Sampling in Thermal Simulation of Processors: Measurement, Characterization, and Evaluation

Ehsan K. Ardestani *Student Member, IEEE,* Francisco J. Mesa-Martínez *Member, IEEE*, Gabriel Southern *Student Member, IEEE*, Elnaz Ebrahimi *Student Member, IEEE*, and Jose Renau *Member, IEEE,*
Dept. of Computer Engineering, University of California Santa Cruz
http://masc.cse.ucsc.edu

***Abstract*—**
Power densities in modern processors induce thermal issues which limit performance. Power and thermal models add complexity to architectural simulators, limiting the depth of analysis. Prohibitive execution time overheads may be circumvented using sampling techniques. While these approaches work well when characterizing processor performance, they introduce new challenges when applied to the thermal domain. This work aims to improve the accuracy and performance of sampled thermal simulation at the architectural level.

To the best of our knowledge, this paper is the first to evaluate the impact of statistical sampling on thermal metrics through direct temperature measurements performed at run time. Experiments confirm that sampling can accurately estimate certain thermal metrics. However, extra consideration needs to be taken into account to preserve the accuracy of temperature estimation in a sampled simulation. Mainly because, on average, thermal phases are much longer than performance phases. Based on these insights, we introduce a framework that extends statistical sampling techniques, used at the performance and power stages, to the thermal domain. The resulting technique yields an integrated performance, power, and temperature simulator that maintains accuracy while reducing simulation time by orders of magnitude. In particular, this work shows how dynamic frequency and voltage adaptations can be evaluated in a statistically sampled simulation. We conclude by showing how the increased simulation speed benefits architects in the exploration of the design space.

***Index Terms*—Infrared thermal measurement, Thermal behavior characterization, Thermal simulation, Thermal-aware statistical sampling.**

I. Introduction

TEMPERATURE is an important limiter to processor performance. It can significantly affect leakage, clock frequency, and reliability. Methods to accurately measure the dynamic thermal behavior of computing devices, at run time, are very limited. Therefore, the thermal evaluation of a processor design is highly dependent on simulation.

Simulation allows for the evaluation of ideas early in the design phase. However, the complexity in time and space required to model performance and temperature can be prohibitive. To address this, architects use various techniques to reduce the execution time and resource requirements of their simulators. Some of approaches rely on hardware-based acceleration to speed up computation [1], [2], [3]. Fast algorithms [4] are used to develop efficient thermal models as well. Conventional sampling techniques such as SimPoints [5] or SMARTS [6] are also applied to accelerate performance and power simulation. Table I shows the scope for various techniques proposed to accelerate simulation. The simulation technique proposed in this work (*TASS*) extends the use of statistical sampling to accelerate the temperature modeling stage of an architectural simulator.

TABLE I
Categorization of various acceleration techniques.

Stage	Hardware	Algorithm	Sampling
Performance	[1], [2]	-	[5], [6], [7], *TASS*
Power	-	-	[5], [6], [7], *TASS*
Thermal	[3], [8]	[4], [9], [10]	*TASS*

Statistical sampling uses a small, but representative subset of instructions for detailed simulation. This allows the simulator to fast-forward through the non-sampled sections of the program, which in turn results in significant speedups. Sampling, however, introduces challenges for the estimation of temperature. Thermal samples manifest longer dependencies to previous samples compared to performance samples. Furthermore, sampling collapses the execution time of the application, because the progression of time is not modeled during intervals which are fast-forwarded. This complicates the estimation of thermal behavior, which is time dependent.

This paper evaluates the applicability of statistical sampling to the thermal modeling of a design by directly measuring temperature of a processor at runtime. This measurement system uses an infrared camera to capture transient temperature fluctuations with a high degree of confidence. Our experimental data confirms that sampling techniques have the potential to provide accurate thermal assessments. However, the results also indicate that thermal phases are longer than typical performance phases. This means that architectural simulations implementing sampling have to address two concerns: estimate the progress of time that is collapsed via sampling, and introduce longer temperature warmup periods before computing the temperature at any given sample. Based on these insights we introduce a sampling method that provides an order of magnitude speedup in an integrated simulation of performance, power and temperature, while maintaining accuracy.

The rest of the paper is organized as follows: Section II reviews related work in this area of research, Section III

presents the direct temperature measurement infrastructure and evaluates its accuracy. Section IV characterizes the thermal behavior of processors and evaluates the impact of statistical sampling on thermal metrics. Section V discusses the computational aspects of sampled thermal simulation, and presents a framework to extend performance sampling to account for the thermal domain. Section VI concludes the work.

II. Related Work

A majority of studies regarding thermal considerations in processor design revolve around simulation approaches. Skadron *et al.* [11] introduce HotSpot, a thermal model for architectural simulation. Following these insights an architectural simulator (*e.g.*, SESC [12]) can be augmented with a power model (*e.g.*, McPAT [13]) and a thermal model like HotSpot to estimate power and thermal metrics as well. Using a similar framework, [11] studies different dynamic thermal management techniques, while [14] uses such a framework to propose a thermal-aware floorplanning scheme.

Different techniques have been proposed to accelerate the simulation speed by accelerating different components of the simulator. For example, Bartolini *et al.* [2] propose accelerating the performance characterization stage using native functional emulation. While Atienza *et al.* [1] propose an FPGA-based acceleration infrastructure. Sridhar *et al.* [3] and Che *et al.* [8] implement a GPU-based acceleration of thermal computation. Faster than traditional algorithms are considered as well (*e.g.*, [4], [9], [10]).

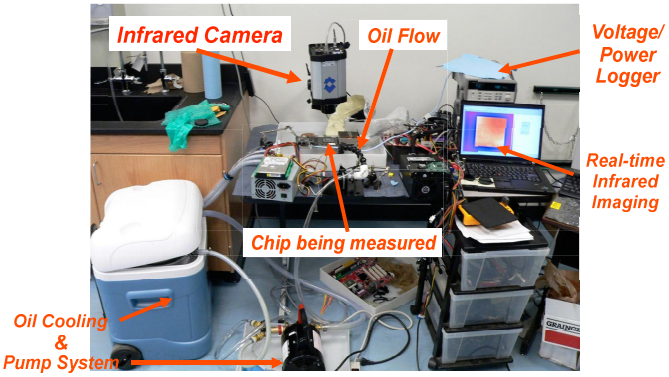
Orthogonal to these techniques is sampling [5], [6]. Sampling techniques, categorized in *Phase-based* and *Statistical* sampling, reduce the simulation time by simulating a subset of the program execution. These techniques can also incorporate hardware-based accelerating approaches mentioned earlier. In [15] the impact of phase-based sampling on temperature simulation at the architectural level is evaluated. The study shows that phase-based sampling has the potential to produce accurate thermal metrics. However, extra consideration has to be taken into account while applying sampling to a thermal simulation and Coskun *et al.* [7] do so. Even though they lower the simulation time for performance and power modeling, the computation of the temperature stage remains a bottleneck and quickly diminishes the benefit of sampling elsewhere in the simulator.

This paper presents an in-depth study of statistically sampled thermal simulation. We start with the problem of measuring chip temperature. Live experiments have been previously used in a limited way (*e.g.*, [16], [17], [18]). Their results depend either on performance counters or available on-chip sensors to obtain the temperature of a processor. Other experimental approaches directly capture the thermal profile of a processor with a fine degree of granularity [19], [20], [21], [22]. Concerns about the validation of the measurements due to the modified cooling solutions needed by these experimental approaches are also addressed in [23].

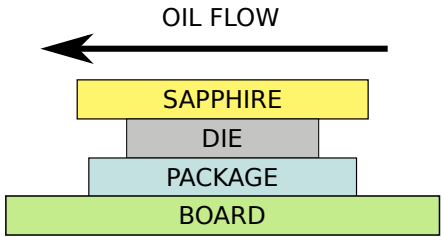
For the first time to the best of our knowledge, we evaluate the impact of statistical sampling on thermal assessments using measured temperature traces. We build on the insights from our experiments and introduce *TASS*, a technique to extend statistical sampling to the thermal domain [24]. In this work, we show the impact of such technique on enabling architects in their exploration of the design space. Then, we show an implementation of Dynamic Frequency and Voltage Scaling (DVFS) in a statistically sampled simulation.

III. Temperature Measurement

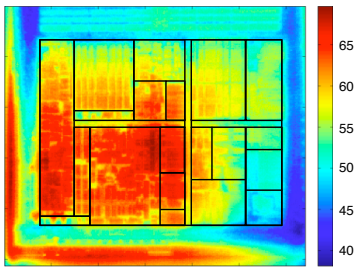
Our setup uses a FLIR SC-8000 IR camera to capture the detailed thermal map of the chip. Its sensor operates on the 3-5μm wavelength (MWIR), a range of light where silicon is partially transparent. As a result, the IR camera is capable of measuring temperature "through" the chip being tested. Modern high performance processors are manufactured using flip chips, exposing the silicon substrate. Using flip chips greatly simplifies the task of measuring junction temperatures. The system is capable of capturing data with a spatial resolution of $10 \times 10\mu m$ at 100Hz rates, and it can be used to measure different devices in a relatively simple manner. Figure 1a shows the major components of the measurement setup.



(a)



(b)



(c)

Fig. 1. (a) Measuring setup; (b) Oil-based heatsink with laminar flow, (c) a snapshot of a device with floorplan mapped on it (Temperature in Celsius).

To keep the processor operational, we implement an IR-transparent heat sink to diffuse the dissipated power under nominal operational frequency and voltage by allowing mineral oil (Fluka Mineral Oil 69808) to flow on top of the silicon substrate (Figure 1b). Fluka oil is designed for infrared spectrography.

The setup is capable of dissipating up to 100W. We keep 2 liters of oil in the oil reservoir and connect a small radiator to guarantee minimal temperature oscillations during each run. Figure 1c is a snapshot of the device running a workload. Floorplan of a processor is also mapped onto the picture.

Measurement Validation Setup: To validate the temperature measurements from our setup, we perform a series of experiments using a calibrated testchip with a $484mm^2$ die area implemented on a BGA GL771 package, as shown in Figure 2. The chip is partitioned into multiple regular blocks, and each block has its own power supply and the ability to be cycled independently. A thermal diode in each block senses temperature with sub-millisecond response time granularity. The testchip enables us to evaluate the accuracy of the infrared temperature measurements by comparing the reading from the infrared setup against the readings from the testchip's thermal sensors. We also study the impact of different cooling solutions on the observed thermal behavior.

In our study, we isolate a single block, P, which has an area of 4.84 mm^2. It is used to generate and measure different power densities and thermal responses. Two other blocks, $S1$ and $S2$, are also used for the validation process, especially with respect to spatial proximity concerns.

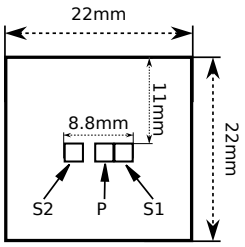


Fig. 2. Testchip floorplan.

A. Thermal Image Processing

Due to their operational characteristics, infrared cameras need to be calibrated to compensate for different material emissivities, lens configuration, temperature range for the object/material to be measured, and many other factors.

To calibrate the camera, we perform measurements involving oil on top of the surface of an inert processor at two different uniform temperatures: 289K (cold) and 344K (hot) respectively. Figures 3a and 3b show different behaviors for the infrared measurements under different oil temperatures. We observe that for cold oil (Figure 3a), the center of the image closely resembles the measured temperature, while the side pixels can have up to 6°C of error (15°C vs 21°C). The opposite effect is observed when the camera measures uniformly hot mineral oil (72°C vs 62°C). After calibration the thermal error was reduced to 3%.

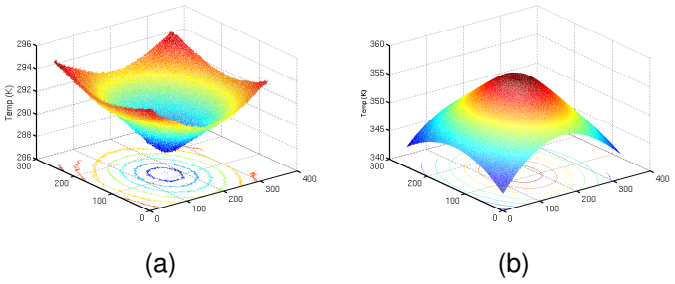


Fig. 3. (a) IR Measured temperature with low and (b) high temperature behavior for IR camera. (Temperature scale in Kelvin)

Oil Flow Direction Impact: Additional image correction is performed to compensate for the different cooling efficiencies across the die due to the direction of the oil flow. In the worst case, we observe a maximum temperature gradient of 4°C between opposite sides of the test chip, which corresponds to approximately 0.2°C correction for each $1mm$ that the oil flows over a hot block.

If all the blocks are uniformly heated, applying the $\frac{0.2°C}{mm}$ correction along the flow direction 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. 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 $1mm$ that the oil flows over a block, we adjust the correction by $0.2 * \frac{\texttt{Block Temp}-45°C}{10°C}$. We never let the correction be negative. This is a simple algorithm with linear cost that provides a fast and effective solution.

TABLE II
OIL FLOW DIRECTION IMPACT. UNCORRECTED VALUES IN PARENTHESES.

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)

To evaluate the accuracy of the correction and the impact of the oil flow, we measure the temperature for blocks P, $S1$, and $S2$ on the test chip while block P is powered with 7.8W. We apply the oil from four possible directions. Table II presents the results. 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.

B. Cooling solution

Non-uniform thermal resistance raises possible issues with the IR measurement setup [25]. The characteristics of our IR-transparent cooling solution are evaluated in order to assess the overall validity of the thermal measurement setup. We show

the steady-state and transient response of the oil heat sink, and reveal the observed differences with a metal heat sink.

To make the oil heat sink better resemble conventional metal heat sinks, we developed a twofold solution. First, a sapphire window serves as an infrared transparent composite on top of the die to compensate for the change in heat resistance. Sapphire window increases the thermal capacitance and improves lateral heat spreading. Copper has a $400\frac{W}{mK}$ thermal conductivity while sapphire only has $45\frac{W}{mK}$. Figure 1b shows the block diagram of the system with the sapphire window.

Second, we adjust the oil flow to match the cooling performance of the equivalent metal heat sink solution. However, there are physical limits or lower bounds beyond which the oil flow stops behaving like a laminar flow. To safely avoid oil flow artifacts, we set the oil flow speed to $10\frac{m}{s}$, and restrict the minimum oil thickness to $1mm$ to keep the flow laminar.

The other missing factor is the thermal interface material (TIM). Typical TIMs have thermal conductivity between 1 and 4 $\frac{W}{mK}$. For the oil solution, we use oil to soak the surface of the chip itself as an IR transparent TIM.

Steady State Response: The thermal resistance of the cooling solution determines the steady-state response, and is also referred to as the performance of the cooling solution, because it determines the final temperature of the silicon. The overall resistance is:

$$R_{overall} = R_{Si} + R_{TIM_{oil}} + R_{SW} + R_{conv\_oil} \quad (1)$$

It should match the overall thermal resistance with the metal heat sink, which is defined as:

$$R_{overall} = R_{Si} + R_{TIM_{MHS}} + R_{MHS} + R_{conv\_air} \quad (2)$$

Since the same $R_{overall}$ is desired, we can use different TIM_{oil} liquids ($R_{TIM_{oil}}$), adjust the SW thickness to linearly increase/decrease the vertical thermal resistance (R_{SW}), or control the oil thickness/speed (R_{conv_oil}).

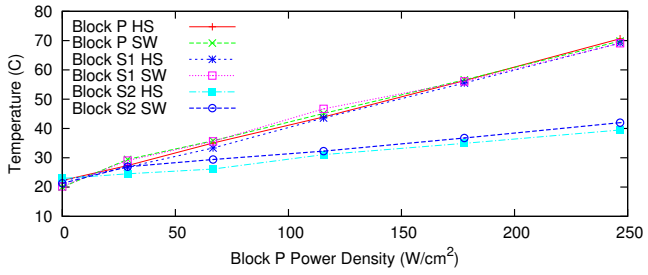


Fig. 4. Temperatures for blocks P, S1, and S2 with different constant power in block P. HS and SW stand for AMD Mobile heat sink and sapphire window respectively.

Figure 4 shows the temperature for blocks P, $S1$, and $S2$ in the test chip when different steady-state power consumptions are applied to block P. The oil flows from top to bottom. When block P gets powered from 0 to $250\frac{W}{cm^2}$, we measure and observe a consistent linear increase in temperature for the heat sink (HS) and the sapphire window (SW). This implies that the overall vertical thermal resistance of the oil and the metal heat sink are very similar.

We also measure blocks $S1$ and $S2$ to validate the lateral thermal resistance. These two blocks are located in different distances from the block P ($S1$ is located next to P and $S2$ is $5mm$ away). The difference in their temperature shows the lateral resistance of the cooling solution as the resistance creates thermal gradients. We observe that both heat sink and sapphire have a consistent slope.

Transient Response: The thermal time constant (TC) is proportional to the product of overall capacitance and resistance:

$$\tau_{overall} \propto R_{overall}(C_{Si} + C_{SW} + C_{oil}) \quad (3)$$

$$\tau_{overall} \propto R_{overall}(C_{Si} + C_{MHS}) \quad (4)$$

To keep the transient response of the oil solution the same as the metal heat sink, we would want Equation 3 and Equation 4 to match. To do so, we can adjust the oil thickness (C_{oil}) (there is physical limitation) and the SW thickness (C_{SW}). As Table III shows, sapphire has 14% less thermal capacitance ($\frac{J}{mK.mm^3}$), which means that a small thickness adjustment is enough.

TABLE III
MATERIAL PROPERTIES. R AND C STAND FOR RESISTANCE AND CAPACITANCE.

Material	R ($\frac{m.K}{W}$)	C ($\frac{J}{mK.mm^3}$)
Oil	–	1419
Silicon	120	918
Sapphire	40	2977
Copper	401	3441
Aluminum	250	2435

Sapphire also affects 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. The overall material properties are shown in Table III. As a result, copper and sapphire have equivalent thermal capacitances, with the sapphire window having a more attenuated thermal response.

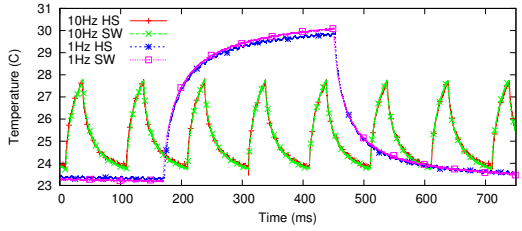


Fig. 5. Thermal transient response for a test chip when an 10Hz and 4Hz power pulse is applied. HS and SW stand for heat sink and sapphire window respectively.

To validate the transient response characteristics for the IR-transparent cooling solution, we place a 3mm thick sapphire window with a 50mm diameter on top of the test chip. The

mineral oil is temperature-controlled by a heat exchange keeping the oil temperature between 15°C to 20°C. We maintain a laminar flow with a speed of 10 $\frac{m}{s}$.

Figure 5 shows the temperature of block P when a $10Hz$ power pulse is applied to the block and the rest of the chip is idle. The same power pulse is applied for the heat sink (HS) and the sapphire window (SW). We observe that the thermal transients of the heat sink and the sapphire are very close. Authors in [25] mention that an oil cooling solution with such power pulse would have a significant error for fast transients. The measured results show that a sapphire window solves the problem. The same conclusion is drawn with $1Hz$ power pulses as well.

By combining the accuracy of fast and slow transient responses with the validation of the cooling efficiency 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.

IV. Temperature Characterization

After introducing the measurement infrastructure and studying the validity of the measurements produced, we proceed to the actual measurements. This section has two goals: first to characterize the thermal behavior of a processor running different benchmarks, second to evaluate the impact of statistical sampling on thermal metrics. We use measured thermal traces from an AMD K8-based processor, and apply sampling to them. The thermal metrics computed from the sampled trace are compared with the original trace, which shows the impact of sampling on the thermal metric, isolated from the side effects of simulation. In [15], we studied the impact of phase-based sampling. To the best of our knowledge, this paper is the first to evaluate the impact of statistical sampling on thermal metrics using measured temperature traces.

A. Temperature-Aware Metrics

Temperature has a different impact on several key design factors such as timing integrity, reliability, leakage power, and cooling cost. Based on the previous work [15], we define three categories of metrics: Timing, Reliability, and Power. The timing category tracks maximum temperature ($MaxT$) and maximum temperature gradient across the chip ($gradT$). The reliability category consists of 5 metrics: Electro Migration (EM), Stress Migration (SM), Time-Dependent Dielectic Breakdown ($TDDB$), Negative Bias Temperature Instability ($NBTI$), and Thermal Cycling (TC). We only track leakage power ($Leak$), because only the temperature dependent component of the power consumption for the experiment is of interest. For detailed information on each metric please see [15].

B. Characterization Setup Parameters

To gather IPC traces we use the utility pfmon [26]. The length of each sample in our experiments is $10ms$, while just about 2% of the instruction population is periodically sampled.

For each category of thermal metrics (performance, reliability, power), we report the metrics based on constants specified for 65nm technology files. We measure an AMD K8-based processor, running at 1.7GHz. The spatial and temporal resolution of each temperature sample is 10×10 um, and 10 ms respectively.

1) Workloads: 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. Workloads 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 workloads involving I/O: System Boot, Linux make, pdflatex, emacs, and BDB.

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

C. SPEC Characterization

Table IV presents the performance, reliability, and energy thermal metrics for our target processor executing SPEC00, SPEC06, and IO workloads. $MaxT$ and $GradT$ are in °C. The reliability metrics are normalized to a Mean Time Between Failures (MTBF) of 57.08 years, in the same fashion leakage numbers are 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 reason for this lack of correlation is that $MaxT$ and $GradT$ report the maximum temperature or the maximum temperature difference. These values are not closely 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.

By analyzing the correlations in Table IV, 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, the thermal behavior for the SPEC suite can be categorized as $Smooth$ and $Varied$. The $smooth$ category includes the applications that are thermally predictable. For example, the thermally predictable category (Figure 6a) for the SPEC06 benchmarks have a predictable plateau after the initial warmup. $Varied$ category comprises the applications that are thermally variable. Figure 6b shows over 5°C oscillations once the warmup is over.

D. Length of Thermal Simulation

Figure 7 shows the error or inaccuracy suffered when only a subset of the application's execution 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.

Performance and power thermal metrics ($MaxT$, $GradT$, $Leak$) require longer simulation times, requiring over 10

TABLE IV
PERFORMANCE AND THERMAL METRICS FOR SPEC00, SPEC06, AND IO WORKLOADS.

	Apps	IPC	MaxIPC	*MaxT*(°C)	*GradT*(°C)	EM	SM	TDDB	TC	NBTI	Power
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

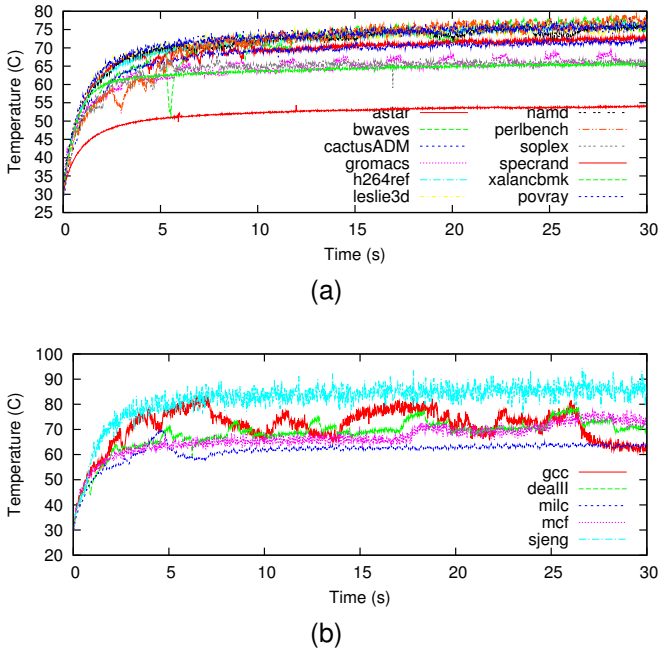


Fig. 6. (a) Smooth and (b) Varied application categories from thermal standpoint.

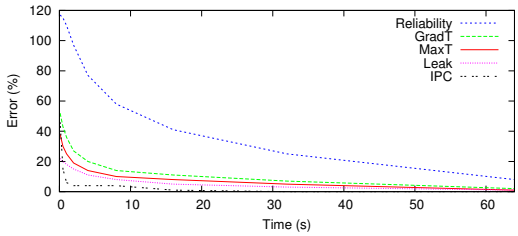


Fig. 7. Impact of simulation time on accuracy.

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 plot selects a subsection of the application execution after warmup. The same type of results holds with warmup.

The longer execution time required for thermal simulations demands faster simulation methodologies.

E. Studying Sampling Impact on Measured Traces

The most common method for applying statistical sampling to architectural simulation is SMARTS [6]. Periodic sampling is used to take many small samples at evenly spaced intervals throughout program execution [1] . It also requires a warmup period and detailed simulation to make sure that the microarchitectural state is valid at the time it is sampled.

To determine this error, we gather a thermal trace for the entire processor, which is measured by an IR setup during program execution. We also gather performance metrics, in real-time, from the program executing within the same processor. A statistical sampling approach similar to SMARTS is used to gather thermal samples from the traces. We evaluate the impact on the accuracy compared to the full thermal trace.

We observed a key problem with Sampling. The final temperature of each sample can be very different from the starting temperature of the following sample. This leads to inaccurate simulations if the samples are concatenated for thermal simulation. Second, samples are too short to capture significant transients within them. The overall evaluation of the temperature is obtained by aggregating the samples, and the transition of temperature between the samples.

Figure 8 shows the impact of statistical sampling on various thermal metrics. There are three comparison points. *Full* corresponds to the complete execution of the program, *i.e.*, no sampling is applied. *Oracle* sets the initial temperature of each sample to the correct temperature. This information is not available in a sampled simulation (Next Section introduces a framework to preserve such information). *Typical* sets the initial temperature of the current sample to the final temperature of the previous sample.

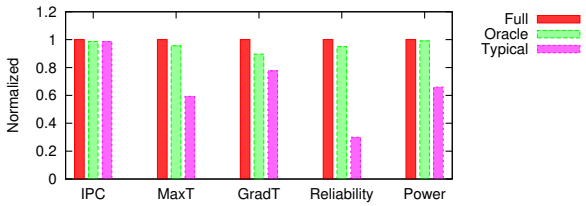


Fig. 8. Impact of statistical sampling on the accuracy of thermal simulation.

As expected, IPC is well captured by all the methods. *Typical* has around 20% error for *MaxT*. The most problematic metric is reliability. For the thermally variable subset of applications, the difference is even worse than the reported normalized average values.

Insights Derived from Experimental Measurements:

- Thermal phases are longer than performance phases.
- Thermal metrics require longer simulation time.
- The simulations can possibly yield correct results if the *initial temperature* for each sample is estimated correctly.

V. Thermal-Aware Sampling

The experimental data confirms accurate thermal simulation using sampling. However, the required warmup period and correct estimation of the initial temperatures for each sample present significant issues.

In order to decouple thermal computations from the effects of sampling in the performance stage, thermal simulations perform sampling first, and then reconstruct the power trace. Afterwards, a full thermal model computation is performed on the reconstructed power to generate a detailed temperature trace. Hence, the thermal stage performs the same amount of computation with or without performance sampling (We evaluate this method as *SS* in Section V-D). The slower state transitions in temperature compared to power and performance suggests the potential for fewer thermal calculations. Previous studies have considered such potential, averaging the power for

[1]Samples are evenly spaced regarding the number of instructions between them. The actual time between samples varies as performance varies

10K to 1M cycles in the majority of cases. Finally the thermal computation is performed for a given timestep [11], [27], [28]. Timesteps are set to be around an order of magnitude smaller than the thermal time constant (TC) to provide accurate transients, thus avoiding thermal computations for every cycle.

Nevertheless, while sampling methods drastically reduce the time spent on performance simulation, the execution time for thermal simulation remains a limiting factor.

A. TASS Method

A significant challenge when extending sampling methods to the thermal domain is that unlike performance sampling, temperature sampling requires a very long warmup. In other words, temperature has a strong dependency on the previous state, which is at odds with the idea of sampling. However, as mentioned earlier, temperature state changes much more slowly than power and performance. This is leveraged to avoid excessive thermal computations.

In general, precise estimation of temperature at time $T_1 = t$ depends on correct estimation of three parameters:

- temperature at time $T_0 = t - \delta$
- power consumption for the $(T_0 : T_1)$ interval
- the length of the interval, δ

For sampling, δ is the length of the thermal interval [2]. With a longer δ, more distribution information may be skipped. Therefore, δ can be adjusted for appropriate trade-offs between speed and accuracy. We set the length of the thermal interval equal to the performance sampling intervals ($TPr = 1$, TPr explained later in this section). Note that thermal sampling frequencies higher than the frequency of performance samples do not provide temperature traces that are any more accurate. A point which previous methods have neglected, and as a result they use a constant timestep, which is independent of the performance sampling interval (For example see [28]).

Furthermore, the beginning of thermal and performance intervals must be synchronized. The only temperature value computed for the thermal interval is located at the end of the interval. Accurate estimation of temperature at T_0 itself is interdependent on the previous power and δ values. Therefore, the main challenge is the correct estimation of *Power* and *Time* for each interval. However, these values are only available for the samples rather than the whole sampling interval. For the fast-forwarded intervals we predict these missing parameters.

The length of sampling intervals is always a fixed number of instructions (Table IX). However, due to the time variant nature of *CPI*, the actual length of each interval varies in terms of cycles. Using Equation 5, the problem of prediction of time is transformed to the problem of prediction of *CPI*.

$$T_i = \sum_{k=1}^{i} \delta_k, \delta_k = \frac{Cyc_k}{ClkFreq}, Cyc_k = CPI_k \times Inst \qquad (5)$$

[2]Thermal models like HotSpot [11] or SESCTherm [29] have an internal time step that defines the temporal resolution of thermal computation and should not be mistaken with the δ variable defined here.

1) Estimation For The Fast-Forwarded Intervals: We evaluate different methods for predicting *CPI* and power, as summarized in Table V. The simulation setup is described in Section V-C. The benchmarks are executed in full timing simulation mode. We divide the execution into intervals summarized in Table IX, and compare the estimated *CPI* or power of the intervals with the actual values. Figure 9 shows the estimation accuracy for each method.

TABLE V
DIFFERENT METHODS TO PREDICT CPI AND POWER OF THE INTERVALS BETWEEN SAMPLES.

Method	Description
Naive	Average value across all the benchmarks
Last	value of the last sample
MA	**M**oving **A**verage of last 3-7 samples, similar to *WMA*, but all the weights are set to 1.
WMA	**W**eighted **M**oving **A**verage of last 3-7 samples: as formulated in Equation 6.

For *CPI*, *Last* and *WMA* generate the most accurate predictions. For power, *MA* and *WMA* result in the best predictions. Based on this, we use *WMA* for both *CPI* and power prediction as it also results in a better standard variation, and better thermal results eventually. *WMA* works as a filtering mechanism to smooth high frequency power and performance spikes. power spikes in the samples affect the reconstructed trace, which will increase the error in the reconstructed temperature even more, given that temperature effects tend to remain longer due to the thermal time constant effect. That is the reason we consider standard deviation as a secondary factor when selecting the prediction method.

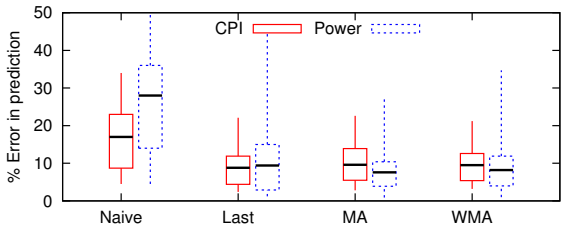


Fig. 9. Error in the prediction of CPI and Power for intervals between samples.

$$\Theta_i = \frac{\sum_{k=0}^{n-1} \alpha_k \times \theta_{i-k}}{\sum_{k=0}^{n-1} \alpha_k}, \alpha_k = \frac{1}{2^k} \qquad (6)$$

In our experiments, we set $n = 5$ in Equation 6. The *measured CPI* and *measured* power from intervals i to $i-5$ are used to obtain an *estimated CPI* and *estimated* power for the whole interval i. We also perform the evaluation with $n = 3$ and $n = 7$, with the result being more or less similar ($n = 3$ results in slightly more accurate predictions, but with a higher standard deviation).

Instead of applying a filter, we can increase the length of each sample (i.e. simulating more instructions for each sample). However, this leads to longer simulation times, since

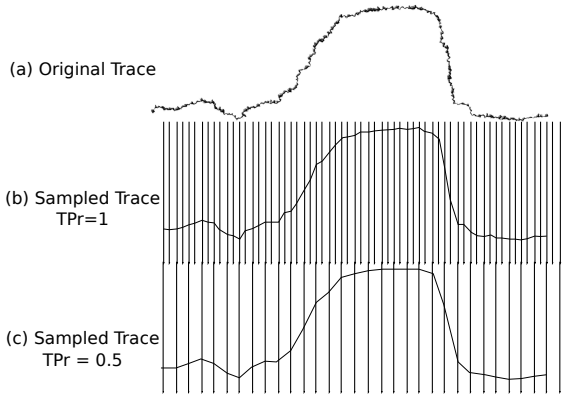


Fig. 10. Depiction of TPr effect. A TPr of lower than 1 results in an even less thermal computation demand as it reduces the number of times the thermal model is called. Consequently, the generated thermal trace has a lower resolution.

detailed timing simulation is slow. The simple filtering process relaxes the demand for longer sampling length, while still providing stable estimations. As the evaluation in Section V-D shows, the recommended performance-driven statistical sampling parameters (*SS*) yields inaccurate results due to this spike sensitivity during reconstruction.

2) Sampling Parameters: SMARTS [6] specifies the length of each sample and the sampling intervals based on the observed variability in the program behavior. Given the standard deviation (σ) and the mean (μ) of the *CPI* of the sampled population, coefficient of variation of samples are computed ($\hat{V} = \frac{\sigma}{\mu}$). The number of samples can be obtained by $n \geq [(z \cdot \hat{V}).\varepsilon]^2$, where $z = 100(1-\alpha/2)$, $(1-\alpha)$ is the desired confidence level, usually set to 0.95 or 0.98. $\varepsilon \cdot \bar{X}$ is the confidence interval. Also the impact of detailed warmup (*DW*) has to be taken into account as an interdependent factor in determining the length of samples in detailed timing (*DT*).

We use a similar methodology. In addition, the coefficient of variability of the power samples can also be used to guide parameter selection. Our observation is that longer samples need to be measured to provide stable temperature results. Haskins *et al.* [30] explain efficient memory warmup approaches, we implement a similar technique to reduce the length of warmup in performance stage. Table IX shows the parameters.

When thermal phases are longer than performance phases, it implies using even longer thermal intervals than performance intervals. We define the real number **T**hermal to **P**erformance sampling **r**atio (*TPr*) as follows:

$$TPr = \frac{freq_{Thermal-samples}}{freq_{Performance-samples}}, 0 < TPr \leq 1 \qquad (7)$$

For example, $TPr = 0.5$ means that thermal samples occur in half the frequency that performance samples do, *i.e.*, each thermal interval is as long as 2 performance intervals in terms of number of instructions. Figure 10 depicts the impact of *TPr*. Figure 10a shows a sample thermal trace. Figure 10b shows the trace with a temperature sample for each performance sample ($TPr = 1$, each vertical line is a sample.). Figure 10c shows the trace obtained by $TPr = 0.5$. The number of thermal samples is half the performance samples. As a result, the trace is generated with a lower resolution, but the thermal computation demand is reduced to half.

We perform a set of experiments with different thermal sampling intervals, while the performance sampling intervals stay the same. The setup is explained in Section V-C. The results are shown in Figure 11. The y-axis shows the Root Mean Square Error (RMSE) of comparing the resultant thermal traces against the shortest thermal interval of 10K. The results confirm that as long as the thermal statistics are gathered at the end of thermal interval, where the temperature is estimated, longer thermal sampling intervals can be tolerated. As a results, $TPr < 1$ can be used to further accelerate the simulation speed.

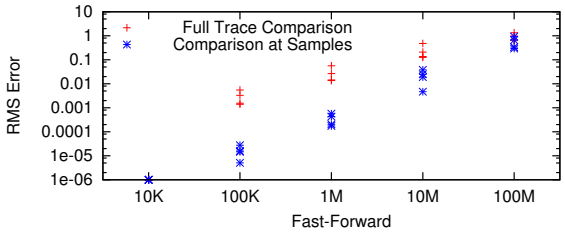


Fig. 11. Accuracy of estimated temperature at thermal samples with different thermal sampling intervals

B. All Together: The Simulation Flow

Figure 12 shows the simulation flow for Full simulation, SMARTS, *SS*, and *TASS* in our *functional emulation first* setup.

Full Simulation: the functional emulation keeps track of program flow and sends the dynamic stream of instructions to the timing model. The timing model generates the performance information, and passes activity counters from the functional units to the power model for power estimation. The estimated power and execution times are passed to the thermal model to compute the evolution of temperature transients.

In our setup, there is a feedback path from the thermal model to both power and timing models. The feedback to the power model updates the leakage power according to the current temperature profile of the chip. The feedback to the timing model notifies the model of the thermally induced performance penalties, *e.g.*, the performance degradation due to thermal throttling.

SMARTS sampling: The SMARTS [6] statistical sampling simulation selects a subset of the instructions executed by the functional emulator to be sent to the timing model. The timing model is the slowest component in the simulation, and the goal of sampling is to reduce the load on this component. To avoid a bias due to stale microarchitectural state at each sample, Memory Warmup is performed for the non-sampled instructions to keep track of the memory references. Note that no detailed timing simulation is performed yet. Immediately prior to executing the sample, the timing model will receive a predefined number of instructions in order to warm-up the pipeline structures.

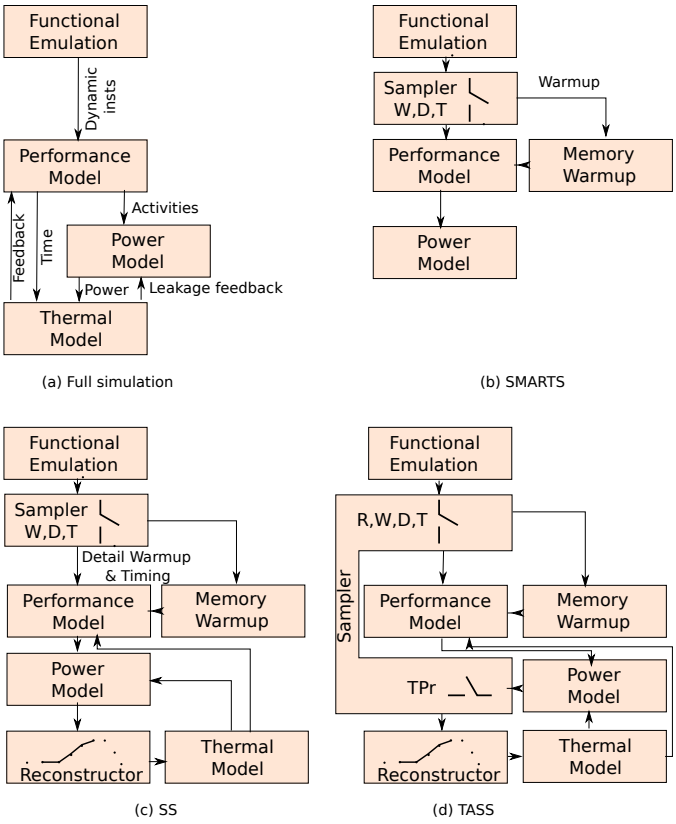


Fig. 12. The flow of simulation in Full (non-sampled) simulation, SMARTS, *SS* and *TASS*.

The majority of instructions are fast-forwarded with memory warm-up which is fast as it lacks any timing or power information. Since the execution time collapses in a sampled simulation (there is no timing information during warm-up modes), SMARTS cannot perform accurate temperature simulation.

SS: *SS* performs statistical sampling at performance and power stage similar to SMARTS. It adds a reconstruction module to reconstruct time and power, and shield the sampling from thermal computations. The thermal model is called periodically every 100K cycles. *SS* is oblivious to the sampling at performance and power stages. Table IX summarizes the parameters used for *SS*.

TASS: *TASS* performs statistical sampling at performance and power stage similar to *SS*, and also extends the sampling to the thermal stage. Once the samples are passed to the timing model and the power estimation is performed, the result is sent back to the sampler to perform extra sampling (if required, decided by TPr defined in Section V-A2) for the thermal computation. Then the reconstruction module reconstructs the execution time, and power, and eventually generates the temperature trace. *TASS* incorporates the sampling in performance and power stage in the thermal computation. Unlike *SS*, *TASS* synchronizes the beginning and end of temperature and performance samples to minimize the number of calls to the thermal model. The size of each simulation mode (Rabbit, Memory Warmup, Detail Warmup and Timing, defined in Table VII), as well as TPr are predefined through the process explained in Section V-A2 and summarized in Table IX. For *TASS*, the sampling parameters have been revised for thermal simulation (longer samples).

Pros and Cons: In fact, sampling trades off accuracy for simulation speed in a bounded manner. While the speed increases by orders of magnitude, the accuracy loss is in the order of couple of percentage points in the metrics of interest. This trade-off is controlled by selecting the sampling parameters as mentioned in Section V-A2 and explained in detail in [6]. More frequent and longer samples generates more accurate results, but the simulation will be slower.

Architects have to be aware that the optimum parameters vary per application. However, it is desirable, and practical, to run the evaluated experiments with only one set of sampling parameters. A conservative sampling parameter selection can ensure the accuracy for all the applications (*i.e.*, slightly more and longer samples). The sampling parameter selection procedure is highlighted in V-A2, and is discussed in more detail in [6].

C. Simulation Setup

Table VI lists the methods we evaluate in this work. *SS* uses the same sampling parameters as suggested in [6]. It perform thermal computation after reconstructing power trace with 100K cycle timestep.

TABLE VI
SIMULATION METHODS EVALUATED IN THIS WORK.

Method	Description
Full	Full simulation, (no sampling)
SS	Statistical-based sampling (Perf. + Power sampling)
TASS	Thermal-aware SS, (Perf. + Power + Thermal sampling)

TABLE VII
SIMULATION MODES.

Phase	Description
Rabbit	Fast-forward emulation or native co-execution
Memory ***W***arm-up	Tracks Memory references to maintain accurate state
Detail Warmup	Cycle-accurate modeling to warmup the pipeline, statistics are discarded
Timing	Cycle-accurate timing modeling

For performance simulation, we use a modified version of SESC [12] that uses QEMU [31] as the functional emulator executing user mode ARM instructions. The simulator offers 4 different execution modes, explained in Table VII, to support sampling. We configure SESC to pass activity counters to McPAT [13] (every 100K instructions max) which we use for calculating power. We use a modified version of SESCTherm [29] as our thermal model. SESCTherm uses an approach similar to HotSpot [11] to solve the thermal equations. Our implementation scales leakage power consumption according to temperature and device properties.

We simulate two single core processors: one is an Intel Nehalem-like high performance (HP) core, and the other is an AMD Bobcat-like low power (LP) mobile core (Table VIII). Table IX shows the sampling and the thermal simulation parameters respectively.

TABLE VIII
ARCHITECTURAL PARAMETERS.

	Parameter	Value		Parameter	Value
HP	Freq	3.0 GHz	**LP**	Freq	1.6 GHz
	I$	32KB 2w (2c hit)		I$	32KB 2w (2c hit)
	D$	32KB 8w (3c hit)		D$	32KB 2w (2c hit)
	L2	1MB 16w (12c hit)		L2	512KB 4w (12c hit)
	Mem.	180 cyc		Mem.	90 cyc
	BPred.	Hybrid 76Kb mem		BPred.	Hybrid 38Kb mem
	Issue	4		Issue	2
	ROB	256		ROB	56
	IWin.	48		IWin.	20
	Reg(I/F)	128/128		Reg(I/F)	80/64
	Tech.	32 *nm*		Tech.	32 *nm*

TABLE IX
SAMPLING PARAMETERS. R, W, D, T DEFINED IN TABLE VII.

Method	Parameter
SS	R = 0, W = 997e4, D = 2e4, T = 1e4
TASS	R = 257e4, W = 250e4, D = 2e4, T = 7e4, History Size = 5, TPr = 0.5 unless mentioned otherwise

The benchmarks that we selected from CPU2000 and CPU2006 suites are shown in Table XI. We also use the same set of metrics explained in Section IV-A

D. Simulation Results

We cluster the benchmarks in categories as shown in Table XI, and averaged the result of HP and LP configurations as they are statistically similar. The accuracy of each benchmark is calculated in comparison with *Full* and average result for each category is reported. We also report the minimum and maximum error for each category.

1) Accuracy: The results shown in Figure 13 show *SS* suffers from the sampling parameters that suits the performance sampling, and are not adjusted for thermal. As a result, it does not provide accurate results as expected (errors within 20% of the non-sampled simulation on average).

On average, across all the benchmarks and metrics, *TASS* performs accurately with 3.6% average error.

2) Speed: The last column in Table XII shows the simulation speed for each method on average across all the benchmarks. An observation is that thermal simulation accounts for a large portion of the simulation time in these methods. The reason is that sampling reduces the simulation time for performance simulation, whereas thermal simulation time stays the same. While in a non-sampled simulation the temperature simulation accounts for 20% of the simulation time, in a sampled simulation, this ratio increases to over 80%.

The prohibitively large portion of simulation time spent on thermal simulation shows the need for various techniques to reduce the time spent on thermal computation. Fast thermal models (for example see [10]) or parallelized computation (for example through GPGPU implementation of the thermal solver [3]) can be used to reduce this time. Extending the sampling to thermal computation is another way of reducing the computation demand. Nonetheless, all the above mentioned techniques are orthogonal and can be combined to speed up the thermal simulation even further.

Full runs at 0.76 *MIPS*[3] on average with thermal simulation. *SS* increase the simulation speed up to 2.8 *MIPS* respectively due to reduction of simulation time in the performance stage. However, it does not deploy thermal sampling. *TASS*, on the other hand, extends sampling to the thermal stage, and reaches a maximum speed of 30 MIPS and an average speed of 18 MIPS. These results are generated with $TPR = 0.5$ (explained in Section V-A2) for *TASS*, which translates to thermal fast-forwarding of 10M instructions.

[3]Million simulated Instruction Per Second. Experiments run on an AMD Opteron(tm) processor 6172 with 120GB of memory.

TABLE X
THERMAL PARAMETERS.

Parameter	**HP**	**LP**
Package RC	10 ms	
Ambient Temp.	308 K	
Chip Thickness.	0.83 mm	
Chip Area	65 mm^2	26 mm^2
Throttling Threshold	363 K	348 K

TABLE XI
EVALUATED BENCHMARKS.

Suite	Category	benchmark
CPU2000	Varied	swim, gcc, mesa, facerec, lucas, bzip2
	Smooth	gzip, mgrid, applu, vpr, crafty, twolf
CPU2006	Varied	gcc, milc, dealII, mcf
	Smooth	perlbench, soplex, astar, povray, namd, h264ref

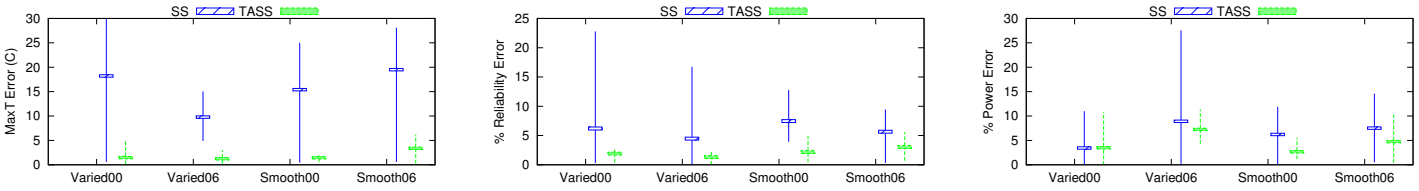
TABLE XII
BREAKDOWN OF SIMULATION SPEED IN MIPS WITH AND WITHOUT THERMAL COMPUTATIONS.

Method	Speed w/o thermal R	W	D	T	Speed w/o thermal	Speed w/ thermal
Full	-	-	-	1.0	1.1	0.8
SS	-	49	0.4	0.3	44	2.8
TASS	20	19	0.9	1.0	22.5	18

Extending sampling to the thermal stage accelerates simulation speed around 7 times, while the accuracy degrades from 4.7% to 6.7% on average. It also affects the maximum error with the same rate. Table XII shows the breakdown of execution time for each simulation mode, as well as the simulation speed with and without thermal sampling. The execution time with thermal for the thermal-aware method (*TASS*) is an order of magnitude less than their thermal-unaware counterpart (*SS*). *SS* without thermal is faster than *TASS* as the sampling parameter for *TASS* is adjusted for thermal accuracy. The execution time at different stages of simulation is distributed more or less evenly in the same order with thermal time now being around 25% of total execution time for both methods. Therefore, according to Amdahl's law, further reduction in thermal simulation time once the sampling is employed will not result in radical increase in the overall simulation speed (33% maximum).

E. Case Study: Dynamic Thermal Management

Modern processors adapt to the thermal effects, and implement different DTM techniques and policies. In this section we explain how these techniques can be implemented in a statistically sampled simulation. Hence, we implement two DTM policies: Throttling and DVFS. This section shows an



(a) Max Temperature (b) Reliability (c) Power

Fig. 13. Average, minimum, and max error of different methods.

example of how architects can benefit from an integrated sampling infrastructure (performance, power, and temperature), as it drastically reduces the simulation time, and increases their productivity and depth of exploration.

For throttling, when the chip temperature exceeds the threshold (*e.g.*, 90°C, close to junction temperature), the processor stops the execution to prevent physical damage to the chip. Engagement of throttling and resuming of the execution involves a delay. The delay is included into the time it takes the processor to cool down. The performance impact of throttling is fed back to the performance domain as well.

DVFS is another technique that we implemented. With DVFS, the processor adapts the operating voltage and frequency of its core to manage its thermal behavior. DVFS policies are managed by the operating system. The temporal resolution is OS ticks, which is usually in order of milliseconds. In statistically sampled simulation, we update DVFS state at each sample, and the state remains unchanged for the intervals between samples. The interval between samples are usually in order of millisecond as well (the exact value changes based on the processor's performance).

TABLE XIII
DVFS STATES AND POLICIES.

State	Frequency (MHz)	Voltage (V)	$DVFS-P$	$DVFS-T$
S0	3000	1.1	$T < 86$	$T < 80$
S1	2700	1.045	$86 \leq T < 87.5$	$80 \leq T < 85$
S2	2400	1.00	$87.5 \leq T < 88.5$	$85 \leq T < 87$
S3	2100	0.945	$88.5 \leq T < 90$	$87 \leq T < 90$
TT	-	-	$T > 90$	

Any changes in voltage and frequency of the core causes CPI to change. The reason is the relative difference in core's speed compared to the memory subsystem. In the sampled simulation, the estimated length of each interval is obtained as formulated in Equation 5. Both parameters affected by DVFS appear in this formula (*ClkFreq* and *CPI*). As a result, the length of each sampling interval will expand in time by lowering the frequency and voltage, and shrink otherwise. As it can be seen, by updating DVFS state once per sampling interval, the implementation easily integrates into the *TASS* framework. Table XIII summarizes different techniques and policies we implement for our case study.

The DTM policies we implement are as follows:

Throttling: When a block in the processor reaches a trigger threshold (90°C, defined in Table X), the clock is gated and the processor stops processing. It will reduce the power consumption of the chip down to the static power. The processor remains throttled until the temperature drops well below the defined threshold (*e.g.*, 86°C).

DVFS-P: The processor implements 4 frequency-voltage states, as summarized in Table XIII. The goal is to maintain the processor's performance level while keeping the temperature below the throttling threshold.

DVFS-T: The processor implements 4 frequency-voltage states, as summarized in Table XIII. The goal is to minimize critical temperature across the chip, and lower the average heat dissipation.

1) Impact on performance and temperature: We ran the set of benchmarks for 5 billion instructions (around 2-5 seconds of execution). We evaluate the $Throttling$ and $DVFS$ policies. Figure 14a shows the impact of DTM policies on processor performance. Figure 14b shows the average temperature of the hotspot across the chip throughout the execution. The maximum temperature does not exceed 90°C, because at that point throttling will be in effect. While $DVFS-T$ substantially reduces the hotspot, the reduction causes performance loss. $DVFS-P$, however, improves both the performance and thermal profile. This is done by maximizing utilization of higher frequency states while keeping the hotspot temperature just below the throttling threshold. $DVFS-P$ and $DVFS-T$ improve *EnergyDelay* product of the processor by 24.8% and 25.2% respectively.

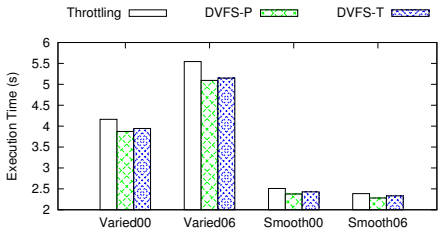
Table XIV shows the utilization of each state on average. For example, with $Throttling$ policy, 100% of instructions are executed in $S0$. No execution is performed while the core is throttled. Nonetheless, the processor spends 26.6% of the time in throttling state. $DVFS-P$, on the other hand, had a different distribution for the number of executed instructions in each state. It reduced the throttling time to less than 7%.

As it can be inferred from Figure 14a and Figure 14b, the *Varied* category of benchmarks provides more potential for the processor to exploit DVFS for higher energy efficiency, and to reduce the average hotspot formation across the chip.

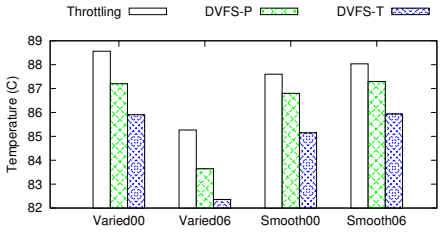
TABLE XIV
DISTRIBUTION OF EXECUTED INSTRUCTIONS AT EACH DTM STATE (S0-S3). FOR TT, THE PERCENTAGE OF TIME SPENT IN THROTTLING IS PROVIDED. NO INSTRUCTION IS EXECUTED IN TT.

State	$Throttling$	$DVFS-P$	$DVFS-T$
S0	100	41.2	28.7
S1	0	19.6	20.8
S2	0	20.6	23.0
S3	0	18.6	27.5
TT	26.6	6.8	4.0

2) Length of Simulation: As discussed in IV-D, thermal effects develop slower than performance effects. Hence, thermal



(a) Execution Time



(b) Average Hotspot Temperature

Fig. 14. Performance of each DTM policy.

studies require longer simulations. For example, comparing *Throttling* and $DVFS-P$ for 500 million instructions (after skipping 1B), both DTM techniques perform equally in terms of impact on performance of the processor and its thermal behavior. The reason is that this amount of instructions on a modern processor accounts for less than 0.4 second of execution. It is not long enough time for the progress of temperature effects. However, slow simulation speed prohibits the architect from performing longer simulations.

In addition, adjusting the parameters for a DVFS policy involved several set of such runs for the set of benchmarks. Full integrated simulation of 500M instruction for our configuration takes around 10 hours for each benchmark. Sampled simulation at performance stage only and performing full thermal computation (*SS*) reduces this time to 3 hours, which is still prohibitive even for a short 500M simulation. Sampled simulation at both performance and thermal stage (*TASS*) takes less than 15 minutes to finish, which makes it possible for architects to run longer simulations and to adjust the configuration through several iterations. This renders a fast technique such as *TASS* a necessity.

VI. Conclusion

This work explores several key aspects of the thermal evaluation of modern processor designs. We start from experimental methods to capture temperature metrics during run time. Then we study the techniques for thermal-aware architectural simulation and how to increase their speed and accuracy.

We describe the implementation and validation of a mechanism for directly measuring the thermal characteristics of processors nominally at run time. The measurement setup uses an infrared camera for studying the thermal behavior of a computing device. Using such measurement system, we study and evaluate the impact of direct deployment of statistical sampling on thermal simulation. The experimental data confirms that sampling can potentially result in accurate estimation of thermal behavior; However, the sampling-based architectural simulation should first consider that thermal phases do not match the performance phases and longer thermal phases require a longer warmup period. Secondly, accurate sampled evaluation depends on the accurate estimation of the initial temperature of the samples.

For a typical integrated architectural simulation, thermal computations can take over 80% of the total simulation time after applying sampling to the performance and power simulation stages. We present a technique for applying sampling to the thermal modeling stage. The evaluation shows that our approach reduces the thermal computation time to as low as 25% of the total simulation time, and speeds up the simulation by an order of magnitude. All while still providing accurate results within 3.6% of the non-sampled simulation. We discuss implementation of runtime adaptations in the processor (*e.g.*, DVFS), and through a case study, we then show how architects can perform a more extensive exploration of the design space.

References

[1] D. Atienza, P. Del Valle, G. Paci, F. Poletti, L. Benini, G. De Micheli, and J. Mendias, "A fast hw/sw fpga-based thermal emulation framework for multi-processor system-on-chip," in *Proceedings of the 43rd annual Design Automation Conference*. ACM, 2006, pp. 618–623. 1, 2

[2] A. Bartolini, M. Cacciari, A. Tilli, L. Benini, and M. Gries, "A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores," in *Proceedings of the 20th symposium on Great lakes symposium on VLSI*. ACM, 2010, pp. 311–316. 1, 2

[3] A. Sridhar, A. Vincenzi, M. Ruggiero, and D. Atienza, "Neural network-based thermal simulation of integrated circuits on gpus," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, no. 1, pp. 23–36, 2012. 1, 2, 11

[4] Y. Yang, Z. Gu, C. Zhu, R. Dick, and L. Shang, "ISAC: integrated space-and-time-adaptive chip-package thermal analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 86–99, 2007. 1, 2

[5] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *ACM SIGARCH Computer Architecture News*, vol. 30, no. 5. ACM, 2002, pp. 45–57. 1, 2

[6] R. Wunderlich, T. Wenisch, B. Falsafi, and J. Hoe, "SMARTS: accelerating microarchitecture simulation via rigorous statistical sampling," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*. IEEE, 2003, pp. 84–95. 1, 2, 7, 9, 10

[7] A. Coskun, R. Strong, D. Tullsen, and T. Simunic Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. ACM, 2009, pp. 169–180. 1, 2

[8] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using cuda," *Journal of parallel and distributed computing*, vol. 68, no. 10, pp. 1370–1380, 2008. 1, 2

[9] Y. Zhan and S. Sapatnekar, "High-efficiency green function-based thermal simulation algorithms," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 9, pp. 1661–1675, 2007. 1, 2

[10] A. Ziabari, E. K. Ardestani, J. Renau, and A. Shakouri, "Fast thermal simulators for architecture level integrated circuit design," in *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2011 27th Annual IEEE*. IEEE, 2011, pp. 70–75. 1, 2, 11

[11] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *ACM SIGARCH Computer Architecture News*, vol. 31, no. 2. ACM, 2003, pp. 2–13. 2, 8, 10

[12] J. Renau, F. Basilio, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, "SESC simulator," 2005, http://sesc.sourceforge.net. 2, 10

[13] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*. IEEE, 2009, pp. 469–480. 2, 10

[14] E. K. Ardestani, A. Ziabari, A. Shakouri, and J. Renau, "Enabling power density and thermal-aware floorplanning," in *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2012 28th Annual IEEE*. IEEE, 2012, pp. 302–307. 2

[15] F. Mesa-Martínez, E. K. Ardestani, and J. Renau, "Characterizing processor thermal behavior," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 1. ACM, 2010, pp. 193–204. 2, 5

[16] A. Coskun, T. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*. IEEE, 2007, pp. 1–6. 2

[17] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Proactive temperature management in MPSoCs," in *Proceeding of the thirteenth international symposium on Low power electronics and design (ISLPED)*, 2008, pp. 165–170. 2

[18] T. Heath, A. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini, "Mercury and freon: temperature emulation and management for server systems," in *ACM SIGARCH Computer Architecture News*, vol. 34, no. 5. ACM, 2006, pp. 106–116. 2

[19] H. Hamann, J. Lacey, A. Weger, and J. Wakil, "Spatially-resolved imaging of microprocessor power (SIMP): Hotspots in microprocessors," in *Proceedings of the tenth intersociety conference on Thermal and Thermomechanical Phenomena in Electronics Systems*, 2006, pp. 125–129. 2

[20] F. Mesa-Martínez, J. Nayfach-Battilana, and J. Renau, "Power model validation through thermal measurements," vol. 35, no. 2. ACM, 2007, pp. 302–311. 2

[21] A. Nowroz, R. Cochran, and S. Reda, "Thermal monitoring of real processors: Techniques for sensor allocation and full characterization," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 56–61. 2

[22] R. Cochran, A. Nowroz, and S. Reda, "Post-silicon power characterization using thermal infrared emissions," in *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*. ACM, 2010, pp. 331–336. 2

[23] E. K. Ardestani, F. Mesa-Martínez, and J. Renau, "Cooling solutions for processor infrared thermography," in *Semiconductor Thermal Measurement and Management Symposium, 2010. SEMI-THERM 2010. 26th Annual IEEE*. IEEE, 2010, pp. 187–190. 2

[24] E. K. Ardestani, E. Ebrahimi, G. Southern, and J. Renau, "Thermal-aware Sampling in Architectural Simulation," in *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '12. New York, NY, USA: ACM, 2012, pp. 33–38. 2

[25] 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 *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–10. 3, 5

[26] S. Jarp, R. Jurga, and A. Nowak, "Perfmon2: a leap forward in performance monitoring," *J. Phys.: Conf. Ser.*, vol. 119, p. 042017, 2008. 5

[27] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *ACM SIGARCH Computer Architecture News*, vol. 28, no. 2. ACM, 2000, pp. 83–94. 8

[28] 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*. ACM, 2006, pp. 298–303. 8

[29] J. Nayfach-Battilana and J. Renau, "SOI, interconnect, package, and mainboard thermal characterization," in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*. ACM, 2009, pp. 327–330. 8, 10

[30] J. Haskins Jr and K. Skadron, "Accelerated warmup for sampled microarchitecture simulation," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 2, no. 1, pp. 78–108, 2005. 9

[31] F. Bellard, "Qemu, a fast and portable dynamic translator," in *Proceedings of the annual conference on USENIX Annual Technical Conference*. USENIX, 2005. 10

Ehsan K.Ardestani is a PhD candidate in Computer Engineering at the University of California, Santa Cruz. His research interests include computer architecture, power modeling, power and thermal-aware design, and simulation methodology for scalable multicore systems. He received his BS and MS degree in Computer Engineering and Computer Architecture from Isfahan University and Amirkabir University of Technology, Iran, respectively. He is a member of the IEEE and IEEE Computer Society.

Francisco J. Mesa-Martínez is a collaborator with the MASC group. His research interests include computer architecture, thermal modeling, power-aware thread scheduling, physical and design error tolerance, behavior based control approaches, non-Von Neumann computation, and data compression. He has a MS in Electrical Engineering from the University of Southern California, and a PhD in Computer Engineering from the University of California, Santa Cruz. He is an IEEE and ACM member.

Gabriel Southern is a member of MASC group, currently pursuing a PhD in Computer Engineering at the University of California, Santa Cruz. His research interests include low-level hardware/sofware system interaction and parallel processing. He received the B.S. degree in computer engineering from the Univeristy of Virginia, in Charlottesville, Virginia, in 2002, the M.S. degree in computer engineering from George Mason University, Fairfax, Virginia, in 2008. From 2002 to 2006 he served as an officer in the U.S. Army. He is a member of IEEE.

Elnaz Ebrahimi is a member of MASC group, currently pursuing a PhD in Computer Engineering at the University of California, Santa Cruz. Her research area is elasticity in microarchitecture design and its applications towards managing timing variability and power efficiency in processor design. She received her BS in Computer Engineering from San Jose State University, San Jose, CA in 2008. She received her MS in 2011 from the University of California, Santa Cruz.

Jose Renau is an associate professor of computer engineering at the University of California, Santa Cruz. His research interests include computer architecture, low-power and thermal-aware designs, thread-level speculation, and prototyping. Renau has a PhD in Computer Science from the University of Illinois at Urbana-Champaign. He is a member of the IEEE Computer Society and the ACM.