ARCHITECTURAL ADAPTATION FOR THERMAL CONTROL

BY

JAYANTH SRINIVASAN

B.Tech., Indian Institute of Technology, Madras, 2000

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2002

Urbana, Illinois

Abstract

Thermal power management is an important area of reasearch for general-purpose processors. In order to save on chip packaging costs, Dynamic Thermal Management (DTM) techniques have been proposed. However, these techniques result in a degradation in processor performance when invoked. This paper concerns performance effective DTM for general-purpose processors running multimedia applications.

We make two contributions: (1) Current DTM algorithms are **reactive** in nature. We propose a **predictive** DTM algorithm targeted at multimedia applications. We find that for our applications, our predictive algorithm performs significantly better than existing reactive DTM algorithms, (2)We evaluate DTM response mechanisms using previously proposed accurate thermal models for multimedia applications.

We find that response methods targeted to specific ”hot-spots” on chip are much more effective than previously proposed mechanisms that affect the entire processor.

To Appa and Amma

Acknowledgements

I am grateful to my adviser, Sarita Adve, for her support, guidance, and encouragement. The flexibility she gives her students and her immense patience make her a pleasure to work with.

Thanks to Chris Hughes for always having answers to my endless stream of trivial and not-so-trivial questions - I'm yet to find something he can't help me with. Thanks to Rohit for being a great officemate inside the department and an even better friend outside.

I've had an amazing time so far in Illinois, due in no small part to the great companionship I've found at 503, 1010, and Cosmo. Special thanks to Arun, Sachin and Devatha for always being there for me.

Words can't express my gratitude to my parents and my sister for their constant love and support. Appa and Amma, I owe everything to you.

Table of Contents

1 Introduction

Power dissipation is a significant issue in modern microprocessor design. Although the majority of current research is targeted at reducing energy consumption, thermal power is also emerging as an important design parameter. It is anticipated that in the future, peak power dissipation and consequent thermal considerations will often be the dominant limit to processor performance and a significant component of cost. Borkar [2] estimates that above 35-40W, thermal packaging increases the total cost per chip by $1/W. It is estimated that future processors will have a peak power consumption of nearly 150W [1], neccessitating prohibitively expensive thermal packages. Current thermal packages are designed for peak processor power in order to ensure safe operation at all times. However, the peak processor power is rarely observed, and tends to be much higher than the average processor power. This disparity between peak processor power and average processor power will continue to increase as further techniques are applied to extract ILP from chips. One proposed solution to the packaging problem is to design the thermal packaging for a temperature less than the peak and to use dynamic thermal management (DTM) techniques in the rare cases when the packaging temperature limit is approached [3, 16, 8]. However, when the DTM technique is invoked, there is typically a degradation in performance.

Another trend of interest is that multimedia applications are expected to form a large part of the workload on a growing number of systems, including future handheld computers, wireless telephones, laptop computers, and desktop systems [6, 12]. General-purpose processors (as opposed to DSPs and ASICs) are expected to be increasingly employed for such workloads [6]. Many popular multimedia applications are computation intensive and will result in significant thermal power consumption. Additionally, extensive research has been performed on architectural adaptation and dynamic voltage scaling(DVS) for such systems.

Based on the above, this work concerns DTM techniques employing architectural adaptation and DVS for general-purpose processors running multimedia applications.

Substantial work has been done in the research community on adaptive architectures and DVS for energy reduction (see references in [14]). Although energy and thermal constraints are both related to power dissipation, there are distinct differences between the two, and energy reduction algorithms cannot directly be applied to the thermal problem. Instead, algorithms specific to thermal power must be developed.

All of the previous work in dynamic thermal management for processors is based on *reactive* DTM techniques. While the system is running, if the temperature on chip gets too close to the packaging limit, a response that curtails the performance of the system is initiated (e.g., fetch-toggling), resulting in a temperature reduction. Such schemes suffer from at least two problems. First, reactive schemes have limited time to respond to a thermal emergency. They must use mechanisms which have a low time overhead so that they can be quickly invoked. As a result, mechanisms like dynamic voltage scaling (DVS) and register file resizing which have large thermal benefits but high invocation time overheads cannot be used. Second, engaging the appropriate reactive response at the appropriate time requires significant prior tuning of the system to determine the appropriate temperature trigger for an appropriate response. To overcome this problem, Skadron et al. [16] proposed the use of control theoretic techniques. This approach appears promising; however, it is as yet unclear how to tune controllers for use with the complex, potentially multiple, microarchitectural responses that are possible in the systems we consider (see Section 4). Further, control theoretic schemes also suffer from the first drawback described above (i.e., the inability to use high overhead response mechanisms).

In this paper, we make two contributions. First, we propose a DTM algorithm that takes a *predictive* approach and uses architectural adaptation and DVS to run at thermally safe but performance effective levels for multimedia applications. Our algorithm overcomes the above mentioned problems with reactive approaches. (The algorithm is inspired by, but is different from, a recent energy saving algorithm for multimedia applications proposed by Hughes et al. [10]). Multimedia applications process discrete units of data called frames. Previous work has shown that frames of the same type involve more or less the same type but not the same amount of work [9]. We extrapolate this result to observe that each frame (or a predetermined set of consecutive frames) will reach the same peak temperature, which can be determined by profiling one frame (at a lower voltage and frequency to avoid chip burn). This profile information can be used to predict an efficient and thermally safe architectural configuration for the rest of the frames of that type. Since adaptation is invoked at most once per frame, high overhead adaptation mechanisms are possible. Additionally, our predictive algorithm is simpler in that it does not require any application-specific and response specific tuning of reactive controllers.

Our second contribution is in the evaluation of several DTM response mechanisms for the processor core using accurate thermal models and local ”hot-spots” (as proposed by Skadron et al. [16]) for multimedia applications. Previous work has either used power as a proxy for temperature [8, 3] or has considered only one processor response mechanism (fetch-toggling) [16]. Further, none of the previous work has focused on multimedia applications. Existing work on DTM focuses on mechanisms that curtail the performance of the entire processor, despite the local nature of thermal hot spots. This could lead to a larger degradation in performance than is necessary. Instead, we examine the benefit of using DTM mechanisms like instruction window resizing and switching off active functional units which are local to thermal hot spots on chip.

Our evaluation considers nine multimedia benchmarks, and studies the response mechanisms of instruction window resizing, switching off active functional units, fetch-toggling at various levels, and dynamic voltage scaling. Our findings are as follows:

1. The local reactive DTM mechanisms we examined perform considerably better than the global reactive mechanisms like fetch-toggling for the applications examined. In our simulations, the register file was always the hottest structure on chip and we found that the local adaptations of switching off active functional units and instruction window resizing were very effective in controlling the register file temperature.

2. For the applications examined, our predictive DTM algorithm performs significantly better than the reactive algorithms. This is mainly due to two important observations:

    - For most of our applications, we find that the hottest structures for our system have a roughly constant temperature throughout the run. Although there is considerable variation in the instructions per cycle (IPC) of the application during the run, due to the large thermal time constants of the structures on chip, the corresponding temperature changes are much smaller. Due to this lack of variation in temperature, we can expect our predictive algorithm to perform at least as well as the reactive algorithms, and in some cases better.
    - Our predictive algorithm can use adaptations with high time overhead like DVS and register file resizing. These adapatations result in significant thermal benefit with relatively lower performance degradatation. On the other hand, as mentioned previously, because of the time overhead involved, these adaptations are not available to reactive algorithms.

The rest of the paper is organized as follows - In Section 2, we describe the thermal model used in our simulations. In Section 3, we discuss existing reactive DTM algorithms, and in Section 4, we describe our proposed predictive DTM algorithm. Section 5 describes the experimental methodology used in this paper, Section 6 describes the results obtained, Section 7 has a discussion of some salient points from the results and avenues for future work, and Section 8 has our conclusions.

2 Temperature Model

Various thermal models have been used in previous architectural studies. Brooks et al. [3] use the moving average of thepower over the last 10,000 cycles as a proxy for chip temperature. Although the temperature on chip is a function of the processor power dissipation, using power as a proxy for temperature is unsatisfactory because it does not capture the exponential behavior of temperature with time, caused by on-chip thermal resistances and thermal capacitances. Also, individual structures on chip tend to heat up much faster than the chip temperature as a whole. Monitoring the entire chip's temperature can cause localized thermal "hot-spots" to be overlooked.

Huang et al. [8] use a very similar power model. They use a recursive definition which adds 0.75 times the power consumed in the last millisecond and 0.25 times the power a millisecond ago as a proxy for temperature. Although this attempts to model the exponential behavior of temperature, it is still inferior to an accurate thermal model. This model also misses local thermal hot-spots.

Dhodapkar et al. [5] use a better thermal model in their TEM^2P^2EST power simulator. Rather than using power as indicative of temperature, they use a thermal RC equivalent circuit to model temperature. However, they also ignore local hot-spots by looking at chip wide power.

Our thermal model is based on the one used by Skadron et al. [16]. We track the temperature of individual structures on chip using exponential rate equations, based on each structure's thermal resistance and capacitance. As a result, we can accurately model the local temperature of different parts of the chip.

As in [16], we model temperature at the granularity of a functional block, treating each block a uniform heat source. Our thermal model ignores tangential resistances between different functional blocks, assuming that it would be much larger than each block's individual normal thermal resistance [16] - in other words, in our model, the temperature of a block will not be directly influenced by any other block. Recently, the authors of [16] have quantified the impact of this approximation [17]. Those results show a less than 5% impact on temperature due to the approximation.

Each block functions like a space heater and the power dissipated translates into a temperature increase with a certain time lag. This is equivalent to a conventional RC electric circuit, where the functional block will have a certain normal thermal resistance R_{th} (Kelvin/Watt), and a certain thermal capacitance C_{th}(Joule/Kelvin).

Based on this, the steady state temperature, T_{ss}, of the functional block is given by

$$T_{ss} = T_{ambient} + T_{heat-sink} + R_{th} * P$$

where $T_{ambient}$ is the ambient room temperature, $T_{heat-sink}$ is the *difference* in temperature between the surface of the heat sink and the ambient, and P is the instantaneous power being dissipated by the block.

The steady state temperature of the functional block (T_{ss}) can be used to get an idea of the maximum possible range of temperatures being operated in, and the thermal RC time constant ($R_{th} * C_{th}$) can be used to get an idea of the time taken for the temperature of the functional block to change.

In this paper, we are interested in dynamic variations in temperature. As a result, we ignore variations in the ambient temperature ($T_{ambient}$) and the heat sink temperature ($T_{heat-sink}$), because of their slow rate of change. In the following equations, we assume a constant heat sink and ambient temperature and derive all relations relative to a constant base temperature of $T_{ambient}$+$T_{heat-sink}$.

To obtain an expression for the variation in temperature, we consider a block changing in temperature from T_{old} to T_{new} over a time interval Δt, while dissipating an average power P in the interval. Then, based on the exponential rise equation,

$$T_{new} = PR_{th} - (PR_{th} - T_{old})e^{\frac{-\Delta t}{R_{th}C_{th}}}$$

For $\Delta << R_{th}C_{th}$, we can expand the exponential and ignore all high order terms to give

$$T_{new} = PR_{th} - (PR_{th} - T_{old})(1 - \frac{\Delta t}{R_{th}C_{th}})$$
$$= T_{old} + PR_{th}\frac{\Delta t}{R_{th}C_{th}} - T_{old}\frac{\Delta t}{R_{th}C_{th}}$$

The change in temperature $\Delta T = T_{new} - T_{old}$, is given by

$$\Delta T = \frac{P\Delta t}{C_{th}} - \frac{T_{old}\Delta t}{R_{th}C_{th}}$$

Hence, for any functional block, based on its power consumption, the above equation can be used to calculate the new temperature after any convenient time interval $\Delta t << R_{th}C_{th}$.

In this work, we also analyze the effect of voltage and frequency scaling on temperature. Using the well known proportionality relating power, voltage and frequency, $P \alpha V^2 f$, and substituting for P in the above thermal equation, we can calculate the block temperature at any voltage and frequency. More specifically, the relationship between the temperatures, T_{f_1} and T_{f_2}, at two different frequencies, f_1 and f_2, is given by $\frac{T_{f_1}}{T_{f_2}} = \frac{V_1^2 f_1}{V_2^2 f_2}$, where V_1 is the system voltage at f_1 and V_2 is the system at f_2.

Two features of the thermal model are of specific interest. First, it is evident that temperature is closely related to power consumption. Hence, in order to reduce the temperature of a functional block, we should reduce the power dissipated by the block. Second, the exponent , $\frac{\Delta t}{R_{th}C_{th}}$, implies that unless there is a significant change in power for a period comparable to the thermal RC time constant, there will not be a noticeable change in temperature.

3 Reactive DTM algorithms

As mentioned previously, all of the previous work in dynamic thermal management for processors is based on *reactive* DTM techniques. While the system is running, if the temperature on chip gets too close to the packaging limit, a response mechanism is initiated. This response leads to a reduction in temperature, typically accompanied by a degradation in the performance of the processor. Once the processor resumes safe thermal operation, the reaction mechanisms are shut off. There are two key features in the design of reactive DTM algorithms: When to invoke reactive response mechanisms and what response mechanism to use.

Huang et al. [8] proposed DEETM, a framework capable of dynamically choosing multiple energy and temperature management techniques. Every few milliseconds, thermal conditions are checked and if emergency levels are approached, hardware to affect adaptation is invoked. The reactive mechanisms examined in that work are DVS, entering light sleep mode, and some mechanisms targeted at the memory heirarchy.

Brooks et al. [3] proposed five reactive mechanisms - clock frequency scaling, voltage and frequency scaling (DVS), fetch-toggling (every N cycles, instruction fetch is disabled), throttling (the number of instructions fetched every cycle is reduced from the normal instruction fetch bandwidth), and speculation control (whenever the number of unresolved branches crosses a certain number, instruction fetch is disabled). These mechanisms were invoked when the processor power consumption crossed a predetermined threshold. In their simulations, they set this threshold at a constant of 24W for all their applications. In their analysis, they found clock frequency scaling and DVS to be inefficient because of the time overhead involved in their invocation. Among the other schemes, fetch-toggling was found to perform the best for most benchmarks and speculation control for others.

Skadron et al. [16] proposed the use of control theory algorithms for DTM, using fetch-toggling as their reaction mechanism.

The DTM algorithms proposed by Skadron et al. improve on previous DTM algorithms in two significant ways. One, both the previous algorithms have fixed trigger temperatures for all applications and situations. Consequently, in order to ensure safe operation, the reaction trigger temperature would have to be set at a conservative value, potentially leading to poor performance for some applications. Skadron et al. use control theory to vary their trigger levels based on the application. Second, both the previous DTM algorithms have fixed strength responses to thermal emergencies. Again, using control theory, Skadron et al. vary the fetch-toggling rate based on the thermal stress level.

All these reactive DTM algorithms suffer from certain problems:

- The reaction mechanisms proposed for the processor core restrict the performance of the entire processor even though localized heating occurs at a faster rate than chip-wide heating. Invoking adaptations local to the thermal hot-spot can potentially provide the same thermal benefit with a less severe performance penalty. Huang et al. [8] have proposed techniques local to the memory heirarchy. However, in this paper, we concentrate on processor temperature control.

- Reactive DTM algorithms have a limited time to respond to a thermal emergency. They must use mechanisms which have a low time overhead so that they can be quickly invoked. As a result, mechanisms like dynamic voltage scaling (DVS) and register file resizing which have the potential for significant thermal benefit cannot be used because of their large time overheads.

- Engaging the appropriate reactive response at the appropriate time requires significant response specific and application specific tuning of the DTM algorithm. Although Skadron et al. [16] use control theoretic techniques to overcome this problem, it is as yet unclear how systems with complex, potentially multiple reaction mechanisms can be efficiently tuned.

In order to compare our proposed predictive DTM algorithm with reactive algorithms, we have implemented two reactive algorithms, R-Toggle, and R-IwFu to represent the state of the art.

- **R-Toggle**

  R-Toggle is the best manually tuned(non-control theoretic) reactive algorithm studied by Skadron et al. in [16]. R-Toggle uses fetch-toggling as its reactive mechanism. It proportionately varies the toggling rate from no toggle (normal instruction fetch) to full toggle (no instruction fetch) depending on the proximity to the emergency temperature. Less toggling is used when the thermal stress is low and severe toggling is used when the thermal stress is high. The tuning mechanism we used for R-toggle is described in Section 3.1. We do not model the control-theoretic algorithms studied by Skadron et. al because,(1) their performance impact was very small in relation to R-Toggle (on average, R-Toggle slowed their system by 0.04% while their best control theoretic scheme slowed their system by 0.025% [16]),(2) we do not yet understand how to develop control theoretic algorithms tuned for multiple responses for multiple applications as in the R-IwFu scheme described below, and wanted to use similar tuning techniques for the two algorithms to ensure a fair comparison, and (3) for most of the applications, we see that temperature stays fairly constant; therefore, manually tuned algorithms are very likely to perform similar to control theoretic algorithms.

- **R-IwFu**

  As mentioned earlier, in our simulations, we found that the register file was the hottest structure on chip for all our applications. Although toggling reduces the number of instructions in the processor pipeline resulting in less register file accesses, it does not directly target the register file. Other parts of the processor are also affected by toggling. In order to overcome this problem, we consider response mechanisms that more directly target the power consumption of the register file. By reducing the power consumption of the register file, we can bring down its maximum temperature. We consider the response of reducing the number of active functional units and the instruction window size. Reducing the number of active functional units directly affects register file temperature by decreasing the number of active register file ports. In the reactive scheme, instruction window adaptation has an indirect effect on the register file like toggling, but its impact on IPC is less severe than toggling. The reason we chose this response was to contrast the reactive algorithms with our proposed predictive algorithms where instruction window adaptation has a direct impact on the register file (see Section 4.1). In this scheme, when the processor resumes safe thermal operation, the number of active functional units is increased. Similar to R-Toggle, this algorithm is also manually tuned as described in 3.1.

3.1 Manual tuning for reactive algorithms

The reactive mechanisms are invoked when the processor temperature comes within a certain bound of the chip packaging limit. Different reactive mechanisms used on different applications require different bounds to ensure safe operation. In choosing the reaction bound, we must make sure that it is not too large, which though ensuring safe operation, will take away some potential performance, nor too small, which will result in better performance but may cause thermal emergencies. For the sake of fair comparison, the reactive adaptation trigger temperatures chosen for each combination of the reactive algorithms and applications were manually determined to ensure the best possible performance which was also thermally safe.

4 Predictive DTM Algorithm

As discussed in the previous section, reactive DTM mechanisms suffer from certain inherent disadvantages. We propose a predictive DTM algorithm which avoids these problems. Our predictive algorithm is inspired by, but is different from a recent energy saving algorithm for multimedia applications proposed by Hughes et al. [10].

Our predictive DTM algorithm uses architectural adaptation and DVS to run at thermally safe, but performance effective levels. Before discussing the algorithm, we discuss the application characteristics exploited by the algorithm and some other assumptions used. (Recall that multimedia applications typically process discrete units of data called a frame. Some of the applications studied statically distinguish multiple frame types, e.g., I, P, and B frames for MPEG). Our algorithm uses two results for multimedia and communications applications from [9] and [10] as follows:

- For a given application, architecture, and frequency, average IPC and average power dissipation of a frame are almost constant among all frames of the same type. This is because while the amount of work per frame may vary, the nature of the work is roughly the same for all frames of a given type.
- IPC of a frame is almost independent of clock frequency, since little time is spent in memory stalls.

We assume hardware recognizes when a new frame begins, and the type of the frame.

As mentioned, our algorithm uses architectural adaptation and DVS. In our description of the algorithm below, we call the possible architectures *architectural configurations* and the possible combinations of architectures and voltage/frequency *hardware configurations*.

4.1 Predictive algorithm

Our algorithm invokes adaptations at the granularity of a frame. At the beginning of each frame, the algorithm predicts the most performance effective hardware configuration (i.e., the architecture and voltage/frequency) that is also thermally safe.

The algorithm starts with the application in a *profiling phase*. In this phase, it measures the IPC and maximum temperature reached (T_{max}) for an appropriate number of frames of each type for each architectural configuration [1] . In order to ensure that there is no thermal emergency during profiling, it is performed at the lowest supported frequency and voltage (that is, we assume that the hardware supports at least one low

[1]The number of frames profi led should cover a time period of several RC time constants - we beleive that this would still be smaller than the OS scheduling time-slice

frequency/voltage configuration that is thermally safe). We assume and later validate that T_{max} is almost constant for different frames of the same type for a given hardware configuration (since IPC and power dissipation are constant and the nature of the dominant computation is constant [9]). The IPC and T_{max} for the measured set of frames are used as predictions of that architecture for all frames of that type.

Based on the temperature model discussed in Section 2, we can scale T_{max} with voltage and frequency using the relationship shown in Section 2. Using this scaling, the algorithm finds the maximum supported frequency, f_{max}, at which the system thermal constraints are still satisfied for that architecture (i.e., the f_{max} at which the T_{max} is just below the thermal limit of the system). If the f_{max} calculated is not supported in the system, the closest supported frequency lower than f_{max} is used. Since our multimedia applications spend little time in memory stalls, IPC remains almost constant across frequencies [9] obviating the need to scale IPC.

The performance of a given hardware configuration is proportional to the product of its frequency and IPC. Hence, for a given thermal limit, the algorithm ranks architectures in the order of corresponding $f_{max} \times IPC$, and chooses the architecture with the highest $f_{max} \times IPC$ product. This architecture running at f_{max} is predicted to be the fastest thermally safe hardware configuration and is used in the rest of the run. During this run,if there is a significant change in the behavior of the application and there is a change in the IPC and T_{max}, reprofiling can be performed.

Although we expect our predictions to be thermally accurate, there is always the possibility that the hardware configuration chosen is inappropriate, causing the chip to enter thermal crisis. Mechanisms to handle thermal crisis are required in any real system with DTM - in order to avoid permanent damage to the chip, drastic measures like shutting down the processor will be invoked in the event of a thermal crisis. After the crisis has been handled, the predictive algorithm can reprofile the application and select a new hardware configuration. It should be noted that reactive algorithms will also require a similar thermal crisis mechanism.

Based on the exact adaptation mechanisms used, we evaluate three predictive algorithms:

- **P-Arch**

  This predictive algorithm can be used in systems which only support architectural adaptation and not DVS. The algorithm would profile the entire architectural adaptation space and choose the fastest thermally safe architecture. As mentioned previously, we assume that the hardware supports at least one low frequency/voltage configuration for each architecture that is thermally safe.

  Just like in the reactive algorithm R-IwFu, the architectural adaptations modeled in P-Arch are instruction window adaptation and functional unit adaptation. One important difference in the effect of instruction window adaptation as invoked in the predictive algorithm compared to the reactive algorithm is that the former is also able to change the number of active physical registers with instruction window size. A smaller instruction window requires fewer physical registers. Therefore, with the predictive algorithm, the number of active physical registers, of each type, is equal to the number of logical registers (64 for each type) plus the number of entries in the instruction window. We do not allow the reactive algorithm to control the register file size because it is not clear how to do so with our processor model (reducing the size requires "garbage collecting" register contents during the course of the execution of a frame) [14] . This is not a problem with the predictive algorithm because the adaptations are invoked before the start of a frame; i.e., before any state is accumulated in the registers. This distinction is of particular importance in our simulations because the register file was found to be the hottest structure on chip.

- **P-DVS**

  This predictive algorithm can be used in systems which only support DVS and not architectural adaptation like the Transmeta Crusoe processors [7] and the Intel XScale processors [11]. The algorithm would profile the application at the lowest frequency. The highest thermally safe frequency, f_{max}, can then be determined and would be used in the adaptation phase.

- **P-ArchDVS**

  Our predictive algorithm will work best on systems which support both architectural adaptation and DVS. P-ArchDVS will always perform better than P-Arch or P-DVS.

In the following sections, we describe the experimental methodology used and compare the performance of the different reactive and predictive schemes discussed.

5 Methodology

5.1 Architectures

The base non-adaptive processor studied is similar to the MIPS R10000 and is summarized in Table 1. We assume a centralized instruction window that integrates the issue queue and Reorder Buffer (ROB) with a separate physical register file. We also study a version of the base processor with support for continuous dynamic voltage/frequency scaling (DVS). The voltages used for each frequency were extrapolated from the information available for Intel's XScale (StrongArm-2) processor [11] as in [15]. We allow the frequency to range from 100MHz to 2.2 GHz. The corresponding voltages range from 0.7 V to 1.75 V.

Fetch-toggling is activated and deactivated at a time granularity of $1\mu sec$. For R-Toggle, we allow the toggling rate to vary over a range of 20 levels, ranging from no toggle (fetch every cycle) to full toggle (no fetch).

In R-IwFu, the instruction window size is varied in steps of 16 entries. The number of ALUs and FPUs is varied in steps of a single functional unit.

We study processors capable of adapting their instruction window size and/or the number of active functional units and issue width. The instruction window is broken into segments of 8 entries each, and at least two segments must always be active. The only restriction on the number of functional units active is that at least one integer ALU must always be active. The issue width of the processor is equal to the sum

Base Processor Parameters	
Processor speed	2.2GHz
Fetch/retire rate	8 per cycle
Functional units	6 Int, 4 FP, 2 Add. gen.
Integer FU latencies	1/7/12 add/multiply/divide (pipelined)
FP FU latencies	4 default, 12 div. (all but div. pipelined)
Instruction window (reorder buffer) size	128 entries
Register file size	192 integer and 192 FP
Memory queue size	32 entries
Branch prediction	2KB bimodal agree, 32 entry RAS
Base Memory Hierarchy Parameters	
L1 (Data)	64KB, 2-way associative, 64B line, 2 ports, 12 MSHRs
L1 (Instr)	32KB, 2-way associative
L2 (Unified)	1MB, 4-way associative, 64B line, 1 port, 12 MSHRs
Main Memory	16B/cycle, 4-way interleaved
Base Contentionless Memory Latencies	
L1 (Data) hit time (on-chip)	2 cycles
L2 hit time (off-chip)	20 cycles
Main memory (off-chip)	102 cycles

Table 1 Base non-adaptive processor.

of all active functional units and hence changes when we change the number of active functional units. We also model a delay of 5 cycles to power up an inactive functional unit or instruction window segment.

Since we adapt the issue width of the processor with functional unit adaptation, we power down the selection logic corresponding to the functional units that are powered down. Also, when a functional unit is powered down, the corresponding part of the result bus, the wake-up ports to the instruction window, and write ports to the register file are also powered down.

For predictive adaptation with DVS, we profiled all possible combinations of the following configurations (54 total): instruction window size $\in \{16,32,48,64,96,128\}$, number of ALUs $\in \{6,4,2\}$, and number of FPUs $\in \{4,2,1\}$. There was a considerable variety in the configurations chosen by the predictive algorithms for different applications. This usage data is provided in Section 7 [2]

5.2 Workload Description

Table 2 summarizes the nine applications and inputs used in this paper.

5.3 Performance and Temperature Evaluation

We use the RSIM simulator [13] for performance evaluation and we use the Wattch tool [4] integrated with RSIM for temperature measurement. Wattch assumes clock gating for all the components of the processor with 10% of its maximum power charged to a component when it is not accessed in a given cycle. We also assume that the resources that are powered down by our adaptive algorithms do not consume any power.

We adapted Wattch to track per-structure temperatures based on the temperature model discussed in Section 2. The thermal resistances and capacitances used in this paper are listed in Table 3. These are based

App.	**Type**	**Input Size**		**Base**
		Time	**Frames**	**IPC**
GSMdec	Speech codec	20s	1000	3.7
GSMenc		20s	1000	4.6
G728dec		0.63s	250	2.3
G728enc		0.63s	250	2.1
H263dec	Video codec	6s	150	3.4
H263enc		6s	150	2.3
MPGdec		8.33s	250	3.6
MP3dec	Audio	26.1s	1000	3.0

Table 2 Workload description.

[2] It may seem that profiling 54 configurations may be inordinate overhead for a real system. However, it is feasible since only one frame of each type need be profiled for each configuration, the total number of frames in a typical multimedia application is much larger than the number we need to profile (e.g., 30 frames a second for video), and the profiling can be done as part of the application's execution [14].

on the data used by Skadron et al. in [16]. Due to a lack of publicly available information, they based the capacitances and resistances on estimates obtained from the MIPS R10000 die photo. Although these are significant approximations, they are not likely to affect the main conclusions of this paper.

The power consumption of the structures modeled is sampled from Wattch every micro-second which is a small Δt, compared to the RC time constant of $100\mu sec$ (see Section 2. This power information is used to calculate the processor temperature using the equation derived in Section 2. In all our experiments, we assume that the heat sink and ambient temperature has reached a steady value of 100^oC. The individual structure temperatures are then calculated by adding their instantaneous temperature to the the base temperature of 100^oC.

Structure	Area (m^2)	R (K/W)	C (J/K)
LSQ	5.0e-7	2	5.0e-5
Inst. Window	9.0e-7	1.11	9.0e-5
Regfi le	2.5e-7	4	2.5e-5
Bpred	3.5e-7	2.86	3.5e-5
D-Cache	1.0e-6	1	1.0e-4
ALU	1.0e-6	1	1.0e-4

Table 3 **Per-structure area, thermal resistance (R_{th}, and thermal capacitance (C_{th}) estimates.**

5.4 Metrics and Evaluation

- **Performance**

  As in most DTM papers, performance is the main metric of comparison used in this paper. Different DTM algorithms are compared based on the slowdown they cause in applications. Our algorithms seek to minimize the performance impact on applications while maintaining safe thermal levels.

- **Thermal limits**

  The thermal limit is defined as the maximum safe temperature for which the chip heat sink package is designed. The thermal control algorithms discussed in this paper all strive to maintain the processor temperature below this thermal limit. If the processor crosses the thermal limit, it enters thermal crisis. We model different thermal limits in order to explore a wide range of heat sinks and ambient temperatures.

- **Cycles spent in crisis**

  A crucial metric when evaluating thermal control algorithms is cycles spent in thermal crisis, i.e., the thermal limit at which the processor will burn. Understandably, the thermal control algorithm should strive to avoid or at least minimize the cycles spent in crisis (in the latter case, other aggressive techniques might be required to ensure chip safety). In order to limit the design space evaluated, we manually tuned the trigger temperature of different mechanisms to ensure that no algorithm spent any cycles in crisis (see Section 3.1).

6 Results

6.1 Application temperature profiles

For each application, Figure 1 shows the maximum temperature on chip and the IPC over time, collected at the granularity of 1 μsec for the most base non-adaptive architecture without any thermal control mechanisms. The horizontal axis is the time in μsecs. The vertical axis is both the temperature (as maximum temperature - 100^oC) and IPC. These profiles cover the parts of the execution that showed the most variability in temperature. Table 4 lists the range, mean, and standard deviation in temperature of the 2 hottest structures on chip for each application. In our simulations, the hottest structure on chip was always the register file, and the second hottest structure was always the ALU. Table 4 also includes the corresponding IPC statistics.

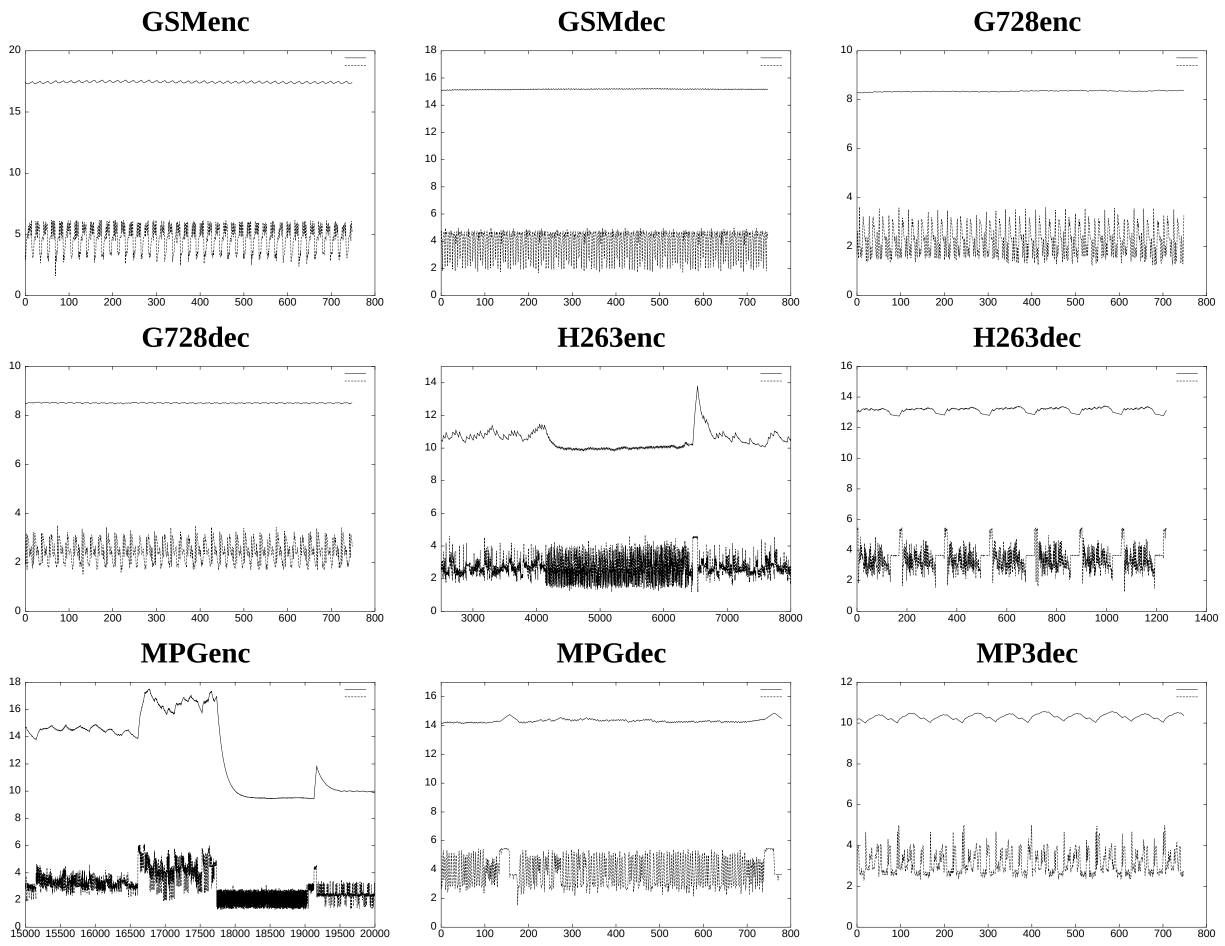

Figure 1 Maximum temperature(upper curve) and IPC profi les(lower curve) at a $1\mu sec$ granularity for the base non-adaptive processor. The y-axis is both maximum temperature (plotted as maximum temperature - 100^oC) and IPC. The x-axis shows the time in $\mu seconds$.

App.	Hottest structure (^oC)			2nd hottest structure (^oC)			IPC		
	Range	Mean	Std. Dev. (%)	Range	Mean	Std. Dev. (%)	Range	Mean	Std. Dev. (%)
GSMenc	117.29-117.62	117.44	0.40	104.93-105.02	104.97	0.40	1.61-6.16	4.72	21.74
GSMdec	115.06-115.24	115.17	0.21	114.29-114.35	114.32	0.22	1.69-5	3.92	24.82
G728enc	108.27-108.40	108.35	0.27	103.05-103.10	103.08	0.27	1.24-3.63	2.20	27.01
G728dec	108.47-108.55	108.51	0.18	103.00-103.03	103.02	0.17	1.52-3.51	2.42	18.18
H263enc	109.79-113.77	110.52	4.41	103.32-103.99	103.44	1.91	1.18-4.68	2.63	20.97
H263dec	112.76-113.42	113.14	1.24	103.58-103.79	103.70	1.28	1.29-5.46	3.46	21.97
MPGenc	109.44-119.13	114.53	14.22	103.27-105.40	104.37	10.47	1.26-6.12	3.40	26.75
MPGdec	114.13-114.86	114.33	0.91	104.07-104.28	104.19	0.75	1.53-5.49	3.92	25.25
MP3dec	110.01-110.57	110.39	1.31	103.50-103.62	103.57	0.68	2.32-5.01	3.13	18.73

Table 4 Statistics for temperature and IPC variability at a 1 μsec granularity.

As can be seen, with the exception of the video encoders, the overall variation in temperature with time is very low for all the applications. Even in the case of the video encoders, distinct temperature phases sustained over long periods of time are visible. The work performed by the video encoders takes more time and tends to be more varied than the work done by the other applications. This leads to phase behavior seen in their temperature profiles.

As is evident from the profiles, local variations seen in the IPC are smoothed out in the temperature curve. This is because of the large thermal time constant for the structures on chip (100 μsec). However, as can be seen, a significant change in IPC over a sustained period of time results in a corresponding change in temperature. Table 4 highlights this observation. The only applications with a standard deviation in temperature greater than 1.5% are the video encoders. However, all the applications experience significant standard deviation in IPC.

6.2 Reactive algorithms

Table 5 compares the performance of the two reactive schemes discussed in Section 3. For each application, the ratio of execution times of R-Toggle to R-IwFu is shown for different thermal limits. The higher the ratio, the better R-IwFu performs compared to R-Toggle. As can be seen, R-IwFu performs similar to or better than R-Toggle for all the points in the table. On average, for a thermal limit of 106^oC and lower, R-IwFu performs at least 2 times better than R-Toggle, and performs 6 times as well in one case. The performance difference decreases as we move towards less stringent thermal limits since the reactive responses are invoked less frequently.

As mentioned in the previous subsection, the hottest structure on chip is the register file. R-IwFu performs well because functional unit adaptation directly affects register file power by reducing the number of active ports in the register file. On the other hand, although toggling reduces the number of instructions in

App.	**Thermal limit. (^oC)**										
	104	**105**	**106**	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**
GSMenc	4.34	4.06	2.76	2.21	1.78	1.57	1.34	1.27	1.19	1.13	1.08
GSMdec	3.25	3.30	2.04	1.60	1.26	1.04	1.03	1.03	1.03	1.02	1.01
G728enc	4.61	2.83	1.73	1.27	1.07	1.00	1.00	1.00	1.00	1.00	1.00
G728dec	4.75	2.85	1.79	1.29	1.07	1.00	1.00	1.00	1.00	1.00	1.00
H263enc	6.00	3.93	2.29	1.72	1.41	1.23	1.05	1.01	1.01	1.01	1.00
H263dec	5.41	3.75	2.39	1.82	1.45	1.23	1.04	1.01	1.01	1.00	1.00
MPGenc	2.01	2.05	1.79	1.71	1.50	1.59	1.36	1.34	1.17	1.12	1.10
MPGdec	5.50	3.90	2.67	2.10	1.70	1.48	1.27	1.18	1.11	1.06	1.02
MP3dec	4.69	2.91	1.92	1.45	1.17	1.04	1.03	1.00	1.00	1.00	1.00
Average	4.51	3.29	2.15	1.69	1.38	1.24	1.12	1.09	1.06	1.04	1.02

Table 5 Reactive schemes - Ratio of execution time of R-Toggle to R-IwFu.

the pipeline at any given time, it does not target the register file thermal power directly. Hence, to acheive the same thermal benefit, R-Toggle results in a larger IPC drop in the application than R-IwFu. This degradation in IPC is more significant at more stringent thermal limits where reactive mechanisms are active for longer periods of time.

The performance improvement of R-IwFu over R-Toggle motivates local adaptations for temperature control. Rather than using globally restrictive schemes like R-Toggle, a local adaptation targeted at the thermal hot spot would likely result in better performance. Also, it should be noted that the register might not always be the hottest structure on chip. In such cases, local adaptations specific to the new hottest structure would have to be invoked.

We also performed experiments with only instruction window adaptation or functional unit adaptation. Each alone was better than toggling. Even though instruction window adaptation is a global mechanism, it results in a lower IPC degradation than toggling. For lack of space, we do not plot those results here. Joint instruction window and functional unit adaptation did better than either alone.

App.	**R-IwFu vs. P-ArchDVS**			**P-DVS vs. P-Arch**			**P-Arch vs. P-ArchDVS**		
	104 ^oC	**108 ^oC**	**114 ^oC**	**104 ^oC**	**108 ^oC**	**114 ^oC**	**104 ^oC**	**108 ^oC**	**114 ^oC**
GSMenc	2.40	1.53	1.12	0.89	1.08	1.05	1.31	1.02	1.00
GSMdec	2.37	1.14	1.00	0.82	0.97	1.00	1.29	1.05	1.00
G728enc	1.46	1.04	1.00	1.19	1.02	1.00	1.00	1.00	1.00
G728dec	1.55	1.07	1.03	1.26	1.04	1.03	1.00	1.00	1.00
H263enc	1.92	1.17	1.00	1.38	1.06	1.00	1.00	1.00	1.00
H263dec	2.27	1.22	1.00	1.38	1.14	1.02	1.09	1.00	1.00
MPGenc	3.62	2.10	1.00	1.65	1.35	1.10	1.00	1.00	1.00
MPGdec	2.42	1.39	1.00	1.47	1.15	0.96	1.00	1.00	1.05
MP3dec	1.72	1.12	1.03	1.04	1.01	1.00	1.05	1.00	1.00
Average	2.19	1.31	1.02	1.23	1.09	1.02	1.08	1.01	1.01

Table 6 Comparison of different schemes - ratio of execution time for different thermal limits.

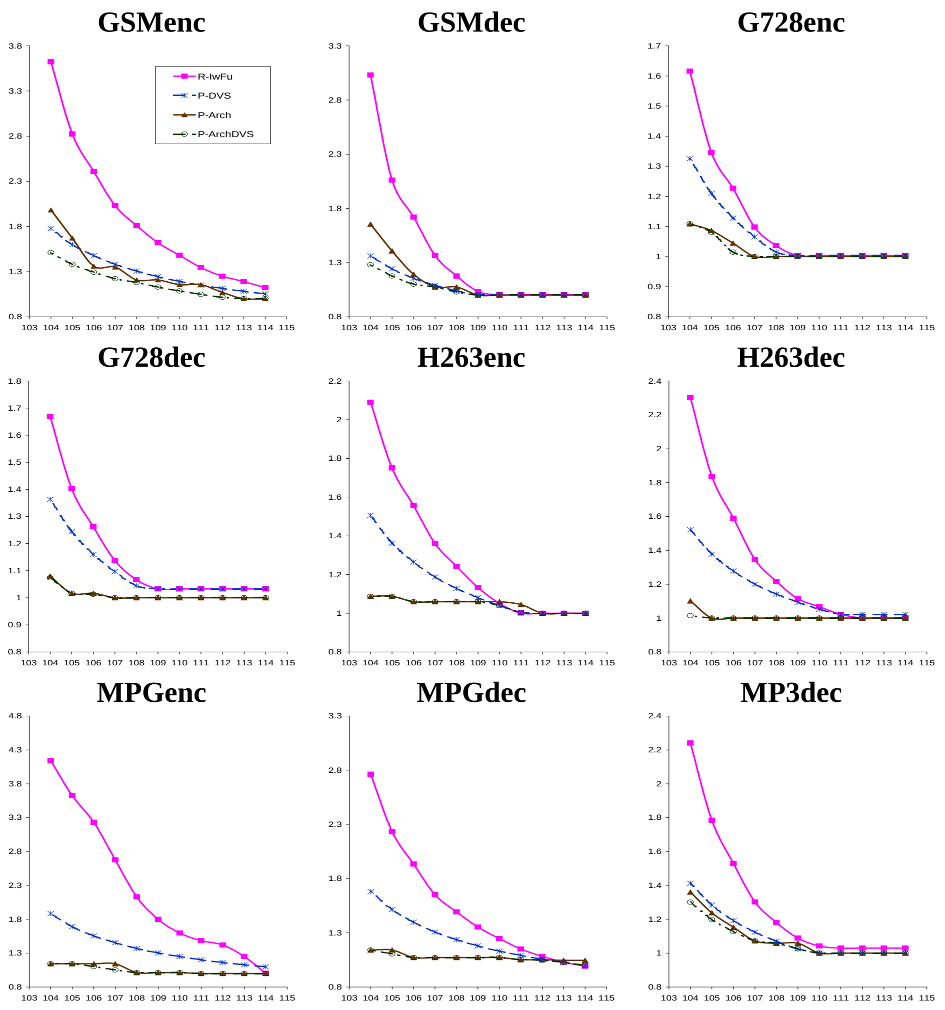

Figure 2 Predictive algorithms for all applications. The y-axis is the slowdown caused by the DTM algorithm over the base non-adaptive architecture. The x-axis shows the thermal limit in ^oC.

Thermal limit (^oC)	**P-Arch Configurations chosen**					
	1-9	**10-19**	**20-29**	**30-39**	**40-49**	**50-54**
114	GSMd,G728e,G728d,MP3d,H263e	GSMe,H263d,MPGd	MPGe			
110	GSMd,G782e,G728d,MP3d,MPGd	GSMe,H263d	H263e	MPGe		
106	G782e,MPGd	GSMd,G728d,MP3d	H263e	GSMe		MPGe
104	G782e,G728d	H263d	GSMe,GSMd,MPGd	MP3d	H263e	MPGe

Table 7 Hardware confi gurations chosen by P-Arch. Lower numbered confi gs. are more aggressive than higher ones.

6.3 Predictive algorithms

Figure 2 shows the performance of the 3 predictive schemes discussed in Section 4, P-DVS, P-Arch, and P-ArchDVS, for different thermal limits. It also shows the same data for the best performing reactive scheme, R-IwFu. The horizontal axis in the graphs corresponds to the maximum thermal limit allowed, ranging from 104^oC to 114^oC, and the vertical axis represents the performance in terms of the slowdown over the base non-adaptive architecture (Note that this is not % slowdown, but the ratio in execution times of the DTM algorithms over the base non-adaptive architecture). As the thermal limit allowed becomes stricter (from right to left on the horizontal axis), the slowdown increases (from bottom to top on the vertical axis). Table 6 provides a pairwise comparison of R-IwFu vs. P-ArchDVS, P-DVS vs. P-Arch, and P-Arch vs. P-ArchDVS for a strict, moderate and relaxed thermal limit.

As can be seen, the predictive schemes perform significantly better than the reactive schemes for all the applications including the video encoders which showed some temperature variability. On average, for a strict thermal limit of 104^oC, the best performing predictive scheme, P-ArchDVS is more than twice as fast as the best performing reactive scheme, R-IwFu. This is primarily because the predictive schemes can use high time overhead adaptations like register file resizing and DVS. These adaptations result in significant power and temperature savings, with a relatively lower penalty in performance. As a result, we find that predictive schemes perform better than reactive schemes. Also, since very little variability is seen in the temperature profiles of most applications, the hardware configuration chosen by the predictive algorithm tends to be close to optimal for the entire execution.

Among the predictive schemes, P-ArchDVS always performs the best. This is rather intuitive - all the hardware configurations available to P-Arch and P-DVS are subsets of the configurations available to P-ArchDVS. P-DVS is less efficient than P-Arch and P-ArchDVS. On average, for a strict deadline of 104^oC, P-DVS is more than 20% slower than P-Arch and 35% slower than P-ArchDVS. This is because P-DVS can not perform instruction window resizing and functional unit adaptation which directly affect the register file hotspot. Instead, it can only perform DVS, which is a global adaptation. Due to voltage scaling, DVS can reduce the register file power like functional unit adaptation. However, P-DVS is more efficient than R-IwFu.

Finally, the performance difference between different schemes reduces as the thermal limit is relaxed (from right to left on the horizontal axis in Figure 2. This is because DTM mechanisms are invoked for a smaller period of time at these temperatures leading to a smaller loss in performance.

6.4 Configurations chosen

Finally, we look at the hardware configurations chosen by the predictive algorithm, P-Arch. This helps us understand the extent to which the adaptive hardware features provided to the predictive algorithm are exploited. Table 7 lists the architectural configurations chosen by each application for each thermal limit. As explained in Section 5, P-Arch is provided with 54 architectural configurations to choose from. To conserve space, Table 7 clubs the configurations into groups. Configurations are ranked in order of instruction window size, then in the number of ALU functional units, and then FPU functional units. Hence, configurations with lower numbers are generally more aggressive than higher numbered ones. An interesting trend in the table is that lower (and consequently more aggressive) thermal limits result in less aggressive configurations being chosen. This result is rather intuitive. Again, this is because the architectural adaptation in the predictive algorithm does more to reduce the register file hotspot.

As can be seen in Table 7, no single architectural configuration is best for all applications and temperatures. A given processor may be designed for multiple applications and thermal limits (the thermal limit as we use the system can vary depending on the heat sink used or the ambient temperature since the thermal limit is set with respect to a fixed ambient temperature). Table 7 indicates that such a processor would utilize a considerable range of architectural configurations, clearly motivating architectural adaptation for thermal control.

Similar results were seen with P-ArchDVS.

7 Discussion and future work

- Although our evaluation has focused on multimedia applications, we feel that this work could apply to other general purpose workloads like the SPEC benchmark suite. Because of the large thermal time constants of the structures on chip, significant changes in IPC over a sustained period in time are required to create noticeable changes in temperature. Smaller variations in IPC tend to get smoothed out in the temperature curve. As a result, we beleive that there is potential for predictive algorithms to be applied to applications that do not possess the regular computation nature of multimedia applications also. To apply such predictive algorithms to other workloads, we would need to profile for the appropriate macro-level phases in the workload execution run. We seek to analyze the potential of predictive DTM for other workloads in the future.

- Although our predictive algorithm performs significantly better than reactive DTM algorithms, there is still room for improvement in performance. Based on the maximum temperature reached during profiling, the predicitive algorithm chooses a hardware configuration that would be safe during the enture application run. However, most of the time, this configuration would be more conservative than necessary, and only at a few points during the execution will the maximum thermal limit be reached. This motivates the need for a fine-grained reactive algorithm piggy backed on our predictive algorithm. This algorithm would try to maximize performance while ensuring that it does not take the processor into thermal emergency. This is a promising line of future work.

- Our thermal control algorithm can be combined with the energy saving algorithm proposed by Hughes et al. [10], with a joint view to save energy and ensure thermal safety.

- Finally, our localized reactive mechanisms were found to be effective because of the register file hotspot in our simulations. However, the register file might not always be the hottest structure on chip. Hence, different local mechanisms targeted at different potential chip hot spots should be explored in future work.

8 Conclusions

Thermal power management is an important area of research for general purpose processors. In order to save on thermal packaging costs, DTM mechanisms have been proposed. However, these techniques result in a degradation in processor performance when invoked. In this paper, we make two contributions in the context of DTM mechanisms for the increasingly important workload of multimedia applications. First, in contrast to the current reactive schemes, we propose a predictive DTM algorithm that exploits certain properties of multimedia applications. We found our predictive algorithm to perform significantly better than existing reactive DTM algorithms, performing twice as well as reactive algorithms on average for the strictest thermal limit and upto 3.6 times as well in some cases. This is because the predictive algorithm can use high time overhead adaptations like register file resizing and DVS which result in significant thermal benefit for a limited loss in performance. Also, our analysis found that there is very little variation in the temperature profiles of several of our multimedia applications. This implies that the hardware configuration chosen by our predictive algorithm will remain close to optimal for the entire application execution run.

Our second contribution is in the evaluation of DTM response mechanisms for the processor core using accurate thermal models and local ”hot-spots” (as proposed by Skadron et al. [16]) for multimedia applications on general-purpose processors. Previous work on processor temperature management has either used power as a proxy for temperature [8, 3] or has considered only one response mechanism (fetch-toggling) [16]. Further, none of the previous work has focused on multimedia applications. We evaluated the benefit of using local adaptations targeted at the local hot spot rather than using chip-wide thermal techniques. We found that local adaptations performed considerably better than global chip wide adaptations for our systems and applications. For our register file hot-spot, we found that functional unit adaptation and instruction window resizing performed upto 6 times as well as fetch-toggling.

DTM is still a nascent field and there are many promising avenues of future work. We have detailed some of this work in our discussion section. In particular, we beleive that our predictive approach could be modified to apply to the non-multimedia workload domain (due to the phased behavior observed in temperature). Further, there is scope for a combined predictive/reactive algorithm for applications with multiple temperature phases.

References

[1] In *International Technology Roadmap for Semiconductors, http://public.itrs.net/*, 1999.

[2] S. Borkar. Design Challenges of Technology Scaling. In *IEEE Micro, July-August 1999*, 2000.

[3] David Brooks and Margaret Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. In *Proc. of the 7th Intl. Symp. on High-Performance Comp. Architecture*, 2001.

[4] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proc. of the 27th Annual Intl. Symp. on Comp. Architecture*, 2000.

[5] A. Dhodapkar et al. Tempest: A thermal enabled multi-model power/performance estimator. In *Proc. of the Workshop on Power-Aware Computer Systems*, 2000.

[6] Keith Diefendorff and Pradeep K. Dubey. How Multimedia Workloads Will Change Processor Design. *IEEE Computer,* September 1997.

[7] Tom R. Halfhill. Transmeta Breaks x86 Low-Power Barrier. *Microprocessor Report*, February 2000.

[8] Michael Huang, Jose Renau, Seung-Moon Yoo, and Josep Torrellas. A Framework for Dynamic Energy Efficiency and Temperature Management. In *Proc. of the 33rd Annual Intl. Symp. on Microarchitecture*, 2000.

[9] Christopher J. Hughes et al. Variability in the Execution of Multimedia Applications and Implications for Architecture. In *Proc. of the 28th Annual Intl. Symp. on Comp. Architecture*, 2001.

[10] Christopher J. Hughes, Jayanth Srinivasan, and Sarita V. Adve. Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications. In *Proc. of the 34th Annual Intl. Symp. on Microarchitecture*, 2001.

[11] Intel XScale Microarchitecture. http://developer.intel.com/design/intelxscale/benchmarks.htm.

[12] Christoforos E. Kozyrakis and David Patterson. A New Direction for Computer Architecture Research. *IEEE Computer*, November 1998.

[13] Vijay S. Pai, Parthasarathy Ranganathan, and Sarita V. Adve. RSIM Reference Manual version 1.0. Technical Report 9705, Department of Electrical and Computer Engineering, Rice University, August 1997.

[14] Ruchira Sasanka, Christopher J. Hughes, and Sarita V. Adve. Joint Local and Global Hardware Adaptations for Energy. In *Proc. of the 10th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2002.

[15] G. Semeraro et al. Energy-Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling. In *Proc. of the 8th Intl. Symp. on High-Performance Comp. Architecture*, 2002.

[16] K. Skadron et al. Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management. In *Proc. of the 8th Intl. Symp. on High-Performance Comp. Architecture*, 2002.

[17] K. Skadron et al. HotSpot: Techniques for Modeling Thermal Effects at the Processor-Architecture Level. In *THERMINIC*, 2002.