The Importance of Heat-Sink Modeling for DTM and a Correction to "Predictive DTM for Multimedia Applications"

Jayanth Srinivasan and Sarita V. Adve
University of Illinois at Urbana Champaign
Department of Computer Science
{srinivsn,sadve}@cs.uiuc.edu

Abstract

It is anticipated that in the future, increasing power densities and resultant temperatures will often be the dominant limit to processor performance and a significant component of cost. Dynamic thermal management (DTM) techniques address this issue by allowing the thermal solution to be designed for a temperature less than the peak. Evaluating DTM and other processor thermal properties requires reliable tools for processor thermal modeling. In particular, the processor heat-sink model plays a crucial role, but much prior work in DTM, including our own, is based on simplistic assumptions of heat-sink behavior. Using more realistic heat sink simulation, this paper corrects our previous work and underscores the need for more accurate heat sink simulation for future thermal work.

Specifically, we re-examine the DTM results in our paper, "Predictive Dynamic Thermal Management for Multimedia Applications" [12]. We show that although the predictive DTM algorithms discussed in that paper are still effective, incorrect heat sink simulation resulted in erroneous conclusions on the effectiveness of global vs. local response mechanisms. We also describe how for reactive DTM algorithms (which constitute all of the prior DTM work), heat sink simulation is more complex than previously discussed.

1 Introduction

Temperature is emerging as a significant issue in modern microprocessor design. It is anticipated that power consumption and consequent thermal considerations will be the dominant limit to processor performance in many future systems. With increasing processor power densities, cooling solution costs are increasing rapidly, making cost-effective deployment of future computer systems difficult [1].

In order to combat rising cooling solution costs, researchers have proposed *dynamic thermal management* (DTM) [3, 5, 10, 11, 12]. DTM refers to a range of possible hardware strategies to control a chip's operating temperature at runtime. Traditionally, the cooling solution for a processor is designed to maintain a safe operating temperature even when the chip is dissipating the maximum power possible for a sustained period of time, and therefore at its highest (or peak) temperature. However, this worst case scenario is unlikely during normal processor operation and such expensive packaging is overkill. DTM provides a more cost-effective solution by allowing the thermal solution to be designed for a temperature that is less than the peak. In the (hopefully) rare case when the chip approaches the packaging temperature limit, DTM invokes a hardware response to bring down the temperature, typically by reducing system performance. Possible response mechanisms include a variety of architectural adaptations (e.g., fetch toggling or throttling) and dynamic voltage scaling (DVS).

In order to evaluate DTM techniques, accurate and reliable simulation tools for processor thermal modeling are required. In particular, since the processor heat-sink temperature affects the temperature of every structure on chip, the thermal properties of the heat-sink play a crucial role in processor thermal evaluations. However, much of the prior microarchitectural work on DTM, including our own, is based on simplistic assumptions of heat-sink behavior.

This paper makes two contributions. First, it proposes an iterative method for improved heat sink modeling when simulating reactive DTM algorithms. Second, it uses updated heat-sink models to re-evaluate the DTM results in our paper, "Predictive Dynamic Thermal Management for Multimedia Applications" [12]. We show that although the predictive DTM algorithms in that work are still effective, some of the conclusions drawn are erroneous due to incorrect heat-sink simulation. Thus, our results highlight the importance of appropriate heat-sink modeling for future thermal studies.

1.1 Modeling the Heat-Sink

Heat-sink temperature simulation is an essential part of any processor thermal simulation. As discussed in [11], the

large thermal time constant of the heat-sink makes its temperature simulation difficult. Due to its large thermal capacity, the time constant of the heat-sink (which dictates the rate at which its temperature changes) tends to be of the order of 10s of seconds. This time granularity is much higher than the time of most architectural simulations, which usually simulate less than one second of real processor time. As a result, most architectural simulations will not run long enough for the heat-sink to reach its steady state temperature on its own. Instead, it is essential that the heat-sink temperature be *initialized to the correct value* **before** *the start of the thermal simulation*.

Some previous thermal models (e.g., [10]), ignored the processor heat-sink temperature. Instead, an **arbitrary** constant heat-sink temperature is assumed for all simulations. This is clearly inaccurate. More recently, Skadron et al. proposed a more advanced thermal model, HotSpot [11], which is recognized as state-of-the-art among current publicly available models. HotSpot addresses many of the shortcomings in previous thermal simulators and also models the thermal properties of the heat-sink. However, even using Hotpot, it is essential that the heat-sink temperature be initialized appropriately for accurate thermal simulations.

The temperature of the processor heat-sink depends on the average power consumption of the processor calculated over periods smaller than the heat-sink's thermal time constant. Applications and architectures with higher power consumption will result in a hotter heat-sink. For an accurate thermal simulation, the initial temperature of the heat-sink, which has to be set before the run, should be calculated using the average processor power consumption, which can only be determined after the run. The authors of [11] address this issue for non-adaptive processors by running every simulation twice – the first run is used to calculate the average total power consumption of the chip, from which the steady-state temperature of the heat-sink is calculated. The heat-sink is then initialized with this value for the second run, in which temperature is modeled accurately. This method is only valid for thermal simulations of non-adaptive processors. As discussed next, DTM algorithms which use processor adaptation are more difficult to simulate.

Most work in DTM for processors is *reactive* – the processor typically runs at full performance; only when the temperature gets too close to the packaging limit, a response to curtail performance (and thereby reduce temperature) is initiated. Reactive DTM algorithms add additional complexity to heat-sink modeling. There exists a cyclical dependence between the response of a reactive DTM algorithm and the temperature of the heat-sink. The response of a reactive DTM algorithm depends on the temperature of the hottest structure on chip which depends on the temperature of the heat-sink. At the same time, the DTM response will change the average processor power consumption, which will change the steady state temperature of the heat sink. Hence, since the average power consumption changes during the reactive DTM simulation, the calculated heat-sink temperature at the end of the simulation will be different from the initialization temperature, resulting in an inaccurate simulation. Note that this problem does not arise in situations where there is no DTM as the average power consumption of the processor does not change.

To account for this effect, we propose that reactive DTM algorithms be simulated over multiple iterations. At the beginning of each iteration, the heat-sink temperature is initialized based on the average processor power consumption in the previous iteration. Over multiple iterations, the heat-sink temperature and average power consumption due to the DTM responses will converge to commensurate levels, resulting in an accurate simulation. Most previous work on DTM algorithms is either based on simplistic heat-sink assumptions like arbitrary constant temperatures or does not describe the heat-sink temperature initialization methodology [4, 9, 10, 11].

1.2 Re-evaluation of DTM Algorithms in [12]

We illustrate the importance of correct heat-sink modeling by re-examining the DTM results in our paper, "Predictive Dynamic Thermal Management for Multimedia Applications" [12]. The paper exploited certain properties of multimedia applications to propose new performance effective DTM control algorithms for such applications.

As mentioned, most previous work on DTM proposes reactive algorithms. Such schemes suffer from at least two problems: (1) Given the limited time to respond to a thermal emergency in a reactive scheme, only low time overhead mechanisms can be applied. This precludes the efficient use of mechanisms like dynamic voltage scaling which potentially have large thermal benefits but high invocation time overheads. (2) 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 for a given application.

In response to the above problems, [12] proposed new DTM control algorithms for long running multimedia applications that took a *predictive* approach, by exploiting characteristics of multimedia applications. We evaluated the predictive algorithms using the first thermal model proposed by Skadron et al. [10]. As mentioned, that thermal model assumes an arbitrary constant heat-sink temperature for all simulations. Based on an evaluation of nine multimedia applications, [12] concluded that predictive DTM algorithms were the most performance-effective, and that DTM response mechanisms targeted at specific "hot-spots" on chip were more effective than global DTM response

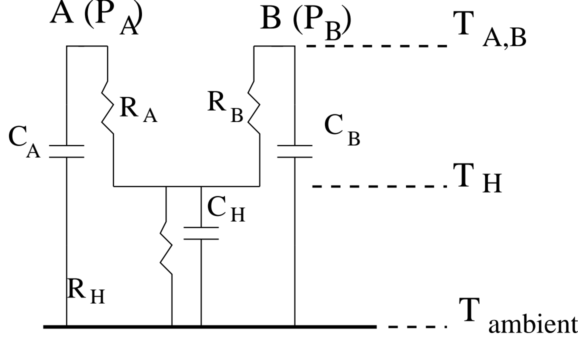

Figure 1. Equivalent thermal circuit for simple processor with two structures.

mechanisms.

In this paper, we re-evaluate the DTM algorithms from [12] using the HotSpot thermal model [11]. We also use the iterative approach discussed in the previous section to properly evaluate reactive DTM algorithms. Predictive DTM algorithms do not require iterative simulation as the DTM response is invoked only once at the beginning of the simulation. Our revised results in this paper show that although predictive DTM algorithms are still the most performance effective, global DTM response mechanisms are more effective than techniques targeted exclusively at hotspots on chip. This difference in results arises almost completely from the improved modeling of heat-sink behavior, highlighting the role of the heat-sink in processor thermal evaluations.

2 Analytical Heat-Sink Model

For each structure of a processor, the power dissipated translates into a temperature increase with a certain time lag. This is equivalent to a conventional RC electric circuit, where the structure has a certain thermal resistance R (Kelvin/Watt), and a certain thermal capacitance C(Joule/Kelvin).

Consider a simple processor with two structures, A and B, and a heat sink. The equivalent thermal circuit for the processor is shown in Figure 1. For the sake of simplicity, in this section, we ignore tangential resistances and capacitances across structures A and B in Figure 1. The steady state temperature of structure A, T_A, is given by

$$T_A = T_H + R_A \times P_A \qquad (1)$$

where T_H is the temperature of the surface of the heat-sink, P_A is the power being dissipated by structure A, and R_A is the thermal resistance of structure A. The steady state temperature of structure B can be calculated similarly. The thermal RC time constant ($R \times C$) dictates the rate at which the temperatures of the structures change. Structures with higher RC time constants take longer to change temperature.

In a similar fashion, the steady state temperature of the heat-sink depends on the total power consumption of the processor, $P_A + P_B$. Hence, the temperature of the surface of the heat-sink, T_H is given by

$$T_H = T_{ambient} + (P_A + P_B) \times R_H \qquad (2)$$

where $T_{ambient}$ is ambient room temperature and is a constant, and R_H is the thermal resistance of the heat-sink. If C_H is the thermal capacitance of the heat-sink, the rate at which the heat-sink temperature changes depends on $R_H \times C_H$.

The time constants associated with structures on chip are much smaller than the time constant of the heat sink. As a result, as recommended by [10], [12] assumed that the dynamic aspects of the heat sink for short time intervals can be ignored, resulting in a constant heat sink temperature. Instead, we only calculated dynamic variations in temperature of individual blocks assuming a constant arbitrary value for T_H. This can lead to incorrect temperature estimates. For example, if the power consumption of structure A increases from P_{A_1} to P_{A_2}, the increase in steady state temperature of A, ΔT_A is

$$\Delta T_A = (P_{A_2} - P_{A_1}) \times R_A + (P_{A_2} - P_{A_1}) \times R_H \qquad (3)$$

If a constant heat-sink temperature is assumed, the steady state temperature of structure A would be overestimated by $(P_{A_2} - P_{A_1}) \times R_H$. This error increases if multiple structures on chip change power consumption. In addition to structure A, if the power consumption of structure B increases from P_{B_1} to P_{B_2}, the increase in steady state temperature of A is

$$\Delta T_A = (P_{A_2} - P_{A_1}) \times R_A + ((P_{A_2} + P_{B_2}) - (P_{A_1} + P_{B_1})) \times R_H \qquad (4)$$

Hence, the increase in power consumption of *both* A and B affect the temperature of structure A due to the effect of the heat-sink. This is of particular importance when examining techniques which target individual “hot-spots” on chip versus global techniques which target every structure on chip. Hence, even if a global technique decreases the power of a structure less than a local structure-specific technique, the global technique might be more effective in reducing the temperature of the structure due to the impact of the heat-sink.

2.1 Effect of DTM on Heat-Sink Temperature

As discussed in Section 1, there exists a cyclical dependence between the response of a reactive DTM algorithm and the temperature of the heat-sink. As seen in Equation 1, the temperature of the hottest structure on chip depends on the temperature of the heat-sink, T_H. At the same time, as seen in Equation 2, the temperature of the heat-sink depends on the power consumption of every structure on chip. If the DTM response reduces average processor power, T_H will decrease, and if the DTM response increases average processor power, T_H will increase. Due to the large thermal time constant of the heat-sink, this change in T_H will not be seen during the simulation. In other words, in the case of Figure 1, the average power consumption of A and B, P_A and P_B, while running the reactive DTM algorithm will not satisfy Equation 2 as the heat-sink temperature, T_H will be incorrect.

To account for this effect, we propose that reactive DTM algorithms be simulated over multiple iterations. At the beginning of each iteration, the heat-sink temperature is initialized based on the average total processor power consumption in the previous iteration. Over multiple iterations, the heat-sink temperature and DTM responses will converge to commensurate levels, resulting in an accurate simulation. Hence, after multiple iterations, values of P_A, P_B, and T_H will be arrived at that satisfy Equation 2. The DTM simulation at this point will provide accurate results.

When simulating predictive DTM algorithms or any non-adaptive processor, this problem does not arise. Hence, two iterations are sufficient for accurate thermal simulation.

3 DTM Algorithms Studied

In this section, we briefly summarize the DTM algorithms evaluated in [12]. We re-evaluate all of these algorithms in this paper.

3.1 Reactive DTM Algorithms

We evaluated two reactive DTM algorithms in [12], **R-Toggle** and **R-IwFu**.

3.1.1 R-Toggle

R-Toggle uses fetch-toggling as its response mechanism. Fetch-toggling reduces the number of instructions in the processor pipeline by reducing the rate at which instructions are fetched. Hence, with fetch-toggling, for every N processor cycles, instructions are fetched in only M of the cycles, where $M < N$. Once the trigger temperature is crossed, the toggling rate is varied linearly from no toggling ($M = N$ or normal fetch) to full toggling ($M = 0$ or no fetch) depending on the proximity to the thermal limit.

3.1.2 R-IwFu

R-IwFu targets the register file which was found to be the hottest structure on chip for all our applications. Specifically, R-IwFu deactivates functional units, which results in a reduced number of active register file ports, reducing register file power. R-IwFu also resizes the instruction window. In the reactive scheme, this adaptation has an indirect effect on the register file. 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. When the processor temperature exceeds the trigger temperature, the instruction window size and the number of functional units are reduced by a fixed amount at each temperature sample. Once the temperature goes below the trigger, the instruction window and functional units are fully activated again.

For the fairest comparison, the trigger temperatures chosen for the reactive algorithms for each application for each thermal limit were manually determined to ensure the best possible performance which was also thermally safe.

3.2 Predictive DTM Algorithms

Multimedia applications typically process discrete units of data called frames. Our predictive algorithms use two prior results for such applications [6]: (1) For a given application, architecture, and frequency, average IPC and average power consumption of a frame are almost constant among all frames of the same type. This is because the nature of the work is roughly the same for all frames of a given type. (2) IPC of a frame is almost independent of clock frequency, since little time is spent in memory stalls. From these observations, since the IPC and power dissipation of a frame are constant, and the nature of the dominant computation is constant, we extrapolate and confirm that the maximum temperatures attained during different frames of the same type will be similar.

The goal of our predictive algorithm, summarized in Figure 2, is to determine the highest performing, thermally safe architectural configuration. The algorithm starts with the application being profiled at a frequency, f_{prof}. For each architectural configuration, A, the algorithm measures IPC_A (the IPC of A), and maximum temperature reached ($T_{max|A,f_{prof}}$) by any structure on chip for an appropriate number of frames of each type. Next, the algorithm determines the maximum frequency, $f_{max|A}$ at which each profiled architecture is still thermally safe. This can be determined using the maximum temperature reached during profiling $T_{max|A,f_{prof}}$, the profiling frequency, f_{prof}, and profiling voltage, V_{prof}, as follows:

For each frame type,

For each architecture, A:

(1) Measure IPC_A and $T_{max|A,fprof}$ (at a common safe profiling voltage/frequency, V_{prof} / f_{prof})

(2) For the thermal limit, T_{limit}, find the maximum safe frequency, $f_{max|A}$, using:

$$\frac{T_{limit}}{T_{max|A,fprof}} = \frac{V^2_{max|A} f_{max|A}}{V^2_{prof} f_{prof}}$$

(3) If $f_{max|A}$ < min. supported freq., then discard architecture A.

(4) If $f_{max|A}$ > max. supported freq, then $f_{max|A}$ = max. supported freq.

(5) $Performance_A \propto IPC_A \times f_{max|A}$

Choose architecture, A, with highest performance, running at $f_{max|A}$.

Figure 2. The predictive algorithm for choosing the best performing thermally safe configuration.

$$\frac{T_{limit}}{T_{max|A,f_{prof}}} = \frac{V^2_{max|A} f_{max|A}}{V^2_{prof} f_{prof}} \quad (5)$$

where $V_{max|A}$ is the voltage required to support $f_{max|A}$, and T_{limit} is the chip's thermal limit (i.e., maximum temperature than any structure is allowed to reach).

The performance of a given hardware configuration is proportional to the product of its frequency and IPC. Since our multimedia applications spend little time in memory stalls, IPC remains almost constant across frequencies [6] obviating the need to scale IPC with frequency. Hence, the maximum thermally safe performance of an architecture is proportional to $f_{max|A} \times IPC_A$. The algorithm therefore chooses the architecture, A, with the highest $f_{max|A} \times IPC_A$ product. This architecture running at $f_{max|A}$ is predicted to be the fastest thermally safe hardware configuration and is used in the rest of the run.

The predictive algorithms can exploit toggling, structural adaptation, and DVS. To isolate the benefits of each, we evaluate four versions: **P-Toggle** only performs toggling and does not support any other form of architectural adaptation or DVS. **P-IwFu** performs instruction window and functional unit adaptation and does not support DVS. **P-DVS** adapts the voltage and frequency, but not the architecture. In order to include all the algorithms evaluated in [12], we also study **P-IwFuDVS** which combines the benefit of P-IwFu and P-DVS. 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 for renaming, resulting in lower power consumption.

Technology Parameters	
V_{dd}	1.75 V
Processor frequency	2.2 GHz
Base Processor Parameters	
Fetch/retire rate	8 per cycle
Functional units	6 Int, 4 FP, 2 Add. gen.
Integer FU latencies	1/7/12 add/mult./div.
FP FU latencies	4 default, 12 div. (not 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.

4 Methodology

4.1 Architectures

Our experimental methodology is very similar to the one used in [12]. The base non-adaptive processor studied is similar to the MIPS R10000 and is summarized in Table 1. We also study a version of the base processor with support for dynamic voltage/frequency scaling (DVS). The voltages used for each frequency were extrapolated from the information available for Intel's XScale (StrongArm-2) processor [8]. We allowed the frequency to range from 100 MHz (at 0.7 V) to 2.2 GHz (at 1.75 V).

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. As mentioned earlier, for the predictive algorithm, we resize the register file based on the instruction window size. For functional units, we require that at least one integer ALU must always be active. The issue width of the processor is equal to the sum of all active functional units and hence changes when we change the number of active functional units. 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. We model a delay of 5 cycles to power up an inactive functional unit or instruction window segment. When a functional unit or instruction window segment is to be powered down, the sys-

App.	Type	No. Frames Profiled	Cycles per Frame	No. Frames Executed
GSMenc	Speech codec	10	8.065+E4	100
GSMdec		10	2.002+E4	100
G728enc		10	9.498+E3	100
G728dec		10	7.415+E3	100
H263enc	Video codec	1	1.544+E7	1
H263dec		1	3.431+E5	5
MPGenc		1	3.405+E7	1
MPGdec		1	1.313+E6	1
MP3dec	Audio	1	6.421+E5	10

Table 2. Workload description. For MPG, only B frames are evaluated.

tem must wait for the units to complete their current tasks before shutting them down. As a result, we do not charge an extra delay for powering down the functional units and instruction window.

For predictive adaptation, 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 $\in \{4,2,1\}$. P-Toggle profiles twenty architectures, each with a different fetch level.

The reactive algorithms sample the system temperature every 1μsec. As a result, reactive adaptations are also invoked at this granularity. 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.

4.2 Workload description

Table 2 summarizes the nine applications and inputs used in this paper. For each application, it gives: (1) the average number of execution cycles per frame, (2) the number of frames profiled for each architectural configuration and frame type (the total time profiled should be much larger than the thermal RC constant for thermal stability), (3) number of frames executed after profiling for evaluating the performance degradation of the DTM schemes.

We do not study multiprogrammed workloads in this paper. Multiprogrammed workloads add an extra layer of complexity to DTM thermal simulations due to the difference in power consumption across different applications for the same thermal limit.

4.3 Simulation Methodology

4.3.1 Simulator

We use the RSIM simulator [7] for performance evaluation. We use the Wattch tool [2] integrated with RSIM for power measurement. Wattch assumes extensive 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.

As explained earlier, we use the HotSpot tool [11] for temperature measurement. The chip floorplan fed to HotSpot resembles the MIPS R10000 floorplan (without L2 cache). The areas of the different structures modeled can be found in [12]. HotSpot represents the state-of-the-art in publicly available thermal simulators and accounts for tangential resistances between structures on chip. In addition, the impact of total processor power on the heat-sink temperature is also accounted for. Temperature measurements are performed at the granularity of 1μ second.

As explained in Section 2, care has to be taken to initialize HotSpot with the appropriate heat-sink temperature, depending on the nature of the thermal simulation. Predictive DTM simulations require two iterations for accurate results. Reactive DTM simulations require multiple iterations. The heat-sink temperature at the first iteration for every thermal limit is initialized to 40 degrees. Each subsequent iteration is initialized with a heat-sink temperature calculated from the power-consumption of the processor in the previous iteration. This process is continued until the difference in heat-sink temperature across adjacent iterations is less than 0.2 degrees.

4.3.2 Thermal limits

We define the thermal limit as the maximum allowed temperature of the chip. The thermal control algorithms discussed in this paper all strive to maintain the processor temperature within this thermal limit. If the processor crosses the thermal limit, it enters thermal crisis. Since it is desirable to design processors to work with a variety of thermal solutions, we model multiple thermal limits ranging from 55^oC to 105^oC.

4.4 Metrics

We use performance as the main metric of comparison. Our algorithms seek to minimize the performance impact while maintaining safe thermal levels. Additionally, the thermal control algorithm should strive to avoid or at least minimize the cycles spent in crisis. As a result, a crucial metric when evaluating thermal control algorithms is cycles spent in crisis. 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*.

App.	Adapt.	Thermal limit. (^oC)											
		55	Δ	**65**	Δ	**75**	Δ	**85**	Δ	**95**	Δ	**105**	Δ
GSMenc	Toggle	35.0	*1.0*	36.9	*2.3*	40.3	*2.9*	44.7	*0.6*	50.0	*2.0*	61.7	*4.1*
	IwFu	40.9	*0.3*	43.0	*4.5*	47.0	*4.3*	52.4	*2.4*	58.9	*3.4*	61.9	*2.3*
GSMdec	Toggle	36.8	*1.1*	40.9	*0.2*	43.3	*3.6*	47.1	*0.7*	49.6	*0.0*	49.6	*0.0*
	IwFu	40.0	*0.0*	44.3	*2.7*	47.1	*2.3*	48.8	*0.2*	49.6	*0.0*	49.6	*0.0*
G728enc	Toggle	32.9	*1.0*	37.3	*1.9*	40.1	*0.1*	47.3	*0.0*	47.3	*0.0*	47.3	*0.0*
	IwFu	38.2	*3.3*	42.8	*3.0*	46.5	*0.2*	47.3	*0.0*	47.3	*0.0*	47.3	*0.0*
G728dec	Toggle	34.9	*2.2*	40.4	*0.2*	43.1	*5.7*	49.6	*0.4*	50.1	*0.0*	50.1	*0.0*
	IwFu	39.0	*0.3*	45.0	*2.2*	48.7	*0.4*	49.9	*0.3*	50.1	*0.0*	50.1	*0.0*
MPGenc	Toggle	33.9	*1.8*	36.2	*2.8*	41.0	*0.2*	51.4	*0.0*	51.4	*0.0*	51.4	*0.0*
	IwFu	40.0	*0.0*	42.6	*1.5*	48.3	*0.9*	51.4	*0.0*	51.4	*0.0*	51.4	*0.0*
MPGdec	Toggle	34.7	*2.2*	40.8	*0.4*	43.9	*2.8*	54.2	*0.0*	54.2	*0.0*	54.2	*0.0*
	IwFu	40.5	*0.4*	46.6	*2.5*	50.6	*3.1*	54.2	*0.0*	54.2	*0.0*	54.2	*0.0*
H263enc	Toggle	33.5	*2.0*	37.4	*2.7*	39.8	*0.1*	47.8	*1.2*	48.5	*0.0*	48.5	*0.0*
	IwFu	39.9	*0.1*	44.0	*1.6*	47.0	*1.1*	48.2	*1.9*	48.5	*0.0*	48.5	*0.0*
H263dec	Toggle	34.1	*2.1*	37.7	*2.8*	40.7	*0.2*	47.8	*0.3*	51.0	*0.0*	51.0	*0.0*
	IwFu	39.9	*0.1*	43.8	*1.6*	47.2	*1.1*	49.6	*0.3*	51.0	*0.0*	51.0	*0.0*
MP3dec	Toggle	33.2	*1.7*	36.7	*2.7*	40.1	*0.2*	47.4	*0.0*	47.4	*0.0*	47.4	*0.0*
	IwFu	38.5	*0.9*	42.2	*2.1*	46.1	*0.5*	47.4	*0.0*	47.4	*0.0*	47.4	*0.0*

Table 3. Steady state heat-sink temperatures for different applications for R-Toggle and R-IwFu at different thermal limits. The absolute difference between the steady-state temperature and the heat-sink temperature after the second iteration of our algorithms is also shown in italics.

5 Results

5.1 Iterative Approach to Determine Heat-Sink Temperature for Reactive DTM

Table 3 shows steady-state heat-sink temperatures for the two reactive schemes, R-Toggle and R-IwFu. For each application, the final temperature determined by our iterative approach is shown for different thermal limits. In addition, the absolute difference between the final temperature and the temperature after the second iteration of our algorithms is shown in italics. Figure 3 shows the intermediate heat-sink temperatures seen during the iterative process for reactive algorithms. Specifically, the intermediate temperatures seen when running R-Toggle for GSMenc is shown for the different thermal limits.

Dependence on application and thermal limit: For the same reactive algorithm, the heat-sink temperature is different for different applications and across different thermal limits. This is because (1) different applications have different power consumption, and (2) at various thermal limits, the magnitude of the reactive response required is different, leading to differences in processor power consumption. Also, it should be noted that for most applications, the heat-sink temperature for R-Toggle and R-IwFu remains constant above a certain thermal limit. This is because the maximum temperature of the application without any adaptation is lower than that thermal limit. For any thermal limit above this maximum temperature, no DTM response is required, resulting in a constant heat-sink temperature. Consequently, the final heat-sink remains constant after the second iteration of our algorithms.

Dependence on DTM algorithm: Table 3 shows that for a given application and thermal limit, the steady-state heat-sink temperature is different for R-IwFu and R-Toggle in many cases. This again is due to the differing effect of the two reactive schemes on processor power consumptions. As can be seen, the heat-sink temperatures for R-IwFu are higher than the temperatures for R-Toggle (in the cases where DTM responses are required). This is because R-IwFu has a lower impact on total processor power and instead targets localized hot-spots. In contrast, R-Toggle, due to its global nature, results in a larger drop in heat-sink temperatures.

Number of iterations: As Figure 3 illustrates, 5 to 6 iterations are required to determine the correct heat-sink temperature for GSMenc. Other applications show similar behavior. Table 3 quantifies this by showing the difference between the steady-state temperature and the second iteration heat-sink temperature (Δ column). Although these differences between steady-state and second iteration heat-sink temperatures may appear small, they translate to significant execution time differences. For example, GSMenc sees an average difference of 2.2 degrees for R-Toggle across all thermal limits but the average execution time difference is 26%. Similarly, GSMenc with R-IwFu sees an average execution time difference of 41% for an average temperature difference of 2.9 degrees. Skadron et al. noted a similar dependence of execution times on heat sink temperatures [11].

Implications: These results clearly show that using a constant heat-sink temperature for all simulations can potentially lead to incorrect results. Further, in addition to dif-

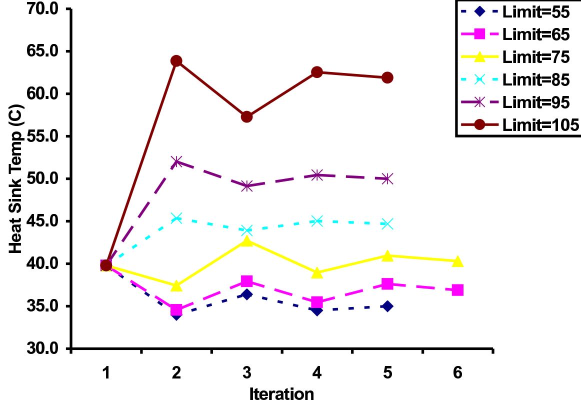

Figure 3. Iterative heat-sink temperatures converging to the correct temperature for GSMenc with R-Toggle for different thermal limits

App.	Thermal limit. (^oC)					
	55	**65**	**75**	**85**	**95**	**105**
GSMenc	1.47	1.34	1.23	1.12	1.07	1.00
GSMdec	1.38	1.26	1.05	1.00	1.00	1.00
G728enc	1.05	1.04	1.01	1.00	1.00	1.00
G728dec	1.15	1.06	1.01	1.00	1.00	1.00
MPGenc	1.43	1.15	1.01	1.00	1.00	1.00
MPGdec	1.34	1.12	1.03	1.00	1.00	1.00
H263enc	1.47	1.28	1.04	1.00	1.00	1.00
H263dec	1.52	1.31	1.08	1.00	1.00	1.00
MP3dec	1.35	1.22	1.07	1.00	1.00	1.00
Average	1.35	1.20	1.06	1.01	1.01	1.00

Table 4. Reactive schemes - Ratio of execution time of R-IwFu to R-Toggle (R-IwFu/R-Toggle) is reported.

ferentiating between different applications and thermal limits (as in [11]), heat-sink temperature differences across different adaptation algorithms should also be accounted for. Finally, as can be seen in the Δ column in Table 3, two iterations are insufficient to determine the correct heat-sink temperature in most cases.

5.2 Reactive Algorithms

Table 4 compares the performance of the two reactive schemes discussed in Section 3.1. For each application, the ratio of execution times of R-IwFu to R-Toggle (R-IwFu/R-Toggle) is shown for different thermal limits. The higher the ratio, the better R-Toggle performs compared to R-IwFu. As can be seen, R-Toggle performs better than or equal to R-IwFu for all the points in the table. At the most aggressive thermal limit of 55^oC, the performance of the processor running R-IwFu is 1.35 times slower than R-Toggle on average. The performance ratio between the two techniques decreases as we move to less stringent thermal limits since

App.	Thermal limit. (^oC)					
	55	**65**	**75**	**85**	**95**	**105**
GSMenc	1.28	1.31	1.15	1.12	1.10	1.07
GSMdec	1.49	1.24	1.02	1.00	1.00	1.00
G728enc	1.35	1.07	1.00	1.00	1.00	1.00
G728dec	1.23	1.12	1.02	1.00	1.00	1.00
MPGenc	1.25	1.09	1.01	1.00	1.00	1.00
MPGdec	1.28	1.14	1.01	1.00	1.00	1.00
H263enc	1.34	1.19	1.03	1.00	1.00	1.00
H263dec	1.47	1.27	1.08	1.00	1.00	1.00
MP3dec	1.29	1.21	1.07	1.00	1.00	1.00
Average	1.33	1.18	1.04	1.01	1.01	1.01

Table 5. Comparison of best reactive and best predictive scheme. Execution time ratio of R-Toggle and P-IwFuDVS (R-Toggle/P-IwFuDVS) is reported.

the reactive responses are invoked less frequently.

As mentioned previously, the register file is the hottest structure on chip for all our applications. R-IwFu directly targets register file power consumption by reducing the number of active ports in the register file. However, the impact of R-IwFu on the remaining structures on chip is limited and small changes in the instruction window size or number of functional units does not result in significant total processor power reduction. As a result, large changes in instruction window size and number of functional units are required for sustained periods of time in order to reduce the heat-sink temperature significantly. This subsequently results in a large drop in IPC when using R-IwFu.

In contrast, although toggling does not directly target the register file, the reduction in instructions in the pipeline reduces the power consumption of many structures on chip. This leads to a larger total processor power reduction, lowering the heat-sink temperature. Hence, R-Toggle can achieve the required temperature reduction with less performance loss than R-IwFu. As a result, in this scenario, a global adaptation technique performs better than a technique targeted at the hot-spot on chip.

5.3 Predictive Algorithms

Figure 4 shows the performance of all of our predictive and reactive schemes for different thermal limits. The horizontal axis in the graphs corresponds to the thermal limit, 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 constraints become stricter (from right to left on the horizontal axis), the slowdown increases (from bottom to top on the vertical axis). Due to the limited adaptation space explored, P-IwFu could not choose thermally safe architectural configurations for some thermal limits, resulting in incomplete P-IwFu lines for some applications.

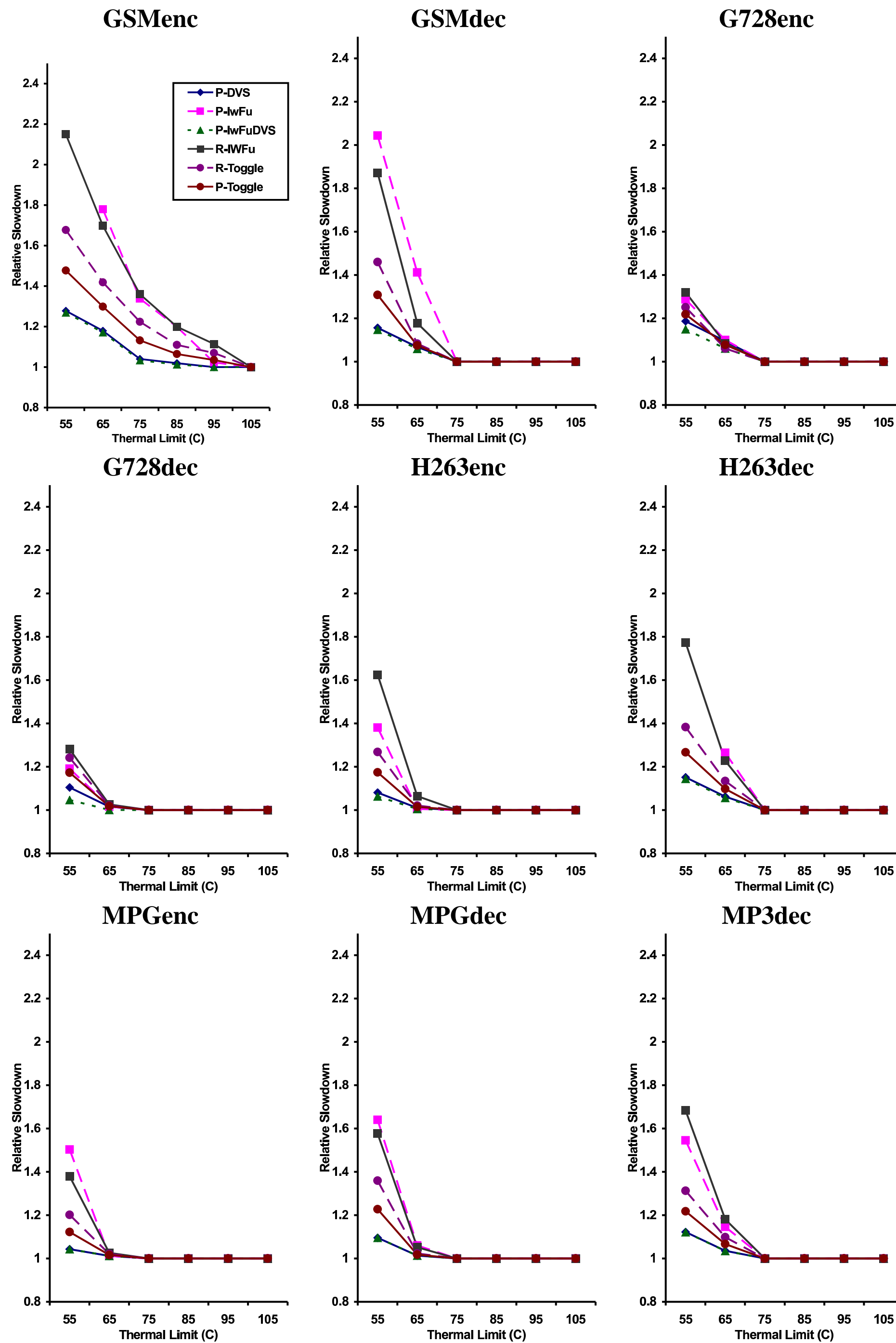

Figure 4. Predictive and reactive algorithms for all applications. The y-axis is the slowdown caused by the DTM algorithm over the base non-adaptive architecture. The x-axis shows the thermal limit in Kelvin. P-IwFu could not choose a thermally safe architecture for some thermal limits, resulting in some incomplete P-IwFu lines.

App.	Thermal limit. (^oC)					
	55	65	75	85	95	105
GSMenc	NA	NA	1.44	1.23	1.16	1.02
GSMdec	1.93	1.73	1.38	1.00	1.00	1.00
G728enc	1.23	1.12	1.06	1.00	1.00	1.00
G728dec	1.18	1.12	1.00	1.00	1.00	1.00
MPGenc	1.23	1.34	1.04	1.00	1.00	1.00
MPGdec	1.33	1.50	1.10	1.00	1.00	1.00
H263enc	NA	1.16	1.04	1.00	1.00	1.00
H263dec	NA	NA	1.25	1.00	1.00	1.00
MP3dec	1.52	1.31	1.08	1.00	1.00	1.00
Average	1.40	1.33	1.15	1.03	1.02	1.00

Table 6. Comparison of global vs. local techniques. Execution time ratio of P-IwFu and P-DVS (P-IwFu/P-DVS). A value of NA implies that P-IwFu could not choose a thermally safe configuration for that thermal limit.

Predictive vs. reactive schemes: As can be seen, P-Toggle, P-DVS and P-IwFuDVS perform significantly better than the predictive scheme, P-IwFu, and both reactive schemes, R-Toggle and R-IwFu, for all the applications. Among the predictive schemes, P-IwFuDVS always performs the best. In addition, the performance of P-IwFuDVS is very similar to P-DVS, implying that P-IwFuDVS prefers using DVS over architectural adaptation. Also, the superior performance of P-IwFuDVS and P-DVS over P-Toggle implies that DVS is a more performance effective technique than toggling.

Table 5 compares the best predictive scheme, P-IwFuDVS, with the best reactive scheme, R-Toggle. For each application, the ratio of execution times of R-Toggle to P-IwFuDVS (R-Toggle/P-IwFuDVS) is shown for different thermal limits. The higher the ratio, the better P-IwFuDVS performs compared to R-Toggle. As can be seen, the best predictive scheme always performs better than the best reactive scheme. At the strictest thermal limit, 55^oC, the processor performance with R-Toggle is 1.33 times slower than with P-IwFuDVS, on average. P-IwFuDVS performs better than the reactive schemes because of the use of DVS which produces significant processor power reduction, resulting in lower temperatures with a relatively low penalty on performance. DVS is not available to reactive algorithms due to the high time overhead involved when its invoked. This highlights an advantage of predictive algorithms over reactive algorithms. However, not all predictive algorithms are better than reactive algorithms. As can be seen in Figure 4, R-Toggle performs better than P-IwFu in most scenarios. Hence, for our applications, a global reactive technique (R-Toggle) performed better than a predictive technique targeted at hot-spots on chip (P-IwFu).

Global vs. local adaptation: Table 6 quantifies the difference between global and local adaptations. The ratio of execution times of P-IwFu to P-DVS (P-IwFu/P-DVS) is shown for different thermal limits. As in previous tables, the higher the ratio, the better P-DVS performs compared to P-IwFu. As can be seen, DVS performs significantly better in all our applications. At the strictest thermal limit, 55^oC, the processor running P-IwFu is 1.40 times slower than with P-DVS on average across our applications. Similar to the comparison of R-Toggle and R-IwFu, the global technique, P-DVS, performs better than the hot-spot based technique, P-IwFu, due to the total processor power reduction.

5.4 Comparison with Results from [12]

There are significant differences between the results in the previous section and the results in [12].

- **Comparison of predictive and reactive schemes:** All the predictive algorithms in [12] including P-IwFu performed better than the reactive algorithms. This was attributed to the benefit of high time overhead adaptations like register file resizing and DVS which result in large power reductions in the predictive schemes. In this paper, we see that P-Toggle, P-DVS and P-IwFuDVS still perform significantly better than the reactive algorithms. However, in this paper, we see that R-Toggle outperforms P-IwFu. Although P-IwFu uses register file resizing which is a high time overhead adaptation, register file resizing only reduces the power of a limited number of structures on chip (the register file, instruction window, and functional units). On the other hand, R-Toggle uses fetch toggling which reduces the power consumption of many structures on chip. This allows R-Toggle to have a larger impact on heat-sink temperature for the same loss in IPC, making it a more effective DTM technique.

- **Comparison of global and local adaptation techniques:** In [12], local techniques performed better than global techniques. Specifically, reactive instruction window and functional unit resizing incurred a smaller performance penalty than reactive toggling, and predictive instruction window, register file, and functional unit resizing performed better than predictive DVS. In this paper, our results show the opposite trend. R-Toggle with global toggling performed better than R-IwFu, and P-DVS and P-Toggle performed better than P-IwFu. Also, the bulk of the benefit in P-IwFuDVS stems from DVS.

  Hence, the benefit of lower total chip power on heat-sink temperature is more beneficial than local hot-spot temperature reduction with lower total chip power reduction. Among global adaptation techniques, DVS performs better than fetch toggling in our experiments.

Given that most of the simulation methodology in this paper and [12] is common, the differences in the results

arise mainly from the thermal models. There are some significant differences between the thermal models in the two papers. HotSpot, which is used in this paper, models the effect of average processor power consumption on heat-sink temperature. In addition, more thermal solution components like the heat-spreader are modeled. Also, the effect of tangential resistances between structures is modeled (i.e., with HotSpot, the impact of adjacent structures on a structure's temperature is modeled).

In order to isolate the role of the heat-sink, we repeated all of our experiments with a constant heat-sink temperature of 40^oC. Using this constant heat-sink temperature, we evaluated our two reactive DTM algorithms and our four predictive algorithms at thermal limits of 55^oC and 65^oC. Without accounting for the role of total processor power consumption on heat-sink temperature, we found that our qualitative results reverted to that seen in [12]. Specifically, all of our predictive schemes including P-IwFu performed better than our reactive schemes. In addition, local techniques targeted at hot-spots (R-IwFu, P-IwFu, and P-IwFuDVS) performed better than the corresponding global techniques (R-Toggle, P-Toggle, and P-DVS). Hence, we can conjecture that the bulk of the difference in the results arises from the modeling of the heat-sink, clearly highlighting the crucial role played by the heat-sink simulation in DTM and processor thermal evaluations.

In addition, even when using models like HotSpot [11] which account for the impact of processor power on heat-sink temperature, care should be taken to initialize the heat-sink temperature appropriately. As discussed in Section 1.1, when simulating non adaptive processors and predictive DTM algorithms, this issue can be handled by running every simulation twice. In contrast, reactive DTM algorithms have to be simulated over multiple iterations.

6. Conclusions

In this paper, we examined the key role played by the heat-sink in processor thermal simulations. Since the processor heat-sink temperature affects the temperature of every structure on chip, the thermal properties of the heat-sink plays a crucial role in processor thermal evaluations.

We highlight the importance of appropriate heat-sink simulation by re-examining the DTM results in [12]. We show that although the predictive DTM algorithms in [12] are still effective for long running multimedia applications, the conclusion that local response mechanisms targeted at hot-spots on chip are better than global mechanisms is erroneous. Instead, our global response mechanisms are more performance effective than localized mechanisms due to their impact on total processor power and heat-sink temperature.

These results highlight the importance of accurate initialization of heat-sink temperature in thermal simulations. This is particularly true when simulating reactive DTM algorithms due to the cyclical relationship between heat-sink temperature and the DTM response.

References

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

[2] D. Brooks et al. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proc. of the 27th Annual Intl. Symp. on Comp. Architecture*, 2000.

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

[4] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *Proc. of the Intl. Symposium on Low Power Electronics and Design*, 2003.

[5] M. Huang et al. A Framework for Dynamic Energy Efficiency and Temperature Management. In *Proc. of the 33rd Annual Intl. Symp. on Microarchitecture*, 2000.

[6] C. J. Hughes et al. Variability in the Execution of Multimedia Applications and Implications for Architecture. In *The International Symposium on Computer Architecture (ISCA)*, 2001.

[7] C. J. Hughes, V. S. Pai, P. Ranganathan, and S. V. Adve. RSIM: Simulating Shared-Memory Multiprocessors with ILP Processors. *IEEE Computer*, February 2002.

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

[9] M. D. Powell et al. Heat-and-Run: Leveraging SMT and CMP to Manage Power Density Through the Operating System. In *Proc. of the 10th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2004.

[10] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management. In *Proc. of the 8th Intl. Symp. on High-Performance Comp. Architecture*, 2002.

[11] K. Skadron et al. Temperature-Aware Microarchitecture. In *Proc. of the 30th Annual Intl. Symp. on Comp. Architecture*, 2003.

[12] J. Srinivasan and S. V. Adve. Predictive Dynamic Thermal Management for Multimedia Applications. In *Proc. of the 2003 Intl Conf. on Supercomputing*, 2003.