the display buffer. Note that for the consecutive frame, we do not have any information about the required decoding time. Without the display buffer, it is difficult to estimate the optimal frequency for decoding the frame. Also, the display buffer with enough space for several frames can make a system determine the optimal frequency without frame misses. Lu, *et al.* uses a feedback controller to adjust the frequency with the number of decoded frames in the display buffer within a region specified by $\{B_l, B_h\}$, where B_l is the lower threshold for the number of frames in the buffer and B_h is the higher threshold [8]. Using the feedback controller for decoding frames in the display buffer assumes that the decoder speed is adequate for decoding the frames as long as the number of the frames in the display buffer is within the specified region. However, there are two serious problems in the feedback controller with the display buffer. The first problem is that the feedback controller using the display buffer does not satisfy the deadline of all

frames. The frequency is adjusted by the value based on the number of the frames in the display buffer within a region specified by $\{B_l, B_h\}$. This problem occurs with the movies containing several complicated scenes, such as Star Wars 3 and Terminator 3. For example, a high frequency may be required even though the occupancy of the display buffer seems to be sufficient to decode upcoming frames. In such cases, frames will be dropped if the the optimal frequency is only derived from the display buffer occupancy. The second problem is that the feedback controller using the display buffer cannot control the temperature to guarantee thermal safety. Without considering temperature constraints, the display buffer decides the optimal frequency using only its occupancy. This is a very critical problem in embedded systems, since most embedded systems do not have cooling systems such as a fan.

4 ADVANCED FEEDBACK CONTROLLER WITH THERMAL SAFETY

In this section, we introduce advanced feedback control schemes using a display buffer and DVFS. Since every frame has a deadline for decoding and displaying, the frequency should be adjusted depending on the computational complexity of a scene as well as each frame's decoding size.

4.1 Advanced Feedback Controller using GOP Information

The relationship among the decoding time, the display interval and the occupancy of the display buffer is defined in Eq. (2). Let D_i be a decoding time of $frame_i$. The decoding time D_i should be finished before all previous frames in the display buffer will be presented. Otherwise, $frame_i$ will miss its deadline. Therefore,

$$D_i < n \cdot t_{interval}, \tag{2}$$

where $t_{interval}$ is the periodic display time in the display buffer and n is the occupancy of the display buffer.

In Figure 2, the number of frames in a GOP decreases when pictures of scenes change rapidly. A single GOP has several frames which consists of I, B and P frames. And for more complex scenes, the number of B and P frames decreases while only a single I frame is allowed for any GOP. Since the display interval time depends on frames per second (fps) and has a value between 25 *msec* and 30 *msec*, we can calculate D_i of the frame that exceeds the display interval time, $t_{interval}$, in this situation. Therefore, the scene complexity can be estimated by adding these D_i values in a GOP.

For example, let's assume that frame f_n should be displayed at time t_n and f_{n+1} should be displayed at t_{n+1}.

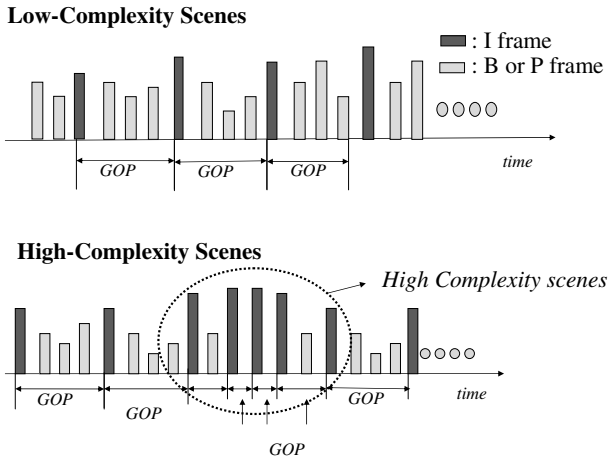


Figure 2. Low-Complexity vs. High-Complexity Scenes

When frame f_n is ready to be displayed, there are four frames from f_{n-3} to f_n in the display buffer at time t_n. Also assume that a decoding time, D_{n+1} of f_{n+1} and D_{n+2} of f_{n+2} takes 3 times more than $t_{interval}$. Under these circumstances, all frames ($f_{n-3} \sim f_{n+1}$) in the display buffer are displayed and the buffer will be empty since the next frame (f_{n+2}) has not been decoded. In this case, the frame f_{n+2} is dropped. In order to avoid the future frames drop, the optimal frequency of f_{n+2} should be determined at time time t_n. However, it is difficult to estimate the frequency for a future frame.

We propose a prediction scheme to find the adequate frequency using the information on decoding the previous GOP. According to our experiments, the frame decoding time depends on the complexity of scenes, which continues to exist in several consecutive frames. It means that frames in each GOP have similar complexity of scenes. Therefore, since the GOP consists of several frames, we can predict the optimal frequency of the current GOP using the information on the complexity of scenes in the previous GOP. If the complexity of the previous GOP is high, the complexity of the current GOP will be also high. Therefore, the complexity of the previous GOP can be used as a weight factor to determine the frequency of the current GOP. The weight factor (α) is calculated as follows:

$$\alpha = \frac{\sum_{i=1}^{k} X_i D_i}{\sum_{i=1}^{k} D_i}, 1 \le i \le k, \tag{3}$$

where D_i is the decoding time of the $frame_i$ and X_i is the indicator which is 1 or 0. With α, the new frequency can be calculated as in Eq. (4). Let $freq_i$ be the frequency of the

decoding time of the $frame_i$. The frequency of the current frame should be configured based on the number of previous frames which have taken less time than the threshold time for displaying that frame.

$$freq_i = (1 - \alpha) \cdot freq_{buf(i-1)} + \alpha \cdot freq_{\max}, \quad (4)$$

where $freq_{buf(i-1)}$ is the frequency value to be calculated by the feedback control based on the occupancy of the display buffer. $freq_{max}$ is the maximum frequency of the processor. Hence, the complexity of scenes can be estimated by the weighting factor, α. For example, if we assume that the previous GOP has twelve frames, and three frames among them have larger decoding time than the selected threshold value, α is calculated as 0.25. It implies that 75% of decoding time of frames in the previous GOP is decoded within threshold and 25% decoding time of frames exceeds the threshold. According to Eq. (4), the frequency of the current frame is adjusted to handle the frames with higher complexity based on the occupancy of the display buffer. Therefore, the frequency for decoding the current frame is selected by the information of the occupancy of the display buffer and the information of previous GOP. The higher the complexity of the previous GOP, the higher the frequency of the current frame. In this model, using this scheme, we can avoid the missed frames when the complexity of scenes is increased suddenly.

4.2 Thermal Control using GOP Information

Although many studies has been focused on the relationship between frequency and power consumption, the relationship between frequency and temperature has to be formulated to find out the optimal frequency within thermal safety. Therefore, we consider a simple thermal model of the processor [1, 6] in that the relation between processor frequency and temperature is the basis for any frequency scaling scheme. By modeling the power dissipation or by increasing the input power, more precise models can be derived from this simple model [13].

We analyze Fourier's Law of heat conduction where the rate of heating or cooling is proportional to the difference in temperature between the object and the environment [13]. We define $T(t)$ and $P(t)$ as the temperature and the power at time t, respectively. Then we can use the Fourier's Law as the following [1, 6]:

$$T'(t) = P(t) - bT(t), \quad (5)$$

where b is a positive constant representing the power dissipation rate. Now, we define $freq(t)$ as the processor frequency at time t. [1] shows that the power consumption of a processor is an increasing convex function of the frequency. Most work assume that power and processor frequency are relevant as follows [1]:

$$P(t) = a(freq^{\gamma}(t)) \quad (6)$$

for some constants a and $\gamma > 1$. Assuming γ=3.0, we can obtain the thermal parameter values for a and b. The values of a and b, are processor-specific but application-independent constants. Also, we can determine the thermal parameters while observing the heating and cooling curves when we run an application which fully occupies the processor. After a long-time execution of the application, the infinite steady-state temperature value $T(\infty) = T_s$ can be observed. Setting $T(t) = T$ and $a(freq_c^{\gamma})/b = T_s$, Eq. (5) and (6) is transformed as follows:

$$T = T_s + (T_{init} - T_s)e^{-bt}, \quad (7)$$

where T_{init} is the initial temperature. From Eq. (7), the slope of this straight line represents the value of b. We obtain b = 0.016. By applying this value b to the relation, the value a is also obtained as a = $3.0E - 28$. With these environmental parameters, we see the effects and decrease in temperature by the suggested DTM for MPEG-4 decoding. Although the temperature variation by the $1.0GHz$ static frequency decoding is shown for the comparison, it is no good in terms of frame miss rate. We show the comparison of the frame miss rate by each method in later section.

4.3 DTM with the Advanced Feedback Controller

To maintain the temperature under the thermal safety, we should use a DTM scheme for the multimedia decoder. The new DTM scheme uses the accurate frequency from the previous GOP frequency. In our scheme, we decide the threshold of temperature to control the overall temperature during a decoder running. In order to decide the temperature threshold, we need the occupancy of the display buffer which can indicate the efficiency of the frequency for decoding the previous GOP.

$$e = \frac{\sum_{i=1}^{n} D_i}{n \cdot t_{\mathrm{interval}}}, 0 < e \leq 1, \quad (8)$$

where $T_{emergency}$ is the maximum allowable temperature and is defined as $80°\mathrm{C}$ in our experiments. And $T_{threshold}$ is the software threshold of temperature during decoding MPEG-4 stream. Therefore, ΔT can be defined as the difference between an emergency temperature and software temperature threshold. The software temperature threshold is the factor that guarantees thermal safety. n is the total number of frames in the display buffer and $t_{interval}$ is the

Algorithm 1 DTM algorithm

Require: Define $Table[\ ]$ for frequency according to threshold temperature

1: Determine a threshold temperature($T_{threshold}$).
2: $i \leftarrow GOP_i$
3: **for** $i = 1$ to GOP_{max} **do**
4: Calculate e in GOP_{i-1}
5: Estimate a current temperature($T_{current}$).
6: **if** $T_{current} > T_{threshold}$ **then**
7: $\Delta T \leftarrow T_{emergency}$ - $T_{threshold}$
8: $T'_{threshold} \leftarrow T_{threshold}$ + (1-e)·ΔT
9: $index \leftarrow index$ + 1
10: $freq_{max} \leftarrow Table[index]$
11: $freq_i \leftarrow$ (1-α)·$freq_{i-1}$ + α·$freq_{max}$
12: **else if** $T_{current} < (T_{threshold}$ - $MIN)$ **then**
13: $index \leftarrow index$ - 1
14: $freq_{max} \leftarrow Table[index]$
15: $freq_i \leftarrow$ (1-α)·$freq_{i-1}$ + α·$freq_{max}$
16: **end if**
17: **end for**

period of displaying the frames. The e is the frequency efficient factor for decoding frames in the previous GOP only when D_i is equal to or less than $t_{interval}$. With the factor e, we decide the new software threshold of temperature as shown in Algorithm (1). If $T_{current}$ exceeds $T_{threshold}$, $T'_{threshold}$ replaces $T_{threshold}$, and $freq_{max}$ is replaced by $freq[T'_{threshold}]$. Therefore, $freq_i$ is determined to maintain the temperature under the thermal safety with Eq. (4) because the determined $freq_{max}$ is smaller than the previous $freq_{max}$.

For example, assuming $T_{emergency}$ to be $80°$C and $T_{threshold}$ to be $60°$C, ΔT is calculated as $20°$C. If e is 0.75 and the current temperature is over the current software temperature threshold, the new software temperature threshold can be adjusted to $64°$C. Consequently, $freq_{max}$ is decreased to the next low frequency by $T_{threshold}$. In this example, $freq_{max}$ is adjusted from 1600 Mhz to 1400 Mhz when $T_{threshold}$ is changed. This new software temperature threshold makes $freq_{max}$ decrease the overall temperature. With this scheme, the adjusting temperature threshold can safety to maintain the overall system temperature. The proposed scheme prevents the system temperature from reaching a dangerous level by controlling $freq_{max}$ and maintaining the temperature within the steady state. This feature guarantees MPEG-4 from suffering any quality degradation

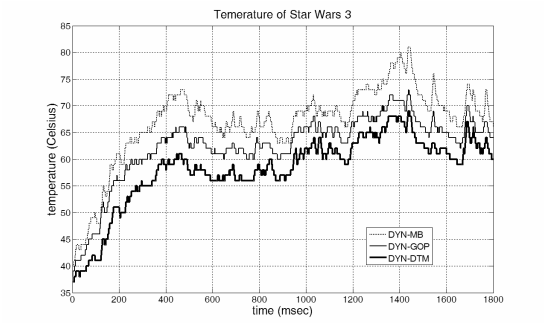
5 EXPERIMENTAL RESULTS

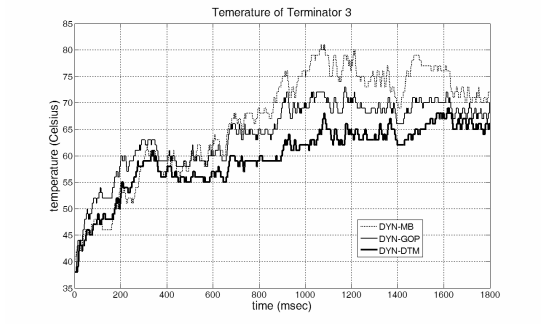
For our experiments, we modified the MPEG4IP source code and measured the temperature using ACPI on Linux. All experiments were performed without any cooling components, and the range of frequency had six levels from 600 Mhz to 1.6 Ghz. We experimented our scheme using twelve MPEG-4 movies. These movies were chosen as representatives of three types of complexity, which were with high-complexity, mid-complexity, and low-complexity. Among them, seven movies were classified as high action because they mostly had high-complexity scenes. The remaining three movies and two movies were classified as mid action and low action, respectively. These movies were presented for 30 minutes which had about $40,000$ frames. And each movies had different sizes and frame per second(fps). Also, all movies were encoded by MPEG-4 (ISO 14496) without any alteration of size and fps of the original data. To demonstrate the benefits of our control algorithm, we compare three schemes in terms of the number of missing frame, fps and the variance of temperature. `DYN-MB` scheme stands for the feedback controller with the display buffer, `DYN-GOP` scheme is the feedback controller with the display buffer and the information of GOP. Finally, `DYN-DTM` is the feedback controller based on the previous GOP information with DTM. Although `DYN-GOP` and `DYN-DTM` use the information of the previous GOP, only `DYN-DTM` supports the dynamic thermal management. TABLE 1 shows the number of missed frames of `DYN-DTM` and two other schemes for twelve selected movies. Since the number of missing frames directly affects the quality of MPEG-4, this can be considered as a performance metric to measure quality. The result shows that `DYN-DTM` is superior to other two schemes since there are no frames.

Table 1. The frame miss of movies

Movie Title	DYN-MB	DYN-GOP	DYN-DTM
Star Wars 3	151	83	0
Terminator 3	595	256	0
Ultra Violet	202	156	0
Any Given Sunday	0	30	0
24 (*season*1)	226	14	0
Under World 1	23	10	0
Under World 2	29	0	0
Blue Storm	0	0	0
Transporter 2	0	0	0
Eragon	166	11	0
Gilmore Girls	21	10	0
Just Friends	102	31	0

Figure 3 describes the variance of temperature in two movies, Star Wars 3 and Terminator 3, which have higher-complexity data than any other movies. It is observed that the `DYN-DTM` scheme controls the temperature more precisely than the other two schemes. This is because the ther-





(b) $Terminator\ 3$

Figure 3. Variance of Temperature of High-Complexity Movies

mal control in DYN-DTM uses the efficiency of frequency and temperature threshold. Another noticeable result is that DYN-DTM maintains the peak temperature to be at least 12% lower than other benchmark schemes. In Figure 3(a), there are high-complexity scenes in the first part and the last part of this movie, while the middle part has relatively lower complexity. Therefore, the DYN-DTM performs efficiently in the first and the last parts while maintaining the software temperature threshold at 70°C. Figure 3(b) also shows that DYN-DTM outperforms other two schemes even in multiple high-complexity scenes that are located at the middle of the movie. From these results, the proposed DYN-DTM scheme reduces the overall temperature up to 13% by using information from the previously decoded GOP and with dynamic thermal management. The most noticeable merit from this scheme is that it prevents all frames from exceeding the threshold temperature without dropping any frame at all.

6 conclusions and future work

In this paper, we proposed a method to find a proper frequency using an advanced feedback controller for the available decoding and display buffer based on the information of the previous GOP. Also, our scheme efficiently adjusts the frequency using a frequency efficient factor while keeping all frames from being dropped and maintaining thermal safety. We have implemented the proposed scheme on Linux and conducted benchmark testings. Experimental results prove that the proposed method does not drop any frames while the temperature is kept under the threshold. In other words, the proposed scheme suggests a solution for thermal constraints without any quality degradation for MPEG-4 decoding. For future work, we will extend our research to develop a system or a real-time application that can handle online input streams such as wired or wireless networks while managing low power and operating temperatures.

References

[1] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of ACM*, 54(1), 2007.

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

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *ISCA*, pages 83–94, 2000.

[4] K. Choi, K. Dantu, W.-C. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a mpeg decoder. In *ICCAD*, pages 732–737, 2002.

[5] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *ISLPED*, pages 217–222, 2003.

[6] J.E.Sergent and A.Krum. *Thermal Management Handbook*. McGraw-Hill, 1998.

[7] K.-J. Lee and K. Skadron. Using performance counters for runtime temperature sensing in high-performance processors. In *IPDPS*, 2005.

[8] Z. Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decode power using feedback control. In *ICCD*, pages 489–497, 2003.

[9] K. Skadron. Hybrid architectural dynamic thermal management. In *DATE*, 2004.

[10] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *HPCA*, 2002.

[11] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture, 2003.

[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. *Microelectronics Journal: Circuit and Systems*, May 2003.

[13] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. In *ECRTS*, pages 161–170, 2006.