Characterizing Processor Thermal Behavior

Francisco J. Mesa-Martínez Ehsan K.Ardestani Jose Renau

Department of Computer Engineering
University of California Santa Cruz
http://masc.cse.ucsc.edu

Abstract

Temperature is a dominant factor in the performance, reliability, and leakage power consumption of modern processors. As a result, increasing numbers of researchers evaluate thermal characteristics in their proposals. In this paper, we measure a real processor focusing on its thermal characterization executing diverse workloads.

Our results show that in real designs, thermal transients operate at larger scales than their performance and power counterparts. Conventional thermal simulation methodologies based on profile-based simulation or statistical sampling, such as Simpoint, tend to explore very limited execution spans. Short simulation times can lead to reduced matchings between performance and thermal phases. To illustrate these issues we characterize and classify from a thermal standpoint SPEC00 and SPEC06 applications, which are traditionally used in the evaluation of architectural proposals. This paper concludes with a list of recommendations regarding thermal modeling considerations based on our experimental insights.

Categories and Subject Descriptors B.8.2 [*Performance and Reliability*]: Performance Analysis and Design Aids

General Terms Measurement, Experimentation, Performance, Reliability

Keywords Microarchitecture, Temperature, Thermal simulation

1. Introduction

This paper presents an experimental approach to measure and characterize the thermal behavior of real processors and their workloads. The aim of our work is to provide experimental data and insights to further validate and address the issues and challenges associated with the simulation and modeling of the thermal characteristics of processor designs.

Temperature has become a first order processor design constrain and has proven to be a key limiter in performance, throttling, clock skew, leakage power, reliability, variability, and cooling costs for modern processors. This has resulted in an increase of thermal publications. Figure 1 shows the number of papers published recently, in top computer architecture and VLSI conferences, addressing thermal issues. The top line in the chart tracks the fraction of those papers using thermal simulation, while the bottom trend line shows the fraction of those simulation-based papers which use HotSpot [26]. [1].

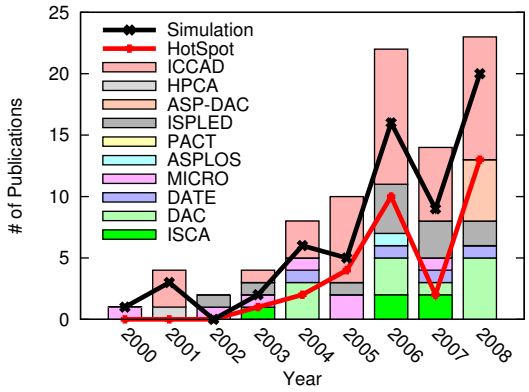

Figure 1. Growth of Temperature related publications over the years.

Most thermal research approaches require methods to calculate and measure temperature. This is traditionally addressed in three ways: using thermal and power simulation infrastructures such as Hotspot [26] and Wattch [2], hybrid approaches mixing thermal simulation and direct measurements of architectural state via performance counters, or via direct measurements using on-chip thermal sensor(s) such as thermal diodes.

Approaches using power and temperature simulation to estimate temperature offer the freedom to analyze different alternatives. However, the architectural simulations required to generate power consumption traces are slow, forcing architects to simplify their models. For example, most prior approaches either perform a single "long" simulation or use profile-based sampling simulations. For most modern processors, 1 billion instructions accounts for less than 1 second of execution. These constraints raise questions about the length of the simulation and the impact of profile-based sampling. Questions which are compounded by the effect in the thermal time response of different design parameters such as heatsink configurations and power densities.

In addition to simulation length and thermal time response (temporal resolution), spatial resolution is another challenging question regarding thermal simulation and measurement. This is mainly due to variable power densities, not just across major processor structures but among some of the substructures that comprise each processor block [1]. Diverse power densities inside major processor structures, together with their cycling and utilization behavior, may

ASPLOS'10, March 13–17, 2010, Pittsburgh, Pennsylvania, USA.

[1] We tracked papers in ISCA, MICRO, ASPLOS, HPCA, PACT, ISLPED, ICCAD, DAC, DATE, and ASP-DAC from 2000 to 2008.

define different thermal characteristics not only across major structures but inside them as well.

This paper attempts to provide educated answers to these questions using experimental data, which is used to provide a thorough thermal characterization of most of the applications in the SPEC CPU 2000 and 2006 benchmark suites. The experimental results show that the majority of SPEC applications have very predictable thermal behavior.

The main contributions of this paper are: the characterization of the SPEC benchmark suites for multiple thermal related metrics (1); show that conventional thermal simulation methodologies based on short execution times using profile-based sampling methods like Simpoint [24] can be ill-suited for thermal characterization (2); provide preliminary results showing that currently used spatial resolution at block granularity has issues with some thermal metrics (3); provide an analysis of the thermal time response of real processor systems and discuss its impact on the required simulation time (4); show that IO intensive applications have more thermal variability than SPEC (5); provide a set of recommendations for future research involving processor temperature (6).

The rest of the paper is organized as follows: Section 2 defines several temperature-dependent metrics, their related performance metrics, and the thermal implications of profile-based sampling. Section 3 addresses some issues related to the thermal measurement setup; Section 4 quickly summarizes the experimental setup and the parameters selected; Section 5 provides experimental results; Section 6 explains the related work; and finally Section 7 shows the conclusions and future work.

2. Temperature-Aware Metrics

In order to analyze applications from a thermal standpoint, a well defined set of temperature-aware metrics is required. Section 2.1 contains a summary of key thermal metrics. Section 2.2 details the related performance metrics. And lastly, Section 2.3 covers profile-based sampling and its potential impact on thermal evaluations.

2.1 Thermal Metrics

Temperature has a different impact on several key design factors such as timing integrity, reliability, leakage power, and cooling cost. For example, temperature has an exponential impact on leakage power but a linear relationship with various timing violations.

Furthermore, since power consumption is usually non-uniform across the die, this implies that temperature does not necessarily have a constant or uniform distribution across area or time. Thus, to properly characterize the thermal behavior of a processor, the pertinent metrics must capture both temporal and spatial temperature behavior of the design.

Thermal issues can be analyzed differently regarding their impact on timing, reliability, and leakage energy consumption. To address this, we define 3 different categories of thermal metrics: timing, reliability, and energy.

2.1.1 Timing

Thermal metrics regarding timing considerations include maximum temperature (*MaxT*) and maximum spatial difference in temperature which we refer to as thermal gradient (*GradT*). *MaxT* affects performance throttling. Furthermore, since temperature affects the operating frequency of circuits, it may limit the speed for the entire design. For the same reason, *GradT* affects performance because a high thermal gradient in the chip can induce skew violations and timing violations.

2.1.2 Reliability

We use RAMP [27] as a quantitative basis for reliability. This work describes 5 Mean Time To Failure (MTTF) wear out failure models: Electro Migration (EM), Stress Migration (SM), Time-Dependent Dielectric Breakdown (TDDB), Thermal Cycling (TC) and Negative Bias Temperature Instability (NBTI).

Since MTTF is not additive, the average Failures in Time (FIT) per block is estimated as the application executes. By default, a block corresponds to a processor architectural unit. Although, our work also considers finer grain spatial resolution. In both cases, the FIT is proportional to the area. At the end of the execution, we add the area-weighted FITs to report the overall FIT value for the entire processors. Like the RAMP authors, we assume that all the different failure mechanisms have the same contribution to the overall FIT value, which is adjusted to a preset value. In our case, we adjust the FIT value for all the SPEC applications to 10,000. This is approximately equivalent to a MTTF of 11 Years which is a short but reasonable lifetime for a 65nm chip.

Electro migration: occurs when atoms migrate from one end of the interconnect to the other, eventually leading to increased resistance and shorts. The model used in this work for EM is defined as follows:

$$MTTF_{EM} \propto (J)^{-n} \times e^{\frac{Ea_{EM}}{kT}}. \quad (1)$$

Stress migration: Materials differ in their thermal expansion rate, and this difference causes thermo mechanical stress, referred to as Stress Migration. We use the following SM model:

$$MTTF_{SM} \propto |T_0 - T|^{-n} \times e^{\frac{Ea_{SM}}{kT}}. \quad (2)$$

Time-dependent dielectric breakdown: It is the result of the gate dielectric gradual wear out, which leads to transistor failure. Ramp uses TDDB model

$$MTTF_{TDDB} \propto (\frac{1}{V})^{(a-bT)} \times e^{(\frac{X+\frac{Y}{T}+ZT}{kT})}. \quad (3)$$

Thermal cycling: Thermal Cycling is another reliability factor since the temporal thermal gradients e.g. power on and off and high frequency changes in power due to changes in workload behavior, affect the lifetime of the processor. There is no validated model for high frequency thermal cycles, but the effects of low frequency cycling can be modeled via:

$$MTTF_{TC} \propto (\frac{1}{T - T_{amb}})^{q}. \quad (4)$$

Negative bias temperature instability: NBTI leads to upward shifts in the transistors' threshold voltage that leads to timing violations. Ramp uses NBTI model

$$MTTF_{NBTI} \propto ((ln(\frac{M}{1+2e^{\frac{N}{kT}}}) - ln(\frac{M}{1+2e^{\frac{N}{kT}}} - H)) \times \frac{T}{e^{\frac{-I}{kT}}})^{\frac{1}{\beta}}. \quad (5)$$

2.1.3 Energy

Energy thermal considerations revolve mostly around leakage power (*Leak*) because of its temperature dependency. The BSIM3 model [14] provides a way to estimate leakage power [2].

$$Leak = P_0 * v_t^2 * e^{\frac{V_{GS} - V_{th} - V_{off}}{n * v_t}} (1 - e^{\frac{-V_{DS}}{v_t}}). \quad (6)$$

$$Leak \propto T^2 \times e^{\frac{\alpha}{T}} \times (1 - e^{\frac{\gamma}{T}})$$

Different processor blocks may have different transistor densities and/or leakage. Since this information is not available for real

[2] An alternative leakage power model can be found in [7]

processors, we assume that leakage is proportional to area. As in previous reliability metrics, we consider both block as well as finer grain spatial resolutions.

2.1.4 Thermal Time Constant

In order to capture the thermal "speed of change", we define a thermal time constant (τ) as the time it takes the chip to cool down to $(T_1 - T_0) * e^{-1}$ once the application stops executing. Similarly, τ can be also defined as the time it takes the chip to warm up from T_0 to $(T_1 - T_0) * (1 - e^{-1})$ when the application starts executing. The behavior is similar to the time constant for an RC circuit. Our experimental data shows thermal time constants for modern processor that are in the order of milliseconds. This gives a clear insight regarding the amount of time needed to simulate to better characterize the thermal behavior of a design. For example, based in our experimental figures, it is necessary to wait between 5 and 300 ms to have a thermal swing of around 63%.

2.2 Related Performance Metrics

Performance metrics can be used as a proxy for thermal behavior. We use a set of performance metrics as an approximation to the associated temperature profile of the application. These metrics include average IPC and maximum IPC. For the latter, the thermal time constant is used to filter the performance spikes that do not have a considerable effect on temperature. The average IPC is, however, calculated over the entire execution time.

2.3 Profile-based Sampling Impact

Profile-based sampling techniques that leverage Basic Block Vectors (BBV) like Simpoint [24] are commonly used to summarize application execution. Their goal is to find a subset of the application that matches the behavior for the entire application, so that a smaller window of execution can be used for simulation instead of executing the program to completion.

Many thermal simulation methods use Simpoint to accelerate execution, with the assumption that thermal phases match performance phases. This assumption, however, induces significant error.

To determine this error, we gather a thermal trace for the entire processor, measured by an IR camera during program execution. We also gather performance metrics, in real-time, from the program executing within the same processor being measured. Simpoint [24] is then used to determine the simulation points and their weights. Each simulation point needs to be adjusted by its own weight, we define this as True Simpoint (*True*). Which is very rare in thermal simulations, since most previous works simply concatenate simulation points without weight (*Typical*). Some authors go one step further, using only one simulation point. Which can be either the most representative, but more commonly, only the first simulation point (*First*) is selected to further reduce simulation time. Although a few papers do not perform warm-up, it is well understood that thermal warm-up is required. Unless otherwise stated, we perform warm-up before the first simulation point.

3. Thermal Measurement Setup

We leverage previous infrastructures [6, 15], developed to characterize chips thermally by directly measuring the temperature at the transistor layer using an infrared (IR) imaging system. Traditional chip cooling solutions use metal heatsinks and heat spreaders, which are opaque to the infrared spectrum. An IR-transparent cooling solution based on a continuous flow of IR-transparent oil is used to cool the target chip, while allowing us to observe its operation under nominal frequency and voltage specifications. A key difference with the [15] setup is the addition of a 3mm thick sapphire window [3] on top of the die to compensate for the replacement of the metal heatsink and heat spreader combo.

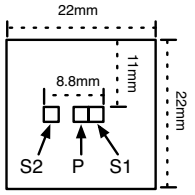

Figure 2. Testchip floorplan.

The objective of this section is to analytically and empirically show that an oil cooling solution could be used to characterize thermal phases. In the following subsection we address questions regarding oil cooling efficiency (thermal resistance), the oil flow direction, and the oil thermal transient response (thermal capacitance).

3.1 Oil Cooling Efficiency

Different cooling solutions can have different specific heat, thermal capacitance, and thermal resistance. For steady-state analysis, only the thermal resistance affects the cooling efficiency of the system. If we apply a constant power, the thermal capacitance does not have any effect. Heat transfer equations are similar to the Maxwell electrical equations. The electrical capacitors do not have any effect when a constant voltage is applied. Similarly, the thermal capacitance does not affect the temperature for a constant power.

Metal has a different vertical and lateral resistance than oil. In addition, there is a Thermal Interface Material (TIM) between the silicon and the heatsink. Non-uniform thermal resistance raises possible issues with the IR measurement setup [9]. We adapt the solution presented in [12]. First, apply an infrared transparent sapphire window (SW) on top of the die to compensate for the TIM and resistance changes. Second, adjust the oil flow to match the cooling performance of the equivalent metal heatsink solution.

A sapphire window increases the thermal capacitance and improves lateral heat spreading. The other missing factor is the TIM. Typical TIMs have thermal conductivity between 1 and 4 $\frac{W}{mK}$. This thermal resistance is placed between the silicon substrate and the metal heatsink. For the oil solution, we use oil as an IR transparent TIM. Liquids are effective TIMs but not commercially used.

Once we have a sapphire window, we control the oil volume flow as previously used by [15]. Nevertheless, there are physical limits or lower bounds beyond which the oil flow stops behaving like a laminar flow. To safely avoid difficult to model oil flow artifacts, we fix the oil flow speed to $16\frac{m}{s}$, and restrict the minimum oil thickness to 1mm. For systems operating under 20W additional adjustments lowering flow thickness and rate may be required.

To evaluate the effectiveness of the previous corrections, we use a testchip with a $484mm^2$ die area implemented on a BGA GL771 package as shown in Figure 2. The chip is divided into a grid of blocks arranged as non-overlapping tiles of similar area. The power consumption for each block in the chip can be independently controlled. Each block also has a unique embedded thermal diode, which can sense sub-millisecond thermal responses with sub-centigrade error in temperature. We focus our attention on 3 specific blocks in the testchip: A single block in the center (P) which is powered arbitrarily, and two adjacent blocks (S1, S2) which remain inert.

Figure 3 shows the temperature for blocks P, S1, and S2 in the testchip when different steady-state power consumptions are applied to block P. The oil flows from top to bottom. When Block

[3] A silicon window has a lower thermal capacitance and resistance.

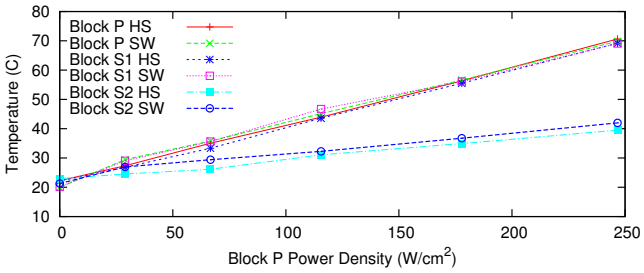

Figure 3. Temperatures for blocks P, S1, and S2 with different constant power in block P. HS and SW stand for AMD Mobile heatsink and sapphire window respectively.

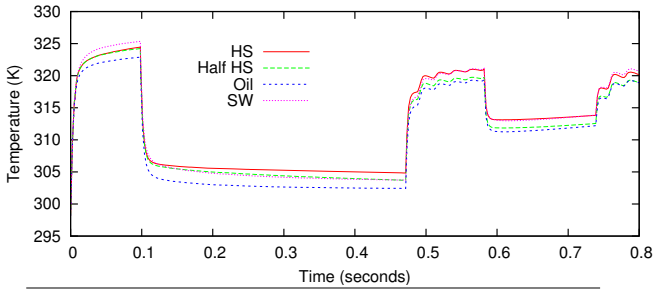

Figure 4. Athlon-like simulated transient thermal response for different cooling solutions.

P gets powered from 0 to 250$\frac{W}{cm^2}$, we measure a consistent linear increase in temperature for the heatsink (HS) and the sapphire window (SW). This implies that the overall vertical thermal resistance of the oil and metal heatsinks are very similar.

To validate the lateral thermal resistance, we measure blocks S1 and S2. S1 is adjacent to block P, and therefore its temperature is very close to P when using both the metal and the sapphire solution. However, the temperature difference between block P and block S2, placed 5mm away from each other, increases as the power density increases. This is because the lateral thermal resistance creates thermal gradients on the die. We observe that both the traditional heatsink and sapphire have a consistent slope.

This simple experiment also validates the behavior of the sapphire window as an alternative cooling solution to a metal heatsink with similar vertical and lateral thermal resistances.

3.2 Oil Flow Direction

To compensate for the different cooling efficiency across the die due to the direction of the oil flow, the IR setup performs an additional image correction over each captured frame. In the worst case, we observe a maximum temperature gradient of 4°C between opposite sides of the testchip. This corresponds to approximately 0.2°C correction for each *mm* that the oil flows over a hot block.

Our experimental evaluation isolated fewer gradients than reported by [9]. The reason is that the sapphire window attenuates the impact of the oil flow because it dissipates the temperature from a more localized area at the silicon substrate to a bigger area at the top of the sapphire window. To further reduce the gradients due to oil flow, it is possible to add a diamond heatsink on top of the sapphire window which increases the lateral resistance. Alternatively, we explore a software correction mechanism.

If all the blocks are uniformly heated, applying the $\frac{0.2°C}{mm}$ correction is a simple and effective alternative. However, real chips do not display such uniform temperature across their dies. Ideally, a model describing the fluid dynamics of the oil should be used to perform the oil flow correction. However, this solution is too compute-intensive especially considering the 4°C worst case. Instead, we have a quick approximation estimating the oil flow correction. For every *mm* that the oil flows over a block, we adjust the correction by $0.2 * \frac{BlockTemp - 45°C}{10°C}$. We never let the correction be negative. This is a simple algorithm with linear cost that provides a fast and effective solution.

Block	Top-Bottom	Left-Right	Right-Left	Bottom-Top
P	64.9 (65.3)	64.9 (65.3)	64.9 (65.3)	64.9 (65.3)
S1	63.9 (63.9)	64.7 (65.4)	63.6 (63.6)	63.5 (63.5)
S2	48.2 (48.2)	48.1 (48.1)	47.8 (48.6)	48.2 (48.2)

Table 1. Oil flow direction impact. Uncorrected value in parentheses.

To evaluate the accuracy of the correction and the impact of the oil flow, we repeat the same experiment as in Figure 3. This time we apply the oil from four possible directions when block P is powered with 7.8W. The values in parentheses are the uncorrected values obtained when the oil flow correction algorithm is not applied. The only blocks affected by the oil flow direction are S1 and S2 when we have a horizontal flow. Without correction, the maximum error is 1.8°C (S1 with Left-Right flow). After the correction, the error is reduced to 0.9°C.

3.3 Oil Thermal Transient Response

To validate the thermal response for the oil flow, we start by performing thermal simulations using an Athlon-like chip configuration with metal heatsink and oil. Then, we continue by performing experiments with the testchip.

Figure 4 shows the simulated thermal transient response for four different cooling solutions using an Athlon-like chip with package. “HS" is a traditional heatsink solution; “Half HS" is equivalent to “HS" but halving the metal spreader thickness; “Oil” shows the theoretical thermal response with an oil cooling solution; “SW” shows the thermal response when a sapphire window is placed between the silicon substrate and the oil flow.

Reducing the heatsink thickness (“HS” vs “Half HS”) affects the metal spreader thermal capacitance and lateral thermal resistance. As a result, a slightly different thermal response is observed but both cooling solutions still show the same thermal phases. While the additional thermal capacitance attenuates the thermal response, the decreased lateral resistance reduces the cooling efficiency.

As reported by [9], “Oil” flow has a lower thermal capacitance and it has a faster response showing clearer thermal phases. The previous section has shown that a sapphire window (SW) reduces the thermal resistance difference between oil and the heatsink. Sapphire also increases the thermal capacitance because it has double the specific heat of copper (0.75 $\frac{J}{g*K}$ vs 0.385 $\frac{J}{g*K}$) but approximately half the density. As a result, copper and sapphire have an equivalent thermal capacitance. In conclusion, “SW” has a more attenuated thermal response than “Oil”.

In our experimental validation, we use a testchip which provides μs sampling capabilities. Figure 5 shows block P temperature when a periodic power pulse is applied to it and the rest of the chip is idle. The same power pulse is applied for a heat sink “HS” and a sapphire window “SW”.

The 25ms power pulse is applied every 100ms (10Hz). We clearly observe that the thermal transients of the heatsink and the sapphire are very close. [9] pointed that an oil cooling solution with this power pulse would have a significant error for fast transients. The measured results show that a sapphire window solves the problem. We also perform 250 *ms* power pulses to validate the equivalence between the heatsink and the sapphire window for

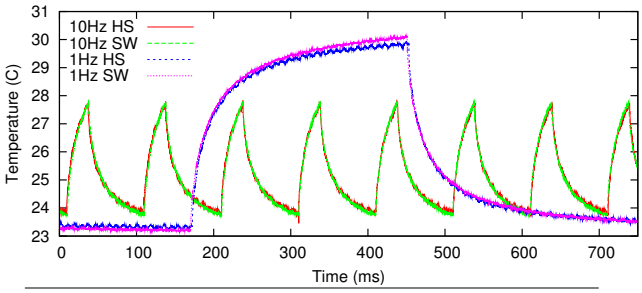

Figure 5. Thermal transient response for a testchip when an 25 *ms* power pulse is applied periodically every 100 *ms* (10Hz). HS and SW stand for Heatsink and Sapphire Window respectively.

slower transients. Again, the proposed cooling solution closely matches the metal heatsink solution.

Combining the fast and slow transient response accuracy with the cooling efficiency validation for the vertical and lateral thermal resistances, we conclude that the oil cooling solution with a sapphire window is an appropriate vehicle to capture existing thermal phases.

4. Evaluation Setup Parameters

4.1 Performance Parameters

To gather IPC traces we use pfmon [11], a performance monitoring tool. One of the machine specific registers (MSR) available in the CPU is used to count the number of retired instructions (RI). We program in a threshold that when reached, resets the RI counter to zero, resumes counting, and generates an interrupt. A time stamp is saved in the interrupt service routine for this event so that we can determine the time it takes to reach the threshold number of instructions to retire. To extract the basic block vectors (BBV), we use $Valgrind-3.3.0$ [19] and $exp-bbv$ plugin [24]. Then, we use Simpoint 3.2 [24] to generate the simulation points.

4.2 Thermal Parameters

The thermal measurement setup uses an infrared camera to directly measure the transistor temperature. The infrared camera is a FLIR SC-8000. This camera provides real-time thermal imaging a resolution of 1024x1024 pixels, with sampling rates of over 100Hz.

For each category of thermal metrics (performance, reliability, energy), we report the metrics based on constants extracted from 65nm technology files. Table 2 summarizes all the metrics and their parameters.

Category	Metric	Parameters
Timing	*MaxT*	
	GradT	
Reliability	*EM*	$a_{EM} = 0.9, k = 8.617343 \times 10^{-5}$
	SM	$n = 2.5, a_{SM} = 0.9$
	TDDB	$V = 1.1, a = 78, b = -0.081,$
		$X = 0.759, Y = -66.8, Z = -8.37$
	TC	$T_{amb} = 293, q = 2.35$
	NBTI	$M = 1.6328, N = 0.07377$
		$I = -0.06852, \beta = 0.3, H = 0.01$
Energy	*Leak*	$\alpha = -1175, \gamma = -0.00005$

Table 2. Thermal metrics constants.

As stated previously, this work focuses on the measurement of thermal characteristics from a real processor operating under nominal frequency and voltage. For this purpose, we measure an AMD K8-based processor, running at 1.7GHz. It offers an out-of-order single core, manufactured using a 130nm process, with a peak power consumption of 70Watts.

4.3 Cooling solution

As explained in Section 3, we use an IR transparent heatsink with a sapphire window that matches the cooling characteristics of the metal cooling solution for a Mobile Athlon processor.

In order to satisfy those requirements, we use an oil-sapphire combination as the cooling solution for this experiment. A 3mm thick sapphire window with a 50mm diameter is placed on top of the chip, which acts like a traditional heat spreader while allowing IR energy to be measured. This window is then cooled by a flow of mineral oil, which is also transparent to IR and removes the heat. The mineral oil is temperature-controlled by a heat exchange maintaining the oil between 15°C to 20°C. This allows the jet of oil to remove heat from the sapphire window optimally, while maintaining a laminar regime of flow with a speed of around 16 $\frac{m}{s}$. We use purified mineral oil with a specific heat of 1.63 $\frac{J}{gK}$.

4.4 Applications

We evaluate almost all of the applications of SPEC00 and SPEC06 suites (24 from SPEC00 and 22 from SPEC06). Reference input sets are used for All the SPEC benchmarks.

Applications with a mixture of computation and IO tend to display more varied thermal behavior as observed with our IR setup. Since all the SPEC applications are designed to be CPU bound, we complement them by also evaluating 5 applications involving I/O: System Boot, Linux make, pdflatex, emacs, and BDB. System Boot includes the time from when the machine is powered up until the Linux boot is finished; Linux make compiles and links the Linux kernel and its modules; pdflatex compiles this paper with pdflatex; emacs performs verilog-mode macro, which creates a module skeleton, performs auto-completion, and syntax check. Once the verilog macro is finished, we conduct a conversation with the ELIZA mode for natural language processing. The BDB test involves a database with 1000K random fixed-size records each containing 32-bit keys and 256-bit data fields. Pages are 32KB, bulk transfers buffer is set to 4MB, the log buffer is also 4MB, and the buffer cache is 8MB. We perform 5K random queries and back-to-back update pairs.

For all the applications in SPEC, the execution time is limited to 90 seconds which is long enough to let capturing thermal transitions and far longer than most architectural simulations.

5. Evaluation

The evaluation is divided in 6 sections. Section 5.1 starts by characterizing SPEC applications and showing all the thermal metrics. Section 5.2 continues analyzing the impact of profile-based sampling with thermal simulations. Section 5.3 shows how to estimate the minimum simulation time. After showing the impact of other issues in Section 5.4 and discussing the impact of IO in Section 5.5, Section 5.6 concludes with enumerating a list of thermal modeling recommendations.

5.1 SPEC Characterization

Table 3 presents the performance, reliability, and energy thermal metrics for our target processor executing SPEC00 and SPEC06.

MaxIPC as well as metrics capturing the maximum temperature and thermal gradient (*MaxT*, *GradT* respectively) only reveal a small portion of thermally-related issues that may be of interest. To complement these metrics, we provide temperature-dependent reliability (EM, SM, TDDB, TC, NBTI) and power (Leak) metrics. The reliability metrics in Table 3 are normalized to a Meat Time

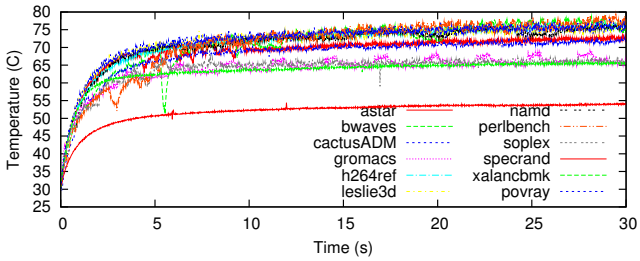

Figure 6. Thermally predictable SPEC06 applications.

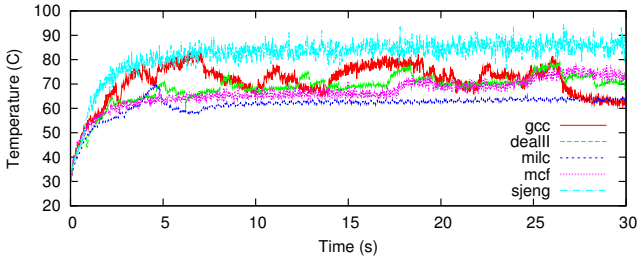

Figure 7. Thermally variable SPEC06 applications.

Between Failures (MTBF) of 57.08 years, in the same fashion leakage numbers are also normalized to 1.

From our experimental data, we observe some interesting thermal aspects triggered by SPEC execution. First, there is a lack of correlation between average IPC and temperature. The correlation between IPC and *MaxT* is 0.36 and between IPC and *GradT* is 0.27. For example, the CFP00 suite has an average IPC of 0.81, which generates an average *MaxT* 76°C, however CINT00 displays a lower average IPC of 0.69 generating a higher average *MaxT* of 80°C. A similar lack of correlation can be observed on a per application basis. For example, sixtrack has the highest average IPCs of 1.59 (with the second highest MaxIPC of 1.6) reaching *MaxT* and *GradT* of 78°C and 41°C respectively. Whereas mcf, with an IPC 77% lower, reaches a 4% higher maximum temperature. The reason for this lack of correlation is that *MaxT* and *GradT* report the maximum temperature or the maximum temperature difference. These values are not very correlated with average IPC. MaxIPC displays better correlation with temperature but the correlation is still low because short IPC spikes are not long enough to increase *MaxT*.

Better correlation can be found between temperature and the EM reliability metric. For example, specrand has both the lowest (*MaxT*,*GradT*) tuples with (55°C, 17°C) respectively, and the highest EM with 232.41 years. At the same time, sjeng has the highest (*MaxT*,*GradT*) tuple with (103°C, 77°C) and the lowest EM of 22.35 years. However, SM does not follow a similar trend since it is more dependent on spatial concerns, such as the temperature distribution across the die.

By analyzing the correlations in Table 3, we can observe that all three categories of thermal metrics have a low correlation. As a result, one metric cannot be approximated from another.

From the transient temperature data captured with our experimental setup, we divide the thermal behavior for the SPEC suite in two categories: thermally predictable and thermally variable. For example, the thermally predictable category (Figure 6) for the SPEC06 benchmarks have a predictable plateau after the initial warm up. In contrast, the thermally variable category (Figure 7) have over 5°C oscillations once the warm up is over. The majority of SPEC06 is fairly predictable, only a few benchmarks have significant performance oscillations. SPEC00 with a shorter execution time has even shorter oscillations, and only gcc, swim, mesa, lucas, gap, galgel, facerec, and mcf [4] could be classified as thermally variable. Of note gcc and facerec show oscillations over 10°C. The temperature of each trace has a constant increase as it goes through time. Figure 6 and 7 show only 30 seconds of the execution, while Table 3 reports the results for 90-second long execution. Thus the maximum temperature reported for each application in Table 3 might be higher than what shown in these figures.

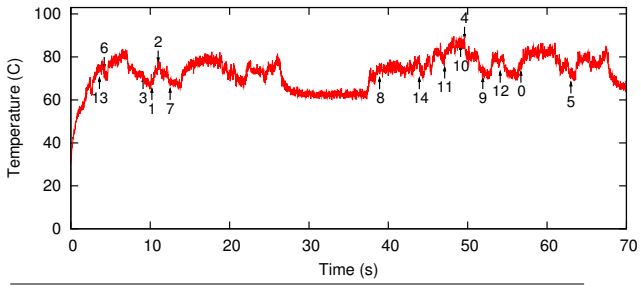

Figure 8. SPEC06 gcc MaxT together with the simulation points.

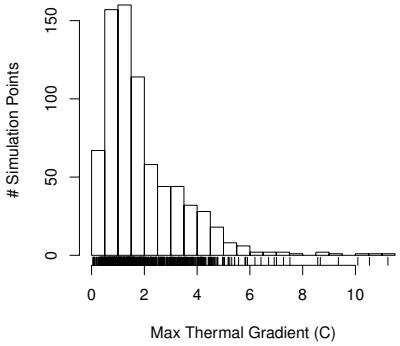

Figure 9. Histogram with the maximum thermal gradient observed in each simulation point for all the SPEC applications.

5.2 Profile-based Simulation Impact

Figure 8 shows the overlap between simulation points and the measured performance trace for the SPEC06 gcc benchmark. Each simulation point lasts only 100M instructions which corresponds to approximately 100 *ms*.

We observe two key problems with simulation points. First, the ending temperature of each simulation point can be very different from the starting temperature of the following simulation point. For example, while simulation point 1 finishes in the high 60s°C, the following simulation point 2 starts in the high 70s°C. This leads to inaccurate simulations. Second, simulation points are too short to capture significant transients. Figure 9 shows a histogram for the maximum thermal gradient observed in each simulation point for all the SPEC applications. The majority of the simulation points have a thermal gradient under 2°C. Some simulation points have over 4°C because they are located during the warm up phase where the thermal gradient is higher.

[4] mcf only has two spikes of thermal variability after 60 seconds of execution

	Apps	IPC	MaxIPC	*MaxT*	*GradT*	EM	SM	TDDB	TC	NBTI	Leak
CINT00	crafty	1.04	1.14	78	41	60.24	162.63	67.14	141.52	63.11	1.0
	vortex	0.97	1.31	84	49	47.09	100.86	58.65	83.14	57.1	1.03
	gcc	0.96	1.61	83	53	88.23	315.22	72.66	377.17	63.94	0.94
	gap	0.87	1.13	86	50	45.99	76.43	57.31	64.42	57.14	1.03
	eon	0.86	0.92	79	44	64.59	181.13	67.5	161.68	62.85	0.99
	bzip2	0.85	1.44	86	50	46.24	64.29	53.42	53.85	54.03	1.03
	parser	0.66	0.96	74	39	87.04	219.9	70.86	212.15	64.94	0.95
	twolf	0.57	1.0	79	46	66.11	117.46	61.36	97.97	59.07	0.99
	vpr	0.44	0.66	81	44	61.49	103.5	58.91	85.69	57.19	1.0
	mcf	0.19	1.09	72	38	137.03	151.83	63.37	134.43	59.41	0.9
	gzip	0.14	0.27	78	42	75.1	220.07	67.5	215.43	61.29	0.96
	Average	0.69	1.05	80	45	64.14	126.08	62.98	109.24	59.82	0.98
CFP00	sixtrack	1.59	1.6	78	41	58.33	65.89	57.0	55.33	57.95	1.02
	wupwise	1.44	1.51	75	32	66.35	40.34	51.12	37.52	53.94	1.0
	applu	0.85	1.27	80	40	52.71	57.25	55.53	49.05	57.14	1.03
	galgel	0.81	1.8	78	41	69.87	49.29	51.38	43.23	53.22	0.99
	mgrid	0.8	1.13	68	28	100.81	72.48	58.48	60.28	59.0	0.95
	lucas	0.74	1.36	88	55	52.18	43.92	50.36	42.51	51.96	1.02
	apsi	0.73	1.18	76	37	70.82	94.4	62.03	78.35	60.95	0.99
	facerec	0.72	1.28	83	51	91.56	193.98	68.35	182.04	62.9	0.95
	mesa	0.7	1.36	69	37	121.37	162.38	67.3	143.41	63.06	0.92
	swim	0.67	0.84	83	42	43.28	47.68	52.76	42.42	54.96	1.05
	equake	0.64	1.32	67	32	114.8	59.36	53.47	50.19	54.61	0.93
	ammp	0.6	0.82	75	38	74.84	47.66	51.53	41.83	53.67	0.99
	art	0.25	0.31	73	44	104.55	645.39	90.75	1331.91	76.1	0.93
	Average	0.81	1.21	76	40	71.19	66.15	57.72	58.62	57.85	0.98
CINT06	perlbench	0.95	1.26	84	50	51.29	433.68	83.43	631.42	72.36	1.01
	h264ref	0.94	1.16	84	39	44.23	23.26	45.87	24.88	51.03	1.05
	hmmer	0.87	0.93	100	77	22.35	109.62	59.75	94.4	57.24	1.12
	libquantum	0.87	1.55	97	77	40.56	119.9	62.05	106.69	58.65	1.04
	gcc	0.84	1.49	90	65	56.67	109.95	61.61	92.6	59.48	1.0
	sjeng	0.82	0.92	100	75	27.42	147.06	62.78	132.23	58.54	1.09
	bzip2	0.78	1.28	87	53	44.5	128.71	65.08	111.47	61.81	1.03
	gobmk	0.76	0.95	95	63	29.02	72.1	55.66	61.6	55.49	1.09
	xalancbmk	0.57	0.94	74	34	84.85	61.08	55.05	51.33	56.27	0.97
	mcf	0.36	1.1	81	45	67.46	67.1	55.1	56.68	55.54	0.99
	astar	0.35	0.93	77	34	58.73	30.58	48.1	29.51	52.38	1.02
	specrand	0.09	0.12	55	17	232.41	101.1	57.31	81.73	55.91	0.85
	Average	0.68	1.05	85	53	44.36	68.9	58.06	63.64	57.47	1.02
CFP06	namd	1.1	1.27	82	40	45.04	58.4	53.63	49.63	54.82	1.05
	gamess	0.97	1.37	84	41	39.66	30.53	47.09	29.63	51.04	1.06
	dealII	0.91	1.62	85	48	62.39	189.58	66.47	176.89	61.39	0.99
	povray	0.88	0.95	81	48	54.75	368.56	75.76	523.19	66.3	1.01
	leslie3d	0.78	0.93	81	38	45.53	20.16	43.33	22.04	48.89	1.05
	cactusADM	0.63	1.08	77	35	58.92	34.33	47.69	32.03	51.15	1.02
	milc	0.62	1.6	75	35	95.85	44.56	50.61	39.15	53.1	0.95
	gromacs	0.61	1.2	72	32	84.96	46.53	51.55	40.38	54.03	0.97
	bwaves	0.45	0.95	83	36	39.86	7.51	36.74	12.64	45.44	1.07
	soplex	0.42	0.92	71	34	90.34	87.39	55.66	71.12	54.87	0.96
	Average	0.74	1.19	79	39	55.87	30.75	50.89	35.52	53.54	1.01
	SPEC Average	0.73	1.13	80	44	57.08	57.08	57.08	57.08	57.08	1.0
Other	BDB	-	-	47	13	952.05	500.64	74.5	1333.86	63.24	0.7
	Emacs	-	-	45	13	1677.6	1534.95	89.97	2081.04	69.78	0.65
	Power Off	-	-	44	15	1813.04	1428.76	88.01	1004.08	68.25	0.64
	System Boot	-	-	67	25	300.54	48.59	50.86	44.32	52.06	0.81
	pdflatex	-	-	43	15	1308.97	1048.95	84.1	9532.39	67.27	0.67
	Linux Make	-	-	93	61	48.98	7.5	41.91	16.28	48.98	0.98
	Average	-	-	61	28	147.21	32.94	61.65	53.26	58.86	0.78

Table 3. Performance and Thermal metrics for SPEC00, SPEC06, and IO Applications.

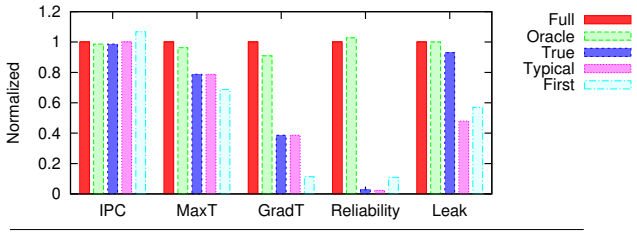

Figure 10. Different Simpoint approaches average results normalized against the full execution.

Figure 10 shows the impact of the simulation points on IPC and different thermal metrics. There are five comparison points. *Full* corresponds to the complete execution of the program. *Oracle* is a weighed simulation point with an oracle setting its correct initial temperature. *True* uses a weighted average and the ending temperature from the last simulation point as the beginning temperature for the current point. *Typical* is the same as *True* sans a weighted average. This is the most typical method to use simulation points with temperature simulations. Some papers use only the first simulation point to perform their simulation (*First*).

As expected, IPC is very close to all the methods except *First*. For these metrics, *First* is not representative. *Typical* and *True* have similar results with the exception of leakage power where *True* has better results. In both cases, there is a 20% error for *MaxT*. The most problematic metric is reliability where the results are as bad as *First*. For the thermally variable subset of applications, the difference is even worse than the reported average normalized values (for example 1.18 vs. 1.03 for *Oracle*).

Oracle shows that simulation has potential to yield correct results if the temperate for each simulation point is set correctly. However, temperature has a significant state, making it necessary to thermally simulate several seconds before the simulation point starts to obtain the correct initial temperature.

Using simulation points with thermal simulations introduces potential issues when evaluating reliability metrics. With a worst case occurring when only one simulation point is used, in that case all the metrics but IPC have significant error. Simulation points have potential to yield correct results if the initial temperature is appropriately set for each simulation point. [3] proposes reusing the power consumption calculated in similar phases to approximate the starting temperature of the simulation point. We cannot verify the accuracy of this system with our setup.

5.3 Simulation Time

A common question by architects is "how long should I simulate?" Section 5.2 shows that profile-based simulation has several issues regarding thermal modeling. The answer is not simple because it depends on package, application, and metrics being tracked.

5.3.1 Package Impact

To understand the "the minimum time required to simulate" due to package constraints, we use the thermal time constant as defined in Section 2.1.4. If we ignore the material property changes due to temperature, the package thermal constant is independent of the power trace.

The package acts as a low-pass filter, attenuating those power cycles with frequencies higher than the cutoff frequency for the filter. Since the thermal constant operates in an RC fashion, we define the cutoff frequency as $\frac{1}{2\pi\tau}$. Therefore, we consider $2\pi\tau$ as the minimum simulation time using the package as a constraint. Shorter simulation intervals are not long enough to pass the filter without the temperature being attenuated. A 50 *ms* thermal time constant (τ) implies that we need simulation periods of over 300 *ms*. In Figure 5 we observe a thermal time constant (τ) of 54 *ms* for our sapphire cooling solution.

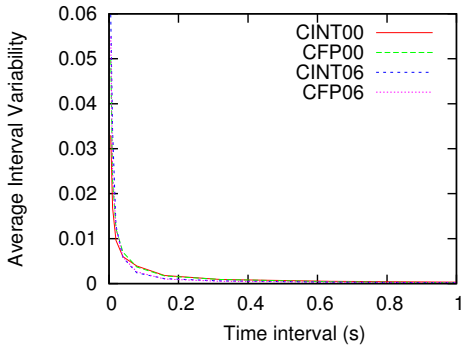

Figure 11. SPEC suites Average Interval Variability.

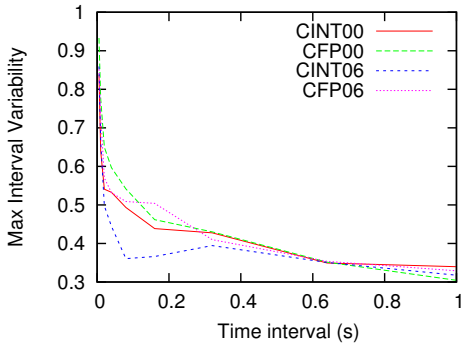

Figure 12. SPEC suites Max Interval Variability.

5.3.2 Application Impact

The package for the processor provides a minimum simulation time required to obtain a meaningful thermal simulation ($2\pi\tau$ or 300 *ms* for the package analyzed in this paper). Nevertheless, applications may have longer phases leading to more moderate changes in power density. This section provides more insight into the minimum time for thermal simulations from the standpoint of the applications being analyzed.

Rangan et al [23] define *Interval Variability* as the difference in IPC of one interval against IPC of the previous interval. The authors observe that 300 cycles have an order of magnitude more interval variability than 10,000 cycles intervals. Both ranges are still much faster than the 300 *ms* cutoff frequency for the package in this study. Figure 11 shows the average *Interval Variability* between 5 *ms* and 0.5 *secs*. It shows that most of the variability happens in time intervals under 150 *ms*. For a thermal frequency cutoff of 300 *ms*, all the changes happening faster are attenuated. This is one of the reasons why so many SPEC applications fall in the thermally predictable category.

Figure 12 shows the maximum *Interval Variability*. The SPEC applications have over 0.3 (or 30%) activity change for all time intervals. Therefore, thermal metrics may be affected by the potential changes in power activity. As a result, we need to look at the thermal metrics because the SPEC applications do not show clear points to constrain simulation time.

5.3.3 Thermal Metric Impact

In addition to the package and application, the metric being observed also has an impact in the simulation time requirements.

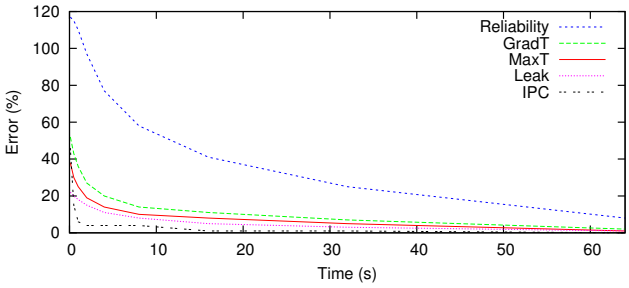

Figure 13. Impact of simulation time on accuracy.

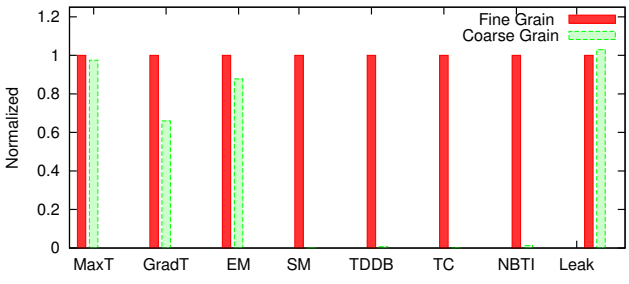

Figure 14. Measurement granularity error.

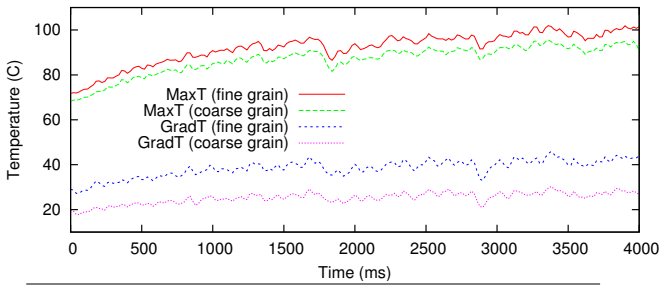

Figure 15. Measurement grain granularity impact on MaxT and GradT.

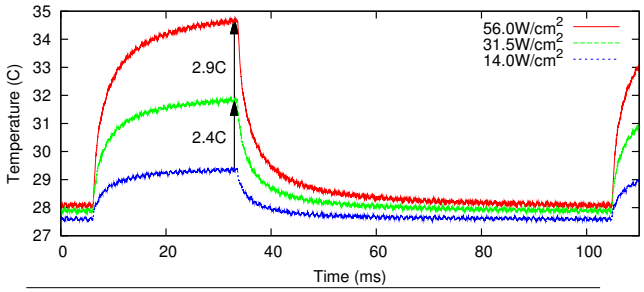

Figure 16. Thermal time response when block P is powered at different power densities.

Figure 13 shows the error or inaccuracy suffered when only a subset of the application is modeled for different metrics. IPC is the metric that requires the least simulation time, a few seconds are enough to yield relatively accurate results. As shown in Section 5.2, simulation points can further improve the accuracy yielding over an order of magnitude accuracy improvement given the same simulation time.

Performance and energy thermal metrics (*MaxT*, *GradT*, *Leak*) require longer simulation times, requiring over 10 seconds to achieve an error of less than 20%. Reliability thermal metrics are the most difficult to capture. They require over 40 seconds to have less than 20% error. The previous plot selects a subsection of the application after warm-up. If warm-up the simulation starts from the beginning of the application the same type of results still hold.

Since thermal metrics depend on the application behavior, it is more desirable to consider the number of instructions than the overall time of execution. For a 1.7GHz processor with an average IPC of 0.73, we need approximately 20 billion instructions for performance and power thermal metrics, and close to 100 billion instructions for reliability thermal metrics. For IPC estimation, we just need around 2 billion instructions after initialization which is a fairly common simulation length.

5.4 Other Thermal Issues

5.4.1 Spatial Resolution Impact

Up to this point, our evaluation has used a spatial resolution of 50x50μm. We now evaluate the impact of averaging the temperature inside each of the floorplan blocks.

Figure 14 compares the different thermal metrics between fine grain and coarse grain. *MaxT* and *Leak* do not have significant differences, but most reliability metrics are completely displaced. The problem resides with the exponential dependence of reliability metrics with temperature, which makes them very sensitive to temperature distribution changes in a given area.

Figure 15 shows the difference in *MaxT* and *GradT* when using fine grain vs coarse grain thermal measurements. Interestingly there seems to be a constant offset between them. This opens to opportunity to further optimizations.

We observe that the spatial resolution has a big impact on many metrics because they are not linear with temperature. Nevertheless, more work is needed to quantify what is the correct granularity and how to manage the reliability metrics because they are the most sensitive to spatial resolution.

5.4.2 Heatsink and Power Density Impact

Finally we consider the impact of the cooling solution and power density on thermal modeling and simulation. While power density affects the maximum temperature, the heatsink affects the max temperature and the thermal time constant.

Power Density Figure 16 shows the impact of changing the power density. Again, we power cycle block P with a 1Hz frequency (150ms active pulse and 750ms inactive pulse), under different power densities: $56\frac{W}{cm^2}$, $31.5\frac{W}{cm^2}$, $14\frac{W}{cm^2}$.

The power density has no impact on the thermal constant, but it changes the max temperature that the system can reach. Power density affects the max temperature or T_1 from the $(T_1 - T_0) * (1 - e^{-1})$ thermal time constant equation. Higher power densities may require faster response to avoid thermal emergencies. This is due to the fact that *MaxT* is reached faster, however it does not change the required simulation time to characterize the application.

Heatsink The thermal time constant is very dependent on the heatsink resistance and capacitance characteristics. Figure 17 shows the impact of the quality of the cooling solution. Block P is power-cycled at 1Hz, with 2 different heatsinks, the default AMD Mobile heatsink (Small) and a massive Cooler Master V8 (Big). Each heatsink is evaluated with two TIMs; TIM1 uses OmegaTherm 201 thermal paste, and TIM2 uses a liquid oil-based TIM. In addition, we use a 2mm thick silver/diamond heat spreader with the big heatsink and TIM2. The best cooling solution with a large heatsink has a 5 ms thermal time constant.

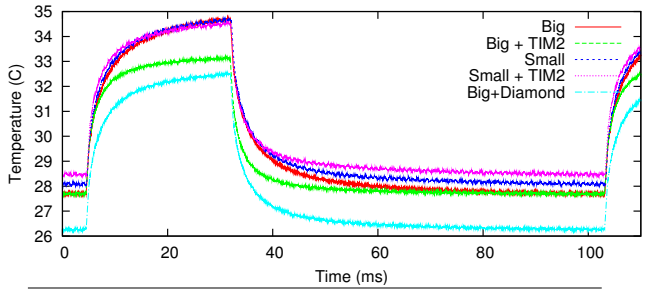

Figure 17. Thermal time response when block P is powered with different heatsink solutions.

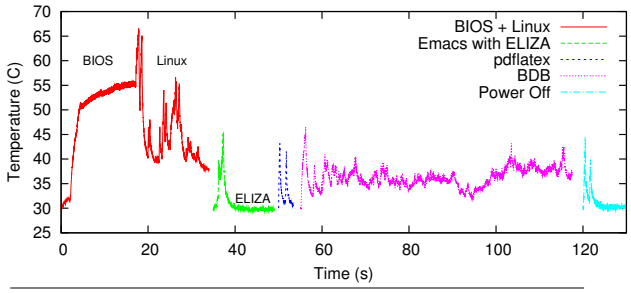

Figure 18. IO intensive applications. BIOS, Kernel, and ELIZA indicate the parts of the benchmark where the BIOS, the Linux kernel, and the Emacs ELIZA are being performed.

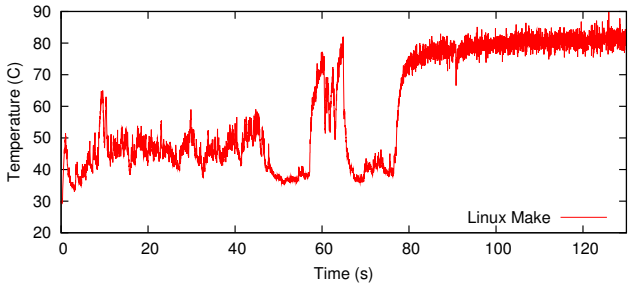

Figure 19. IO intensive Linux Make application.

The thermal response, max and min temperatures are very sensitive not only to the heatsink but also to the TIM solution. Both heatsinks are fairly close for TIM1, but as the TIM material improves, the big heatsink shows a significant cooling improvement. To push the limits of the air cooling solution, we use the diamond solution which improves the thermal characteristics reducing max and min temperature.

From Figure 17, we note that the cooling solution is a key parameter for temperature modeling. Surprisingly, it is disregarded in several papers. This is a major concern because most papers do not specify their cooling approach, or adapt their cooling solution to the new power envelope for that matter. This is especially important with thermal throttling, because different cooling solutions could cause the system to slowdown or not throttle properly.

Another important observation is that different cooling solutions have different thermal time constants, and therefore cutoff frequencies that affect the required simulation time. In our lab, we have measured between 5 and 300 *ms* thermal time constants. The conclusion is that low power cooling solutions require longer simulation times than aggressive ones. As Section 5.3 shows, the simulation time should be the maximum of the package, application, and metrics time requirements.

5.5 IO vs SPEC Workload

As the simulation time section has shown, fast power consumption changes are filtered out. Therefore, applications may not show as many thermal oscillations as IPC oscillations.

Applications are thermally variable when they have power consumption changes that last in the same order as the thermal time constant. The more abrupt the power consumption change, the more significant the thermal oscillation. This means that many common applications involving large amounts of IO, interrupts, or varied distributions of IPC have a high chance of belonging to the thermally variable category.

Figure 18 shows 5 different applications concatenated one after another. Unlike SPEC, these applications perform IO. We start by booting the system which includes BIOS and Linux kernel boot. After power up, the BIOS creates a fast spike reaching 65°C before the Linux kernel starts to boot. The Linux kernel has significant thermal oscillations due to the IO activity periods which the CPU leverages to reduce power consumption.

The second application is Emacs. The first spike of 15°C is due to loading Emacs and to perform a Verilog macro and loading ELIZA. Once we start to "converse" with ELIZA the power consumption of the system decreases significantly. We observed that this was typical in many applications that have a thermal spike while loading but that require little computation afterward leaving the CPU nearly idle. Although not possible to see in the temperature scale, each keystroke generates around 1°C thermal spike.

The third application is pdflatex. It has a behavior similar to Emacs with two thermal spikes followed by a cool down. The fourth application involves a BDB database, it is very IO intensive and shows constant changes in temperature. The last application just powers off the system. Again the same initial spike followed by a cool down is observed.

Figure 19 shows the Linux kernel compilation (Linux Make). The first 40 seconds correspond to different dependency resolution and compilations with significant IO. The right most plateau reaching around 80°C corresponds to the linking of the kernel modules.

With the exception of Linux boot, the previous applications have so much IO that they are not able to reach high temperatures. As it is shown in Table 3, the maximum temperature on average is around 19°C less than that of SPEC. In fact workloads with lowest temperature fall under this category of applications. These applications are thermally variable, with a low chance of triggering a thermal emergency. As is shown in Figure 6 and 7, many of the SPEC benchmarks show predictable thermal behavior while IO workloads mostly show variable thermal behavior. A common characteristic of many IO applications is a thermal spike followed by a cooling down phase. However, the behavior of interactive workloads is closely affected by the way a user interacts with the application. Hence the thermal behavior of these type of workloads is not easy to predict.

Another characteristic is that TC reliability metric is worse than the average SPEC application for Linux Make and System Boot. Also, Linux Make has worse SM reliability than any other application evaluated. While the *MaxT* and *GradT* for Linux Make are close to SPEC06 gcc, the IO activity makes Linux Make much worse than gcc for many reliability metrics. As Table 3 shows, lower average temperatures for IO workloads compared to SPEC applications implies lower leakage consumption.

5.6 Thermal Modeling Recommendations

The three commonly used temperature estimation approaches (using thermal and power simulation, hybrid thermal simulation-measurement, and direct measurement) have their own strengths and weaknesses. Our work provides the following recommendations for improvement that can be applied to all three approaches:

- Profile-based sampling is effective for performance and power phase detection, but it potentially induces an elevated degree of error for thermal modeling, especially in the conventional way it is used in the thermal simulation studies. A way to mitigate the error would be to reuse the power consumption calculated in similar phases as proposed in [3].
- The minimum simulation time depends on the package, the application, and the metrics being tracked. Simulation time should be selected to guarantee that the following three constraints are satisfied.
  - The packages provides an absolute minimum simulation time assuming a worst case application. For for the systems analyzed in this paper, the thermal time constant is 54 ms. Other packages measured have thermal time constants between 5 and 300 ms. As an example using the cutoff frequency from our measured systems, it is recommended a minimum simulation of 31 ms for the most aggressive cooling solutions and 2000 ms for more conservative ones. Embedded systems with even simpler cooling solutions may require longer simulation times.
  - The SPEC suite does not show decay of interval variability. Therefore, the maximum between thermal metrics and package simulation time constraints should be used to select the thermal simulation time.
  - Each thermal metric has a different simulation time constraint. For performance ($MaxT$, $GradT$) and power ($Leak$) metrics 20 billion instructions after initialization keep the error under 20%. For reliability metrics close to 100 billion instructions are required.
- Most papers using HotSpot for thermal simulation do not modify [5] the heatsink. The heatsink should be adapted to the chip power consumption. Any thermal paper should specify their cooling solution capabilities and adjust to their power demands. Ideally, papers should also report the % of time that the systems has thermal emergencies (going over a given max temperature) before and after their optimizations.
- Although SPEC is widely accepted as a representative benchmark for performance and power, we think that more work is required to determine a representative thermal suite. Since I/O intensive applications have a very different thermal behavior, we think that researchers should try to include I/O benchmarks in their workload. This is hinted at the observation that relatively similar applications like gcc and Linux Make show very different reliability metrics.
- Workload selection is always important. This is also true for thermal evaluations because most SPEC applications behave like a simple heat phase followed by a flat plateau. Many benchmarks show under 2°C oscillation after warm up. This may not be so important for papers that only care about temperature to estimate leakage power. Nevertheless, reliability, performance, or thermal modeling papers should try to incorporate IO apps, and dealII, gcc, mcf, milc, and sjeng from SPEC06.
- Commonly used functional unit granularity do not seem to be enough. The measurements show thermal discrepancies consistent with some power densities reports [4]. Current implementations of the three approaches do not provide such intra functional unit resolution. As the technology scales, this problem becomes less important because a single processor becomes a small part of the whole die. Nevertheless, it seems that further research is required to create models with finer spatial resolution for large structures like caches.

[5] The papers do not mention changes in the heatsink configuration, so we assume that they keep the default parameters.

6. Related Work

Increasing power densities have led to a need to further study spatial and temporal requirements for proper power and thermal modeling. [8] investigates the relationship between core size and on-chip hot spot temperature considering spatial low-pass filtering feature of temperature. [13, 17, 18] address the optimum locations and allocations for thermal sensors in reconfigurable and multiprocessor systems. In [16, 20, 28] authors investigate the effect of temperature-aware floorplaning and how the processor can exploit it for improved thermal management. In all these studies, the temperature is estimated using modeling and thermal simulations.

A majority the studies in microarchitecture use the SPEC benchmark suite to evaluate their proposed methods and design proposals. Nevertheless, no thermal characterization of the benchmarks itself is available to the community. (see [21, 22] for the studies addressing benchmark evaluation with objectives other than temperature). We believe that this work is the first to characterize the thermal behavior of the SPEC benchmark, using accurate thermal measurement.

Several studies investigate phase behavior of applications for characterization and optimization purposes [3, 5, 10, 24, 25]. These studies are referred to by many of the simulation-based studies addressing thermal issues in order to take advantage of skipping the simulation of redundant execution phases in the application. Except [3] that exploits reusing power traces, users hold the assumption that performance phases match thermal phases. Nonetheless, there is no study to verify this assumption. This work evaluates it by measuring the exact error imposed by this assumption through live experiments.

7. Conclusions

This paper measures real systems and provides many insights and recommendations to the computer architecture community working with thermal modeling. In addition, the main contribution of the paper is to show that commonly used statistical sampling based thermal simulation methodology has significant problems with key thermal metrics. While Simpoint has less than 1% IPC error, they have a large error for key reliability metrics.

To better understand why conventional thermal simulation methodology based on statistical sampling does not work with temperature, we provide measurements to estimate simulation time impact. We show that thermal constant is over 50 ms for a modern package, as a result without significant changes in power density thermal simulations with a few milliseconds does not make much sense. Simulators need to perform large simulations to capture thermal phases available in programs. Papers that perform sub second simulations with a realistic heatsink configuration do not have time to perform significant thermal transients.

Leveraging the measurements, we propose a list of recommendations for researchers performing thermal simulations. The evaluation also thermally characterizes SPEC00 and SPEC06 for the first time. We classify thermal applications between thermal predictable and thermal variable. In addition, we show that IO applications not seen in SPEC benchmarks show more variable thermal behavior.

Acknowledgments

We like to thank the reviewers and Dean Tullsen for their feedback on the paper. Special thanks to Sai Ankireddi and Sun Microsystems for their test chip. This work was supported in part by the National Science Foundation under grants 0546819, 0720913, and 0751222; Special Research Grant from the University of California, Santa Cruz; Sun OpenSPARC Center of Excellence at UCSC; gifts from SUN, NVIDIA, Altera, Xilinx, and ChipEDA. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] R. M. Averill, K. G. Barkley, M. A. Bowen, P. J. Camporese, A. H. Dansky, R. F. Hatch, D. E. Hoffman, M. D. Mayo, S. A. McCabe, T. G. McNamara, T. J. McPherson, G. A. Northrop, L. Sigal, H. H. Smith, D. A. Webber, and P. M. Williams. Chip integration Methodology for the IBM S/390 G5 and G6 Custom Microprocessor. In *IBM Jornal of Research and Development*, pages 681–706, Sep 1999.

[2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the 27th annual International Symposium on Computer Architecture (ISCA)*, pages 83–94, Jun 2000.

[3] A. Coskun, R. Strong, D. Tullsen, and T. Rosing. Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors. In *Proceeding of the 11th International joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 169–180, Jun 2009.

[4] B. Curran, E. Fluhr, J. Paredes, L. Sigal, J. Friedrich, Y.-H. Chan, and C. Hwang. Power-constrained high-frequency circuits for the IBM POWER6 microprocessor. *IBM J. Res. Dev.*, 51(6):715–731, 2007.

[5] A. Dhodapkar and J. Smith. Comparing Program Phase Detection Techniques. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 217–228, Dec 2003.

[6] H. Hamann, J. Lacey, A. Weger, and J. Wakil. Spatially-resolved imaging of microprocessor power (SIMP): hotspots in microprocessors. In *Thermal and Thermomechanical Phenomena in Electronics Systems*, pages 121–125, May 2006.

[7] S. Heo, K. Barr, and K. Asanović. Reducing power density through activity migration. In *Proceedings of the 2003 International Symposium on Low power electronics and design (ISLPED)*, pages 217–222, Aug 2003.

[8] W. Huang, M. Stant, K. Sankaranarayanan, R. Ribando, and K. Skadron. Many-core design from a thermal perspective. In *Proceedings of the 45th annual conference on Design Automation (DAC)*, pages 746–749, Jun 2008.

[9] W. Huang, K. Skadron, S. Gurumurthi, R. Ribando, and M. Stan. Differentiating the roles of IR measurement and simulation for power and temperature-aware design. In *Proceedings of the 2009 International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 1–10, Apr 2009.

[10] C. Isci and M. Martonosi. Phase Characterization for Power: Evaluating Control-Flow-Based and Event-Counter-Based Techniques. In *Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA)*, pages 121–132, Jan 2006.

[11] S. Jarp, R. Jurga, and A. Nowak. Perfmon2: a leap forward in performance monitoring. *J. Phys.: Conf. Ser.*, 119:042017, 2008.

[12] E. K.Ardestani, F. Mesa-Martínez, and J. Renau. Cooling Solutions for Processor Infrared Thermography. In *Proceeding of 26th Annual Thermal Measurement, Modeling and Management Symposium (SEMI-THERM)*, Feb 2010.

[13] B. Lee and T. Kim. Optimal allocation and placement of thermal sensors for reconfigurable systems and its practical extension. In *Proceedings of the 13th Asia and South Pacific Design Automation Conference(ASP-DAC)*, pages 703–707, Jan 2008.

[14] W. Liu, X. Jin, J. Chen, M.-C. Jeng, Z. Liu, Y. Cheng, K. Chen, M. Chan, K. Hui, J. Huang, R. Tu, P. Ko, and C. Hu. BSIM 3v3.2 MOSFET Model Users' Manual. Technical Report UCB/ERL M98/51, EECS Department, University of California, Berkeley, 1998. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/1998/3486.html`.

[15] F. Mesa-Martínez, J. Nayfach-Battilana, and J. Renau. Power model validation through thermal measurements. In *Proceedings of the 34th annual International Symposium on Computer architecture (ISCA)*, pages 302–311, Jun 2007.

[16] H. D. Mogal and K. Bazargan. Thermal-aware floorplanning for task migration enabled active sub-threshold leakage reduction. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 302–305, Nov 2008.

[17] R. Mukherjee and S. Memik. Systematic temperature sensor allocation and placement for microprocessors. In *Proceedings of the 43rd annual conference on Design automation (DAC)*, pages 542–547, Jul 2006.

[18] R. Mukherjee, S. Mondal, and S. Memik. Thermal sensor allocation and placement for reconfigurable systems. In *Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided design (ICCAD)*, pages 437–442, Nov 2006.

[19] N. Nethercote and J. Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. *SIGPLAN Not.*, 42(6):89–100, Jun 2007.

[20] V. Nookala, D. Lilja, and S. Sapatnekar. Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design (ISLPED)*, pages 298–303, Oct 2006.

[21] A. Phansalkar, A. Joshi, L. Eeckhout, and L. John. Measuring program similarity: Experiments with SPEC CPU benchmark suites. In *Proceedings of the 2005 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 10–20, Mar 2005.

[22] A. Phansalkar, A. Joshi, and L. John. Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite. In *Proceedings of the 34th annual International Symposium on Computer architecture (ISCA)*, pages 412–423, Jun 2007.

[23] K. Rangan, G.-Y. Wei, and D. Brooks. Thread motion: fine-grained power management for multi-core systems. *SIGARCH Comput. Archit. News*, 37(3):302–313, 2009.

[24] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proceedings of the 10th international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, volume 30, pages 45–57, Oct 2002.

[25] T. Sherwood, S. Sair, and B. Calder. Phase tracking and prediction. In *Proceedings of the 30th annual International Symposium on Computer architecture (ISCA)*, pages 336–349, Jun 2003.

[26] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-Aware Microarchitecture. In *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA)*, pages 2–13, Jun 2003.

[27] J. Srinivasan, A. Sarita, P. Bose, and R. Jude. Lifetime reliability: toward an architectural solution. In *IEEE MICRO*, volume 25, pages 70–80, 2005.

[28] Y. Wu and C. Yang. Joint exploration of architectural and physical design spaces with thermal consideration. In *Proceedings of the 2005 International Symposium on Low power electronics and design (ISLPED)*, pages 123–126, Aug 2005.