Dynamic Thermal Management for MPEG-2 Decoding

Wonbok Lee
Department of Electrical Engineering
University of Southern California
Los Angeles CA 90089
(213) 821-4206
wonbokle@usc.edu

Kimish Patel
Department of Electrical Engineering
University of Southern California
Los Angeles CA 90089
(213) 821-4206
kimishpa@usc.edu

Massoud Pedram
Department of Electrical Engineering
University of Southern California
Los Angeles CA 90089
(213)-740-4458
pedram@usc.edu

ABSTRACT

In this paper, we propose an effective dynamic thermal management (DTM) scheme for MPEG-2 decoding by allowing some degree of spatiotemporal quality degradation. Given a target MPEG-2 decoding time, we dynamically select either an intra-frame spatial degradation or an inter-frame temporal degradation strategy in order to make sure that the microprocessor chip will continue to stay in a thermally safe state of operation, albeit with certain amount of image/video quality loss. For our experiments, we use the MPEG-2 decoder program of MediaBench and modify/combine Wattch and HotSpot for the power and thermal simulations and measurements, respectively. Our experimental results show that we achieve thermally safe state with spatial quality degradation of 0.12 Root Mean Square Error (RMSE) and with frame drop rate of 12.5% on average.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Design Aids

General Terms

Design, Reliability

Keywords

Thermal model, temperature-aware design, MPEG-2 decoding.

1. INTRODUCTION

Peak power dissipation and resulting temperature rise have become the dominant limiting factor to processor performance and 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 al-ready 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.

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 knowledge about the resource utilization and power dissipation demands of a software program running on a micro-processor 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.

In order to reduce chances of creating a major thermal problem at the architectural level, a number of DTM strategies have been proposed [1]-[9]. These techniques rely on two pre-defined levels of thermal limits: a trigger temperature and an emergency temperature. The trigger temperature is a thermal limit over which a dynamic predictive/reactive thermal management schemes will be initiated whereas the emergency temperature is a thermal limit over which the chip may be damaged and hence must be avoided at all cost. Those DTM schemes include architectural adaptations such as fetch-toggling [1] (instruction fetching is stalled for next N cycles), instruction cache throttling [2] (throttle the instruction forwarding from the instruction cache to the instruction buffer), activity migration [3] (dispatching computations to different locations on the die) and dynamic voltage and frequency scaling [4] (DVFS). Those DTM schemes are application-independent schemes. On the contrary, in this paper we propose an application-specific DTM technique, specifically designed for an MPEG-2 decoding program running on a general purpose microprocessor chip.

As computers become faster, absolute decoding time of a frame in MPEG-2 video stream becomes smaller. However, MPEG-2 standard prescribes a fixed frame rate of 29.97framess/sec (NTSC) and 25frames/sec (PAL) [10]. The frame rate is determined in consideration of slow trace/pursuit nature of human visual system (HVS). A frame rate higher than this 25-30 does not effectively improve the perceived quality of image/video streams to human eyes. Hence this, frame rate and processor speed dependent, available residual time from the given frame decoding time can be used to achieve thermally safe state of the processor.

This paper is organized as follows. In section 2, current state-of-the-art in DTM is reviewed. Section 3 introduces the motivational example of our DTM for MPEG-2 decoding while section 4 covers the theoretical parts of our DTM scheme. Our

ISLPED'06, October 4-6, 2006, Tegernsee, Germany.

simulation environment, benchmark programs and implementation details will be followed in section 5. In section 6, we show the experimental results of our work and section 7 is the conclusions.

2. PRIOR WORK

Many of the requisite performance features of microprocessor such as real-time processing and mean-time to failure are significantly affected by the power dissipation and resulting temperature. Hence, 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. To this end, there have been a number of studies that address this problem as explained in the following discussion.

Recently, a number of architectural adaptations based DTM solutions have been proposed. In [1]-[3], the authors propose fetch toggling, instruction cache throttling, and activity migration, respectively. In [4], the authors consider instruction window resizing and switching among active functional units as DTM techniques for multimedia applications. In [6]-[8], Skadron et al. introduce several DTM methods. In [6], they introduce temperature-tracking based frequency scaling, localized toggling and computation migration to spare hardware units. In [7], they propose a hybrid DTM technique that combines fetch gating and DVS. In [8], they propose a formal feedback control theory and use DTM as a test vehicle. DTM is invoked in response to the localized hotspot rather than chip-wide temperature.

In [5], Mircea et al. propose HotSpot which is an accurate yet fast thermal model based on thermal resistance and thermal capacitance at micro-architectural functional block level as well as two dimensional grid levels. Their model is implemented and widely used as a popular thermal simulator [11] . In [9], Lee et al. propose a software solution for temperature sensing that utilizes the built-in performance monitoring unit (PMU) to generate performance related information such as I/D cache access, number of instructions, number of stalls, etc. They derive temperature behavior by associating these performance figures with power.

Many power management (PM) schemes for MPEG have been proposed. In [12], Son et al. propose a dynamic voltage scaling (DVS) on MPEG decoding. Basically, they apply two dynamic voltage scaling schemes on MPEG decoding. One is based on delay and drop rate minimization algorithm and the other is based on predictive (per Group of Picture, GOP) decoding time algorithm. Their delay and drop rate minimization algorithm regulates the system voltage depending on the system clock speed and the current decoding status. The proposed algorithm assumes MPEG decoding in a low-performance machine in which the frame rate is less than 30frame/second.

In [13][14], Choi et al. propose an off-chip latency driven dynamic voltage and frequency scaling (DVFS) for MPEG decoding. In designing their DVFS strategy, the authors utilize the frame dependent versus frame independent parts within MPEG decoding process in [13] and the on-chip versus off-chip (CPU versus memory) dependent workloads within the frame decoding process in [14]. Their schemes are effective in slow machine which has a frame rate of 10 ~ 15.

All of the aforementioned schemes make use of the available, frame rate dependent, slack time to employ various low power strategies, but none of them make use of it for DTM. Since CPU speed and computational power are increasing rapidly, we have more slack time available during the decoding of an MPEG frame (with a fixed deadline of say 33ms). In this paper we propose to gather and distribute this available slack time to achieve thermally safe state of the microprocessor chip during MPEG-2 decoding. The safe thermal state comes at the expense of image/video quality.

Main 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 IntegerALU,1 IntegerMULT/DIV 2 FP ALU, 1 FP MULT/DIV
RUU/LSO size	64/32
Instruction Fetch Queue	8
In order Issue	False
Wrong Path Execution	True
Issue Width	6 instruction per cycles

Table 1. Baseline Configuration of Simulated Processor

Table 1 summarizes the architectural parameters that we use in our simulation.

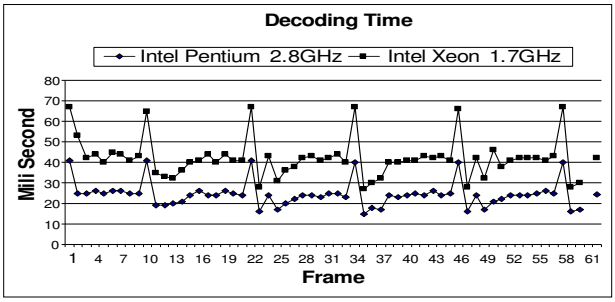

Figure 1 Actual MPEG-2 Decoding Time

3. MOTIVATIONAL OBSERVATION

Figure 1 reports per-frame decoding time variation of a MPEG-2 video stream decoded with MediaBench MPEG-2 decoder program [15]. The video stream has 60 frames and 704x480 resolutions. We run the decoder with two different machines: An Intel Xeon 1.7GHz and an Intel Pentium IV 2.8GHz. The OS is Linux-2.6.15. Note that the actual decoding time varies depending on the types of frame (I, P and B) Since I-frame is computation intensive compared to the other two frame types, its decoding time is longer than the others. In each machine, the average decoding times are 42.01msec and 24.01msec, respectively. Since the MPEG-2 standard specifies its frame rate as 29.97fps (which corresponds to approximately 33msec/frame), we clearly cannot finish the decoding of a frame with this frame rate in Intel Xeon chip. For this reason, MPEG standard has a frame discarding scheme [16] whereby it can drop B frame, P frame and I frame in stepwise manner, depending on the machine's clock speed.

In Pentium IV, on the contrary, the actual decoding time takes less than 33msec/frame. For such a case, MPEG standard has its frame rate control scheme that waits for some time to display the frame in a regular interval. For example, Berkeley MPEG-1 [17] uses 'select' function call to slow down displaying frames in a fixed rate of 29.97fps. Since the current state-of-the-art processors are much faster, it is expected that more residual time will be available within the allowed frame decoding time.

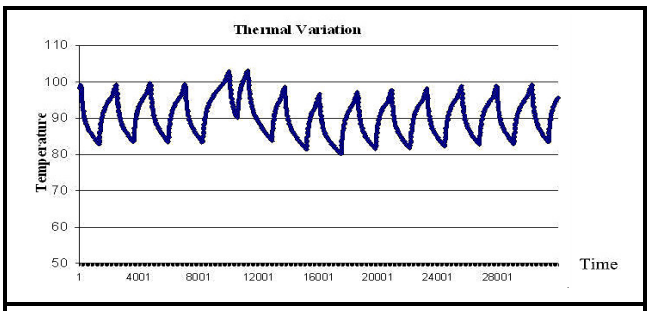

Figure 2 Thermal Variations & Violations in the Simulator

In Figure 2, we simulate this ever-decreasing actual decoding time in our simulator. (Section 5 will explain the simulator in detail) Simply speaking, we assume some fixed number of cycles that correspond to the given decoding time deadline (33msec). If the actual decoding finishes earlier than these many cycles (deadline), we stall the processor inside the simulator for the rest of the cycles until the deadline is reached and only then, we start the decoding of the next frame. The corresponding results are shown in Figure 2 where the X-axis plots the simulation cycles in 10K granularity and the Y-axis plots the temperature. As shown in the figure, the temperature starts to decrease when the actual decoding finishes any time earlier than the given frame decoding time. Note that the peak temperature goes up to 103□, which can invoke logical or timing error in the chip.

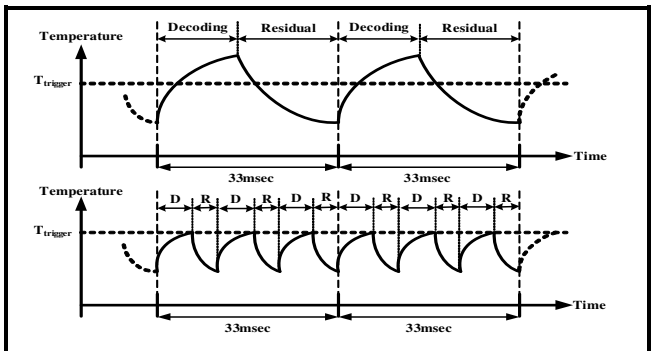

Figure 3 Overview of our DTM Scheme

In order to avoid thermal violation (in Figure 2), we propose a new DTM scheme, especially for the MPEG-2 decoding. Figure 3 shows the basic idea behind: Given a deadline for frame decoding, the conventional MPEG-2 decoder uses the first part of the decoding time to finish the decoding task while it rests in the second part. Unfortunately, the trigger temperature/emergency temperature of the chip may be exceeded in the first part. In our strategy, short periods of decoding are interleaved with short periods of processor stalls so that the chip temperature never exceeds the trigger temperature, yet the decoding task is completed before the deadline.

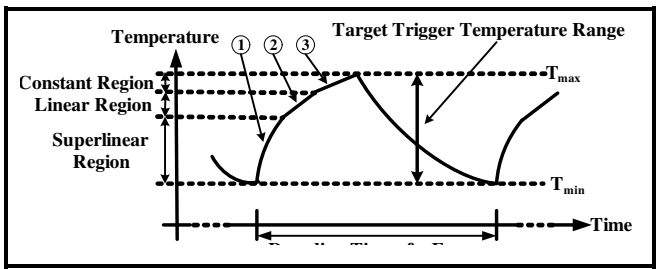

Figure 4 Detailed Analysis of Temperature Variation

4. DTM METHODOLOGIES

4.1. Thermal Models

We use a thermal model developed by Skadron et al. in [8]. In this model, the temperature increase in the chip is represented by:

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

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

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

Let $t_{initial}$ and t_{final} denote two instances of time (and their difference denoted by Δt), respectively. Moreover, assume that the power dissipation is non-zero when the processor is running. Now, the thermal rising gradient with respect to time is calculated as:

$$\text{Rising: } \frac{\Delta T_r}{\Delta t} = \left(\frac{P}{C_{th}} - \frac{T_{old}}{R_{th}C_{th}}\right) \quad (3)$$

In contrast, when the processor is stalled (it is put in the standby mode) during the residual time, assume that the chip power dissipation is negligible compared to active power i.e., P=0. Then, the thermal falling gradient with respect to time is calculated as:

$$\text{Falling: } \frac{\Delta T_f}{\Delta t} = \left(-\frac{T_{old}}{R_{th}C_{th}}\right) \quad (4)$$

In Figure 4, we model this thermal gradient over time as three piecewise linear functions. Since the amount of decoding workloads/steps within a MPEG-2 frame decoding is frame dependent yet more or less the same, we specify the temperature variation during MPEG-2 frame decoding on a DTM-ignorant machine as T_{max} and T_{min} peaks and valleys, respectively. Notice that T_{max} and T_{min} are mostly invariant when a program is in a steady-state but may slightly vary when a program behavior changes. Moreover, obtaining T_{max} and T_{min} is not always possible in the actual system but is always feasible in the thermal simulator, which is aimed at anticipating application's real on-chip thermal behavior. Then, the thermal behaviors of MPEG-2 decoding program are divided into three regions:

1. **Super-linear Region**: $\frac{P}{C_{th}} >> \frac{T_{old}}{R_{th}C_{th}}$ In this region, $\frac{\Delta T_r}{\Delta t}$ changes dramatically and the power term dominates the other temperature term in equation 1. Since the thermal gradient during the rising of the curve is higher than falling counterpart, a longer processor stall time is needed compared to the time it took for the temperature

to rise to the same level.

2. **Linear Region**: $\frac{P}{C_{th}} > \frac{T_{old}}{R_{th}C_{th}}$ In this region, $\frac{\Delta T_r}{\Delta t}$ changes almost linearly and the power term is relatively larger than the temperature term in equation 1. The thermal gradients during the rising and the falling of the curve are comparable and both take almost same amount of time.
3. **Constant Region**: $\frac{P}{C_{th}} \approx \frac{T_{old}}{R_{th}C_{th}}$ In this region, $\frac{\Delta T_r}{\Delta t}$ becomes almost zero and the power term is comparable to the other temperature term in equation 1. Since the thermal gradient during the falling of the curve is higher than the rising counterpart, a shorter processor stall time is needed compared to the time it took for the temperature to rise to the same level.

Unfortunately, those three regions are not sharply bounded in the thermal gradient curve. Moreover, the trigger temperature is a material/architectural parameters dependent value whereas T_{max}/T_{min} is MPEG-2 input file dependent one, the trigger temperature can be located at any level within the thermal gradient curve. Hence, we carry out the following steps in order to build our DTM framework for MPEG-2 decoding.

1) Run a MPEG-2 video stream in MPEG-2 decoder program and get both T_{max} and T_{min} on the machine without any DTM policy applied.

2) Check $T_{trigger}$ of the processor. If $T_{trigger} > T_{max}$, machine is thermally safe and no DTM policy is needed to be applied.

3) If $T_{trigger} < T_{min}$, this means decoding workload is large and DTM policy must do significant quality degradation to achieve thermally safe state.

4) If $T_{min} < T_{trigger} < T_{max}$, which we show as a target trigger temperature range in Figure 4, then, if $T_{trigger}$ lies in the constant region, thermally safe state can be achieved with little or no quality degradation, whereas if it lies in the linear region some quality degradation must be accepted to achieve thermally safe state. Finally, if $T_{trigger}$ lies in the super-linear region, which is the worst case, then thermally safe state can be achieved only at the cost of significant image/video quality degradation.

4.2. DTM Policy

During the residual time, as shown in Figure 3 (top figure), the thermal gradient is high during its initial phase and slowly decreases afterwards. Even though significant amount of time is spent in 'stalling the processor', i.e. doing nothing but waiting for the arrival of frame display time, the drop in the temperature is relatively small. This leads us to the idea depicted in Figure 3 (bottom figure): Stall the processor only for the duration of time till the thermal gradient remains steep.

From the analysis presented in the previous section, high thermal gradient occurs only when the chip operates in either the linear or the constant regions. Our experiments with a set of input files with a trigger temperature of 81.8□ show that the trigger temperature is positioned in the 'linear region' where thermal rising/falling gradients are comparable. Based on this simulation results and extensive experiments, we choose an empirical value of 1 million cycles to stall the processor every time we reach the triggering temperature. This gives us comparable rising and thermal falling gradients

Note we may end up missing deadlines for frames unless we do have enough residual time. In order to reduce these deadline misses, we collect this slack time (actual decoding time subtracted from given decoding time) for the future use. This slack time saving is accomplished by having a buffer in the main memory which has the size of 3 frames. Every time we finish actual decoding before the given decoding time and the buffer has space for frames, we write the decoded frame to the buffer and claim the remaining slack time for the future use. If the buffer does not have space, we wait for the buffer to have a space. If we either miss or predict to miss the deadline for the frame being decoded, we resort to either spatial or temporal quality degradations to meet the deadlines. This deadline satisfaction and slack time collection continue over the whole execution time.

Spatial quality degradation: After enough amount of frame decoding time, e.g. around 100msec, thermal behavior of MPEG-2 decoding program becomes monotonous. Hence in decoding of all subsequent frames, we can predict how many times the thermal curve will reach to the trigger temperature for the following frame decoding. Since we allow stalling 1 million cycles every time we reach the trigger temperature, the total number of stall cycles can be easily predicted. If the predicted stall cycles are larger than the available slack time that we collected, deadline miss is expected hence activation of spatial quality degradation (as explained latter) is triggered.

Temporal quality degradation: Note that spatial quality degradation does not guarantee the deadline satisfaction, i.e., we may run out of slack times to meet the deadline. If deadline miss occurs, we will drop the next frame. The rationale is that this frame has already missed its deadline and it is going to be displayed at the time when the next frame is supposed to be displayed. In other words, instead of delaying the display of the whole sequence of frames, we decide to drop the next frame so that the second to the next frame can be decoded before its deadline.

4.3. Spatiotemporal Quality Degradation

In our definition, spatial quality degradation is the ratio of how the modified frame differs from the original frame. We use root mean square error (RMSE) as a measurement metric of the spatial image quality degradation. The temporal quality degradation, in our definition, is the ratio of how the modified video stream differs from the original video stream. We use the number of skipped/dropped frames as a measurement metric of the temporal video quality degradation. Clearly, the spatial quality degradation is an intra-frame level image distortion whereas the temporal quality degradation is an inter-frame level video distortion. Note that the time saved due to spatial quality degradation is smaller than the temporal quality degradation.

4.3.1. Spatial Quality Degradation

In order to find the best decoding steps to minimally distort the frame image quality, we analyze the typical MPEG-2 decoding sequence shown in Figure 5. Frame decoding in MPEG-2 has several major steps: Variable Length Decoding (VLD), Inverse Quantization (IQ), motion compensation, Inverse Discrete Cosine Transformation (IDCT), dither, display, etc. Among those steps, we observe the SNR scalability and the saturation control, since they are employed to enhance the image quality. SNR scalability

provides the enhancement of video quality by means of enhancement layer. Basically, it has two levels of layers: a base layer and an enhancement layer. Base layer includes the coarse level of DCT coefficients and the enhancement layer includes the finer level of DCT coefficients. On the contrary, the saturation control is clipping the results of IQ. Those two fine granularity scalability (FGS) techniques in MPEG-2 are initially introduced to cope with the time-varying bandwidth for the smooth image quality degradation. Our experiments show that these two steps consume approximately 10% of the total frame decoding time. Since they are spatial quality related metrics and relatively easy to be divided from MPEG-2 decoding steps, we choose them to do the spatial quality degradation in our DTM framework.

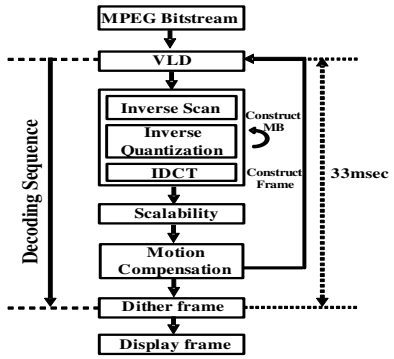

Figure 5 Typical MPEG-2 Decoding Steps

4.3.2. Temporal Quality Degradation

As mentioned earlier, temporal quality degradation is done by dropping the frames when deadline miss occurs. Note that not all the frames can be dropped arbitrarily. If a P frame is dropped, then all the subsequent P frames must be dropped till the next I frame. Whereas a B frame can be dropped arbitrarily since the next B frame does not depend on the B frame currently being dropped. In this paper hence we decide to drop only B frames so if the next frame to be dropped is not B then we keep decoding other I and P frames till we get the B frame.

Input files	No. of frame	Frame resolution	I: P: B frame distribution
gitape.m2v	14	720 x 480	1· 4· 9
mei60f.m2v	50	704 x 480	5· 13· 32
hhilong.m2v	45	720 x 576	3· 8· 34
time_015.m2v	50	704 x 480	5· 12· 33
soccer_015.m2v	51	640 x 480	4· 14· 33
tens_015.m2v	47	352 x 192	5· 12· 30
cact_015.m2v	50	352 x 192	5· 12· 33

Table 2 MPEG-2 Input Files Used in the Experiments

5. SIMULATION ENVIRONMENT

For our experiments, we modify and combine Simplescalar [18] Wattch [19] and HotSpot [11]. The simulated micro-processor model is based on ALPHA 21364, which is has the feature size of 0.18μ, V_{dd} of 1.6V and a clock speed of 1GHz. The power model used in the simulation does not model leakage power. In order to avoid modifying the default floor-plan in HotSpot, we use the same feature size and linearly scale both V_{dd} to 1.8V and a clock speed to 1.2GHz. The trigger and emergency temperature are set to 81.8 and 85.0□, respectively [6], whereas the ambient and initial temperatures are set to 40.0 and 60.0□ respectively. Our combined simulator generates thermal results of each functional unit every 10K cycles.

For the application programs, we use MPEG-2 decoder program of MediaBench benchmark suite [15]. Table 2 summarizes the MPEG-2 input files used in our experiments, which we mostly obtained from [20]. We add a few custom made files for the better comparison and limit the total number of frame to 51 in all experiments. Since 0.1□ rise/fall of temperature may take 100K cycles [6], our profile of thermal behavior of a program has long enough time. Our DTM policy is implemented in the MPEG-2 decoder program such that it interacts with our combined simulator.

Input files	real decoding time (msec)	Max/Min temperature (□)	
		DTM-ignorant	DTM
gitape	21.5	101.5 / 85.5	81.8 / 80.5
mei60f	19.6	99.6 / 83.8	81.8 / 80.5
hhilong	17.2	97.2 / 81.9	81.8 / 80.5
time_015	11.8	91.5 / 76.2	81.8 / 80.5
soccer_015	8.5	82.5 / 70.5	81.8 / 72.4
tens_015	4.0	73.4 / 63.2	N/A
cact_015	4.0	73.4 / 64.1	N/A

Table 3 Thermal Behaviors in the Hottest Functional Unit

6. EXPERIMENTAL RESULTS

In Table 3, we summarize the experimental results. The left column shows the actually measured decoding time and right two columns compare the maximum/minimum temperatures before and after DTM scheme. As you see, maximum temperature in DTM-ignorant system shows that thermal crisis can occur in some cases. Note that the maximum-minimum temperatures for the input files with similar resolution are the same since they have approximately the same decoding workload. When the resolution becomes smaller, the maximum-minimum temperatures are both decreased. The N/A parts mean that no DTM scheme is necessary for those input files.

Input files	Image/Video Quality Degradation			
	Spatial		Temporal	
	Scaled frames	RMSE	Dropped frames	Drop ratio (%)
gitape	5	0.119	5	35.7
mei60f	8	0.125	15	30.0
hhilong	0	N/A	8	8.8
time_015	0	N/A	0	0
soccer_015	0	N/A	0	0
tens_015	0	N/A	0	0
cact_015	0	N/A	0	0

Table 4 Spatial/Temporal Quality Degradation

In Table 4, we summarize the experimental results for the image quality degradation. For the measurement of spatial quality degradation, we use root mean square error (RMSE) of the luminance (Y) values of frames. Note that this RMSE values are

not calculated among all frames but for the spatially scaled frames only. For the temporal quality degradation, we show the number of frames dropped. As shown, when the resolution of a frame becomes smaller, number of dropped frames reduces since we have enough amount of residual time and become non-aggressive, i.e., scale frame frequently instead. Clearly, frames with large resolution will have more number of dropped frames.

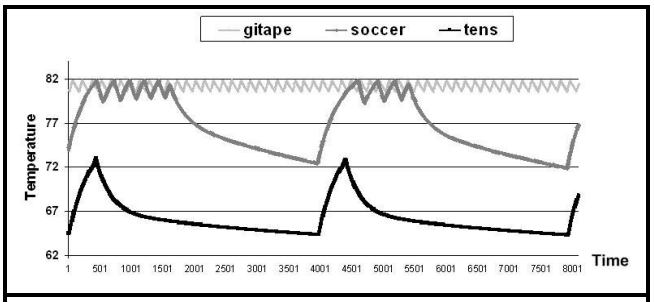

Figure 6 Comparison of the Thermal Variations

In Figure 6, we show the run time thermal behavior during the simulation. For simplicity, we use three input files which have different resolutions: gitape, soccer_015 and cact_015. Each point in the X-axis is the measurement step in 10K cycles and Y-axis is the temperature and all measurements are made when a program reaches to the thermally steady state. We categorize files into three groups based on their workload, i.e. actual decoding time: Files with heavy workload execute DTM aggressively (top graph), files with medium workload execute DTM non-aggressively (middle graph), and files with light workload never execute DTM (bottom graph). Clearly, actual decoding time (in turn, the residual time) has strong relationship with the necessity of DTM scheme and our scheme shows that we achieve thermally safe state with different image/video quality degradation, i.e., number of dropped frame and the number of scaled frames are smaller in soccer_015 than frame dropped/scaled in gitape.

7. CONCLUSIONS

In this paper, we propose an effective DTM scheme for MPEG-2 decoding with the spatiotemporal quality degradation. Our DTM algorithm makes use of the ever-decreasing actual frame decoding time and utilizes the residual time within a frame decoding by distributing it to achieve thermally safe state. As a consequence, quality degradation is observed. Our experimental results show that we achieve thermally safe state with spatial quality degradation of 0.12 in terms of RMSE value and with a frame drop rate of 12% on average. Our future research will carry out the analysis of the other steps of MPEG decoding in order to do more stepwise quality degradation with respect to decoding time.

8. REFERENCE

[1] D. Brooks, M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," *Proceedings of International Symposium on High Performance Computer Architecture (HPCA*), January, 2001.

[2] H. Sanchez, B. Kuttanna, T. Olson, M. Alexander, G. Gerosa, R. Philip, J. Alvarez, "Thermal Management System for High Performance PowerPC Microprocessors," *Proceedings of IEEE Computer Society International Conference (COMPCON),* 1997.

[3] S. Heo, K. Barr, K. Asonovic, "Reducing Power Density through Activity Migration," *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED),* August, 2003.

[4] Jay Srinivasan, Sarita V. Adve, "Predictive Dynamic Thermal Management for Multimedia Applications," *Proceedings of International Conference on Supercomputing (ICS),* June, 2003.

[5] Mircea R. Stan, K. Skadron, M. Barcella, W. Huang, K. Sankaranarayanan, S. Velusamy, "HotSpot: a Dynamic Compact Thermal Model at the Processor-Architecture Level," *Microelectronics Journal: Circuit and Systems,* 2003.

[6] K. Skadron, Mircea R. Stan, W. Huang, S. Velusamy, Kathik Sankaranarayanan, David Tarjan, "Temperature-Aware Microarchitecture," *Proceedings of International Symposium on Computer Architecture (ISCA),* June, 2003.

[7] K. Skadron, "Hybrid Architectural Dynamic Thermal Management," *Proceedings of the Design Automation and Test in Europe (DATE),* 2004.

[8] K. Skadron, T. Abdelzaher, Mircea R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA)*, 2002.

[9] K. J. Lee, K. Skadron, "Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors," *Proceedings of High Performance, Power-Aware Computing (HP-PAC),* April, 2005.

[10] MPEG-2 Standard: International Organization for Standardization/International Electro-technical Commission (ISO/IEC) 13818-2.

[11] HotSpot at http://lava.cs.virginia.edu/HotSpot

[12] D. Son, C. Yu, H. Kim, "Dynamic Voltage Scaling on MPEG Decoding," *Proceedings of International Conference on Parallel and Distributed Systems (ICPADS),* 2001.

[13] K. Choi, R. Soma, Massoud Pedram, "Off-chip Latency Driven Dynamic Voltage and Frequency Scaling for an MPEG Decoding," *Proceedings of Design Automation Conference (DAC),* June, 2004.

[14] K. Choi, K. Dantu, W. C. Cheng, Massoud Pedram, "Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder," *Proceedings of International Conference on Computer Aided Design (ICCAD),* 2002.

[15] MediaBench at: http://euler.slu.edu/~fritts/mediabench

[16] M. Verderber, A. Zemva, A. Trost, "HW/SW Codesign of the MPEG-2 Video Decoder," *Proceeding of International Parallel and Distributed Processing Symposium,* 2003.

[17] Berkeleympeg http://bmrc.berkeley.edu/frame/research/mpeg

[18] Simplescalar turorial at http://www.simplescalar.com

[19] D. Brooks, V. Tiwari, M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proceedings of International Symposium on Computer Architecture (ISCA),* 2000.

[20] MPEG2 streams at http://www.mpeg2.de/video/streams