Power Model Validation Through Thermal Measurements

Francisco J. Mesa-Martınez Joseph Nayfach-Battilana Jose Renau
Dept. of Computer Engineering, University of California Santa Cruz
http://masc.soe.ucsc.edu

ABSTRACT

Simulation environments are an indispensable tool in the design, prototyping, performance evaluation, and analysis of computer systems. Simulator must be able to faithfully reflect the behavior of the system being analyzed. To ensure the accuracy of the simulator, it must be verified and determined to closely match empirical data. Modern processors provide enough performance counters to validate the majority of the performance models; nevertheless, the information provided is not enough to validate power and thermal models.

In order to address some of the difficulties associated with the validation of power and thermal models, this paper proposes an infrared measurement setup to capture run-time power consumption and thermal characteristics of modern chips. We use infrared cameras with high spatial resolution (10x10μm) and high frame rate (125fps) to capture thermal maps. To generate a detailed power breakdown (leakage and dynamic) for each processor floorplan unit, we employ genetic algorithms. The genetic algorithm finds a power equation for each floorplan block that produces the measured temperature for a given thermal package. The difference between the predicted power and the externally measured power consumption for an AMD Athlon analyzed in this paper has less than 1% discrepancy. As an example of applicability, we compare the obtained measurements with CACTI power models, and propose extensions to existing thermal models to increase accuracy.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement Techniques; C.1 [**Processor Architectures**]: General

General Terms

Performance and Experimentation

Keywords

Power and thermal measurements

ISCA'07, June 9–13, 2007, San Diego, California, USA.

1. INTRODUCTION

Temperature and power consumption are first order design parameters for modern high performance architectures. High operational temperatures and large power consumption present possible limits to performance and manufacturability. Due to the importance of energy and thermal factors, modern architects have extended their performance-centric processor simulation infrastructures to accommodate models for power consumption [1, 11] and thermal behavior [3, 10].

Currently, Wattch [1] and related tools are commonly used to model the dynamic power consumption in modern processors. It builds on top of CACTI [11], a very popular power model for SRAM-like structures. For the estimation of static and leakage-associated power consumption, designers employ packages such as HotLeakage [13], which builds on top of the HotSpot [10] thermal model and the Wattch [1] power model.

Each of these tools has been individually validated to a varying degree, but the corroboration of a final integrated model is not an easy process. This is mainly due to the fact that modern processors do not provide sufficient means to validate proposed power and thermal models. This limitation stems from the nature of the verification process associated with this kind of tools.

Validation of real-time processor metrics demands the measurement of real-time response from the processor itself. Designers obtain this real-time data using performance-monitoring structures – such as performance counters. Using these structures, designers can compare the real-time data collected from the processor with that predicted by the simulation environment. For example, different performance counters can provide statistics like IPC and instruction cache miss rate, or more detailed statistics like load-store queue replays. Those statistics make it possible to validate architectural simulators with existing processors.

However, this is not the case for power and thermal models. Unlike performance statistics, modern processors lack structures to gather power and thermal metrics. Adding sufficient power counters to obtain the needed level of granularity would consume a significant amount of die real estate, and impact significantly both power and processor performance.

To validate processor power models, the architecture community would like to observe the actual temperature and power behavior of proposed high performance systems. Specifically, it would be ideal if actual power consumption and temperature statistics could be gathered for each major processor floorplan block (L1 cache, register file...) and further broken down by the leakage and dynamic-related components.

Without the measurement of real-time response from the processor, the best efforts of the architecture community are reduced to best guesses and approximations when modeling the power and

thermal behavior of proposed architectural designs. Further, the cumulative impact of power and thermal approximations may have a significant effect on the resulting accuracy of the simulated systems. Therefore, many architects using integrated simulation environments do not trust the absolute results predicted for the behavior of their systems using current tools – relying instead on relative trends in thermal and power behavior.

This paper proposes a solution to several of these dilemmas by providing a method to empirically determine an accurate, real-time, breakdown of power in the of-the-shelf microprocessor. We propose and evaluate an infrastructure to directly measure temperature on any modern processor. The proposed measuring infrastructure uses infrared cameras to capture transient temperature fluctuations. We apply a genetic algorithm to find an accurate power consumption map that correlates with the measured thermal map with a minimal degree of error. The resulting power measurements are further broken down into dynamic power and leakage power. This infrastructure can be used to quantify the accuracy of power and thermal models.

The paper has the following contributions: we propose an infrastructure to measure temperatures on modern high-performance processors; develop image processing filters to increase the thermal image accuracy; design genetic algorithms to find a correspondence between temperature and power; and measure the floorplan power on a real chip. As an example of application, we compare the results with existing CACTI power reports for AMD cache structures and propose thermal modeling extensions to existing tools.

The rest of the paper is organized as follows. Section 2 describes the proposed infrastructure ; Section 3 describes the setup parameters for our evaluation; Section 4 evaluates the infrastructure proposed, and the accuracy of existing models; Section 5 covers related work; and Section 6 presents conclusions and future work.

2. INFRASTRUCTURE

This paper proposes a system capable of measuring, with a very fine degree of granularity, temperature and power consumption of modern high performance processors. The proposed infrastructure measures processor temperature using an infrared camera. Due to the fact that temperature is partially a function of power consumption in microprocessors, it is possible to build an infrastructure that translates temperature into power consumption. One of the main challenges with this approach, however, is that there is no direct translation function between temperature and power consumption. Certain factors like heat spreading further complicate the analysis. This paper explains in five major steps, the measurement setup and algorithms required for the translation to happen – converting temperature to power.

First, the proposed **measuring setup** (Section 2.1) captures the chip temperature with an infrared (IR) camera. An IR-transparent heat sink is used to allow the IR camera to obtain the die temperature of the processor. This transparent heat sink is capable of dissipating up to 100W, thus it is aptly suited for most modern high performance processors. The setup is capable of capturing up to 125fps with a $10x10\mu m$ spatial resolution, and it can be applied to multiple chips with relative simplicity.

Second, we propose a **thermal image processing** (Section 2.2) correction filter to improve measurement distortions. Modern IR cameras achieve 320x200 pixel resolution with 28mK precision per pixel. Nevertheless, cameras suffer from significant error in the order of several degrees Kelvin between pixels. The significance of the error in the measured data means that calibration for each specific lens, objective, and temperature range setup is required.

Third, we build a detailed **thermal model** (Section 2.3) of the chip being tested. In order to achieve this objective, the thermal model includes additional chip package characteristics including the newly developed infrared transparent heat sink.

Fourth, this paper aims at obtaining the power breakdown for each floorplan block on the measured processor. The **power model** (Section 2.4) used captures the dynamic and leakage power. To obtain the dynamic power breakdown, we run different benchmarks with multiple activity rates. To capture leakage breakdown, we repeat the same experiments at multiple ambient temperatures. Since leakage is power dependent, the different experiments can provide a breakdown in the three major components.

Fifth, to obtain a detailed power map from the thermal measurements, we need the inverse operation, namely **temperature to power** (Section 2.5) conversion. This is not a trivial task, because there is no direct translation (1-to-1 mapping) between our measured temperature maps and the power maps. Our setup has n-to-1 mappings because not all the boundary conditions are known and the equation being solved has to be fitted to every frame. To find the power configuration that generates a specific thermal map with the lowest error, we use a genetic algorithm that iterates multiple (SPEC2000) thermal traces and compares them against the results from our thermal simulator.

2.1 Measuring Setup

Temperature measurements are made both along the surface of the processor and within the oil coolant that flows over the top of the chip surface. To ensure data accuracy, an infrared (IR) camera is used to measure the temperature as close as possible to the transistor junction. Figure 1-(a) shows a picture with the major components of the measuring setup.

An IR transparent heat sink is needed to keep the processor within a safe temperature range. The heat sink is implemented using mineral oil (Fluka Mineral Oil 69808) that flows over the top of the silicon substrate (Figure 1-(b)). Fluka mineral oil is determined to be an appropriate coolant based upon its elevated transparency in the infrared spectrum, high specific heat, relatively high thermal conductivity, relatively low viscosity, and chemical safety. Even though water has around 2.5 times the specific heat of mineral oil, we can not use it because it is not transparent to infrared wavelengths. Fluka oil is designed for infrared spectrography and delivers excellent infrared pictures.

The chosen oil setup is selected as a balance between ease of modeling and accuracy. Turbulent oil flow can remove more heat than laminar flow, but it is significantly more difficult to model accurately. For that reason, we create a laminar flow on top of the processor core. The oil flows from the L2 cache to the processor core. We maintain a high flow rate to minimize heat transfer from the L2 cache to the processor core. Our measurements show less than .1°C oil temperature increase from side to side of the chip.

The oil temperature is continually monitored with multiple digital thermometers (Dallas DS18B20) connected to the measuring computer. The setup is capable of dissipating up to a 100W. We keep 2 liters in the oil reservoir for the system, a small radiator in the closed oil circuit is used to guarantee minimal temperature oscillations during each run.

A detailed thermal map is obtained with an infrared camera (FLIR SC-4000). Using the PC-Link (Gigabit Ethernet), the camera is set up to capture and transfer 125fps with 320x200 pixels of spatial resolution. This camera operates on the 3-5μm wavelength (MWIR) a range of light where silicon is partially transparent [1].

[1]Si has a fairly uniform 55% transmittance from 1.5μm to 6μm.

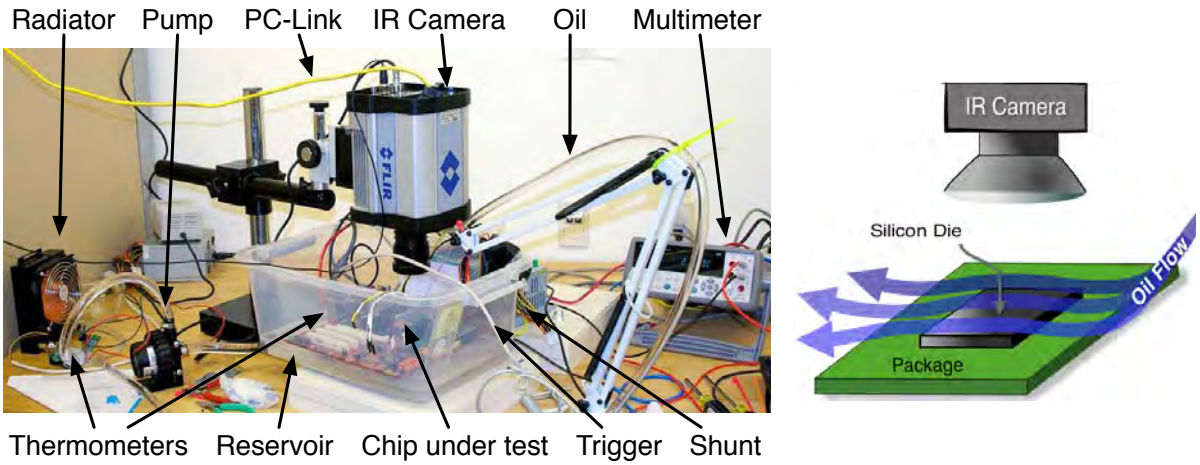

Figure 1: (a) Measuring setup; (b) Oil-based heat sink with laminar flow.

As a result, the IR camera is capable of measuring the temperature "through" the chip being tested. Modern high performance processors are manufactured using flip-chip designs [2] – exposing the silicon substrate. Since the camera can measure temperatures through the silicon substrate, using flip-chips greatly simplifies the task of measuring junction temperatures.

2.2 Thermal Image Processing

Due to their operational characteristics, infrared cameras are calibrated to compensate for different material emissivities, lens configurations, temperature ranges for the object/material measured, and a host of other factors. One approach for calibration is to have the infrared measuring device calibrated for the specific setup by the manufacturer. However, this ignores the temperature range of the object and increases the likelihood of measurements being made outside of the calibrated range. To solve this problem, we perform an in-house calibration.

Indium antimonide (InSb) sensors available on IR cameras, like the one found in the FLIR camera used in the measurement setup, have a high sensitivity per pixel (25mK). This corresponds to the camera's optimal accuracy once it is correctly calibrated. To compensate for the camera error, we perform two controlled measurements: one with cold (16°C) and one with hot (71°C) mineral oil on top of an inactive (off) processor's silicon substrate. Figures 2-(a) and 2-(b) show the IR thermal measurements when the processor is powered off. We observe that for cold oil (Figure 2-(a)) the center of the image closely resembles the measured temperature while the side pixels can have up to 6°C error (288K vs 294K). The opposite effect is shown when the camera measures a uniformly hot mineral oil (345K vs 335K).

The camera specifications indicate that a linear ($\texttt{"Temp"} = A * \texttt{"IR Temp"} + B$) correction should be applied for each camera pixel. Our image filter automatically generates a linear correction factor to compensate for the inaccuracies. A secondary filter is used to compensate for the optical distortion induced by the lens setup.

[2]Low power chips tend to be wire-bond, while more high-performance chips tend to be flip-chip.

2.3 Thermal Model

The thermal model used in this paper requires a high degree of accuracy and must correctly reflect the effects of the liquid cooling setup. To do this, we extend the functionality of HotSpot [10] – currently the most popular tool used by the computer architecture community – to perform thermal modeling. The base model in Hotspot is extended it in two very significant ways. First, we expand the model to capture the effects of silicon-on-insulator fabrication technology, plus chip interconnect and packaging. Second, the metal heat sink is replaced by a laminar oil flow.

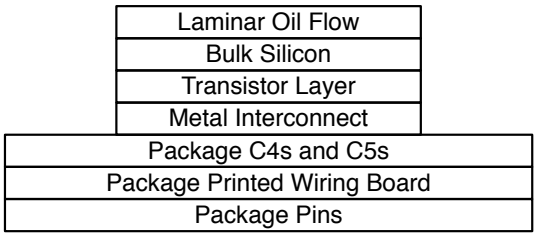

Figure 3: Thermal model layers.

To model the effect of the various chip layers and packaging, we introduce a new layered model (Figure 3); The transistor layer models the thermal effect of the silicon-on-insulator technology used for the processor in the setup analyzed in this paper. Since the package pins and the C4s have a significant specific heat, we add them to the model to better capture transients.

To model the liquid cooling, we estimate the local convection coefficient of the oil flowing over the chip. This is done by assuming laminar flow across the chip in Equation 1. Where, k is the thermal conductivity of oil, Pr is the Pradtl Number of oil, Re is Reynold's Number of oil, and L is the chip length.

$$h_l = \frac{2 * k}{L}(0.332)Re^{1/2}Pr^{1/3} \quad (1)$$

The Reynold's number is given by Equation 2 where V is the upstream liquid velocity and ν is the oil viscosity.

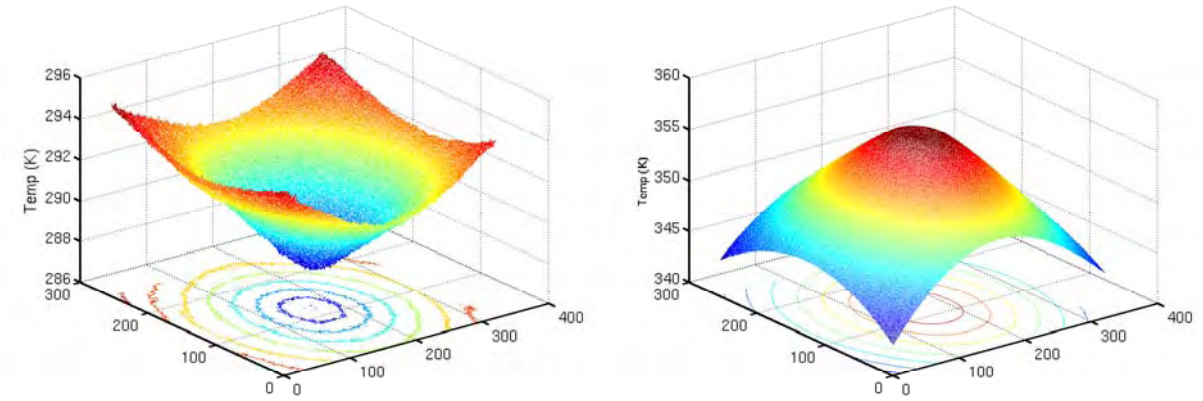

Figure 2: IR Measured temperature with low (Left), and high temperature (Right) behavior for IR camera. (Temperature scale in Kelvin)

$$Re = \frac{V * L}{\nu} \quad (2)$$

To compute the convection coefficient demands that the upstream velocity be known. To find the upstream velocity, we assume a two-dimensional setup, where the oil flows from a jet with specified flow-rate. Using conservation of energy, momentum, and flux, the upstream velocity is determined.

2.4 Power Model

We break down power consumption into two components: leakage and dynamic power. While dynamic power is only a function of activity, leakage power has many factors. A BSIM3 [2] model provides the leakage for each gate. To compute the exact leakage power for a full block, transistor stacking and other factors need to be considered. A full processor floorplan block could be approximated as follows:

$$Leakage = P_0 * \nu_t^2 * e^{\frac{V_{GS}-V_{th}-V_{off}}{n*\nu_T}} (1 - e^{\frac{-V_{DS}}{\nu_T}}) \quad (3)$$

In Equation 3, P_0 is a parameter of the floorplan block, V_{DS} is the voltage difference between drain and source, V_{GS} is the voltage difference between gate and source, and V_{off} is the offset voltage for a minimal region, n is the sub-threshold swing coefficient, and ν_T is the temperature dependent thermal voltage, at temperature T ($\nu_T = \frac{kT}{q}$). All three voltages (V_{DS}, V_{GS}, and V_{off}) are independent of temperature, the only temperature dependent factors in the leakage equation are V_{th} and ν_T. All the other factors are thermally independent. As a result, we could approximate the leakage per floorplan area as $P_{leak0} * T^2 * e^{\frac{P_{leak1}}{T}} * (1 - e^{\frac{P_{leak2}}{T}})$.

Equation 4 shows the expected power consumption per floorplan block and frame pair. P_{dyn} is the dynamic power consumption. To compute the leakage factor, three parameters are required: P_{leak0}, P_{leak1}, and P_{leak2}. Although possible in some systems, our model does not share any leakage parameter across blocks because each floorplan block may have a different threshold voltage and/or supply voltage. While leakage parameters stay constant for all the frames, P_{dyn} is dependent of the processor activity rate, as a result, each program phase has a unique dynamic power.

$$Power = P_{dyn} + P_{leak0} * T^2 * e^{(P_{leak1}/T)} * (1 - e^{P_{leak2}/T}) \quad (4)$$

2.5 Temperature to Power

There has been substantial work done on models for the power to temperature translation problem, the modeling of the reverse problem of converting temperature information into power for modern processors has been explored at a much smaller scale [6]. To our knowledge, no previous work provides a model to perform the translation of transient temperature measurements into transient power consumption for a modern processor. Unlike previous proposed work that rely on simpler steady-state thermal models, transient thermal models may have a n-to-1 power-to-temperature mapping (a temperature map can be generated from multiple power maps) for systems where not all the boundary conditions are known. Some boundary conditions can not be determined because the thermal setup only obtains the silicon substrate/junction temperature. This is, it can not measure the temperature for the pins, package, and other layers modeled. In order to solve this optimization problem, we use a genetic algorithm (GA) to optimize our temperature to power model.

Genetic algorithms are very effective tools for optimization problems. Although they can not claim to find the best solution, GAs can tolerate significant measurement error. In our proposed system, the GA finds the power equation (Equation 4) parameters: P_{leak0}, P_{leak1}, and P_{leak2} for each floorplan block, and the P_{dyn} for each floorplan block and program phase pair. This is achieved after gathering results for several benchmarks at different ambient temperatures (T) and processor activities (AR).

Genetic algorithms start using a set of hand crafted candidate **individuals** or solutions. For each generation during the algorithm's execution, its modus operandi is as follows: **a fitness value** is calculated for each individual so that the fittest members of the population have a higher chance of mating, and individual's fitness is usually assumed to be directly proportional to a defined error model. After members are paired, children are produced by combining (crossing-over) the parents genetic material. A randomly selected set of the fittest children is selected for survival. To avoid getting stuck on local minima, suboptimal individuals make it into the next generation, as well as some of the fittest parents from the previous generation. Members of the population are possible candidates for **mutation** in elements of their genetic information. In our case, the characteristics for legal mutations are predefined, this ensures that the search space is bounded. In addition, we **early kill** combinations/mutations that produce aberrations due to mutation

or cross-over. After the overall fitness of the population has ceased to improve for more than a predetermined number of generations, the search stops and the fittest member of the final generation is assumed to be the optimal parameter configuration for our thermal to power translation model.

The previous paragraph describes a generic genetic algorithm. To further understand the details of the GA used on this paper, we describe the structure of an individual (solution), the fitness value, the mutate process, and the early kill process as follows.

Genes specify a possible solution for a problem. In this case, the power equation (Equation 4) is the problem to be solved. We attempt to select the combination of P_{leak0}, P_{leak1}, P_{leak2}, and P_{dyn} values that when input into the power equation (Equation 4) and applied to each temperature frame, minimizes the cumulative error across all the frames between the reported temperature in the thermal model and empirical data from the IR camera. To accomplish this, parameters are assigned to genes that are allowed to evolve. In this experiment, each floorplan block has a unique power equation, where P_{leak0}, P_{leak1}, and P_{leak2} are shared across all the frames; therefore, each of these parameters have a unique gen. Further, there is a unique P_{dyn} for each program phase and floorplan block pair, where each program phase has a unique processor activity. To reduce the mutation search space, certain restrictions are placed upon the range of possible mutations. Since program phases have nearly the same activity regardless of temperature, P_{dyn} is shared for all cases where a given application is rerun at a new temperature. To further reduce the mutation search space, we restrict the P_{leak1} and P_{leak2} values. Since both parameters are technology dependent, they will be constant across the entire chip. To reduce the search space, P_{leak1} and P_{leak2} are shared for the whole chip, and each floorplan block can adjust the shared value within a limited range.

Fitness Value provides a numeric error for each individual. For each frame fed to the GA, the power equation has two fitness functions: one for power and a second one for temperature. The power fitness finds the average and standard deviation power consumption discrepancies between the power equation and the multimeter measurements. The temperature fitness finds the average and standard deviation between the junction temperature reported by the thermal model and the infrared camera measured temperature. We keep the standard deviation in addition to the average error because between two individuals with similar average error, we give a higher fitness score to the individual with the lowest standard deviation.

Mutation randomly changes the value of a set of genes. Each value from the power equation is expressed as a real value. To reduce the search space, the mutation only can deliver reasonable values. E.g.: P_{dyn} and P_{leak0} only have positive values.

Early Kill removes clearly wrong individuals from the pool. The fitness function will remove bad individuals too, we do an early kill to avoid the costly thermal fitness computation. Floorplan block genes with 10 times more power density than the average power density are early killed. Also, individuals with more than 75% error on the power fitness function are also killed.

3. EVALUATION SETUP

While the measuring setup section (Section 2.1) explains the infrastructure required to measure any modern processor, this section explains the measured configuration parameters. Table 1 summarizes the main processor and thermal parameters.

Figure 4 shows floorplan blocks used on the evaluation. L1I and L1D stand for instruction and data cache respectively. The IRF and the FRF stand for integer and floating point register file respectively. While the Clock Distribution block correspond to the clock distribution circuitry, the Clock block is clock-generation related. Both blocks (Clock and Distribution) are fairly independent of processor activity. The detailed information for larger blocks like the Fetch block are not freely available, so we cluster the fetch (branch predictors and pre-decode logic) into three blocks. The infrared camera captures 320x200 pixels resolution but the genetic algorithm only computes the power for each floorplan block. As a result, we average the temperature for each block before passing this information to the genetic algorithm. It is important to note that the floorplan is a mirrored version of the AMD floorplan commonly published. The reason is that we are measuring "through" the substrate, which flips the image.

Parameter	Value	Parameter	Value
CPU	AMD/Athlon 64	Package	754
CPU Model	AMN2800BIX5AR	Vdd	1.4v
Technology	130nm SOI	Frequency	1.6GHz

Table 1: Main processor and thermal model parameters.

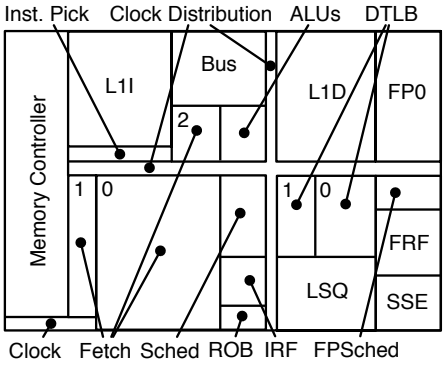

Figure 4: AMD Athlon 64 core floorplan blocks.

3.1 Thermal Modeling

For the thermal simulator, we use the package shown in Figure 3. The interconnect layer models the compounded thermal characteristics of all the metal layers, assuming an interconnect layer stack-up that closely matches an IBM 130nm process. The thermal effect of the interconnect metal density is captured on a per floorplan block basis.

The remaining layers model the thermal effects of the chip package. Using the AMD 754-pin non-lidded micro PGA, we model the effects of the package printed wiring board (PWB), its substrate and pins. While the heat sink is the primary heat flow path, the pins add a secondary flow path through the processor.

3.2 Applications

For evaluation purposes, we gather thermal and power statistics for the first 20 seconds of several benchmark applications measured when running under Linux on our test system. When the applications are launched, the serial port triggers the multimeter for power data capture. As a result, we have the measurements synchronized with the application initialization.

To have a diverse set of instruction streams we execute 14 different applications. We execute the majority of SPEC 2000 benchmarks (ammp, applu, apsi, bzip2, crafty, equake, gap, gzip, mcf, mesa, mgrid, parser, twolf) and a matrix kernel. This kernel per-

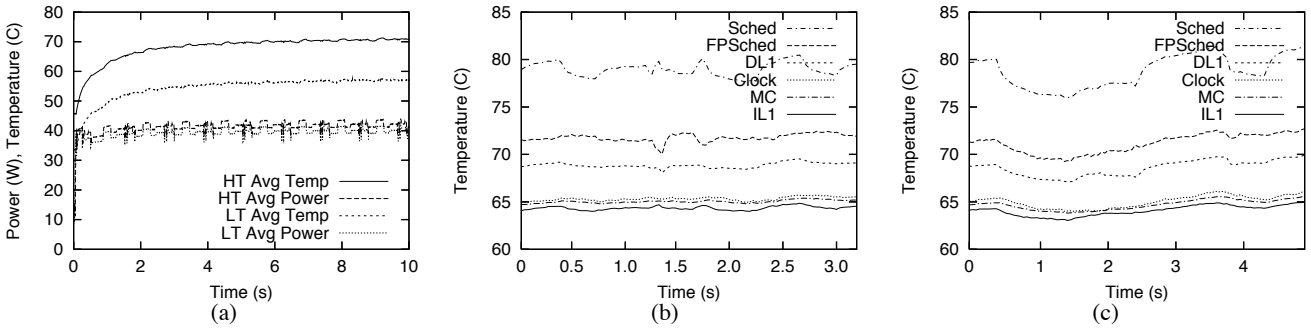

Figure 5: Thermal and power measurements for the first 10 seconds of applu (a) ; temperature profile for the integer scheduler (Sched), data cache (DL1), floating point scheduler (FPSched), clock generation (Clock), memory controller (MC), and instruction cache (IL1) for applu (b) and apsi (c).

forms a matrix multiplications at 1Hz frequency (0.5 seconds matrix multiplication, 0.5 second idle).

4. EVALUATION

The main objective of the paper is to propose an infrastructure to capture temperatures with high resolution and obtain detailed power maps. The evaluation focuses on the accuracy of the proposed measuring setup and highlights potential applications. This section begins by presenting the main results. Then, it proceeds to show the behavior of the image processing algorithms. We continue with a list of recommendations for thermal models, and we finish by comparing the power results with CACTI 4.2 [11].

4.1 Main Results

In order for the genetic algorithm to find a correct power consumption breakdown, the processor executes the same set of applications under two different sets of ambient (oil) temperatures. Figure 5-(a) shows the temperature and power profile for the first 10 seconds of execution (1250 frames) of applu from SpecFP2000. The solid lines correspond to a run where the oil was heated to 33.5°C, the dashed lines correspond to a run with a 22°C oil temperature. Even though the oil temperature is just 11.5°C, the compounded thermal difference is slightly higher (13°C), which is the expected behavior. The overall power consumption is higher on the high temperature (HT) run than on the low temperature (LT) run. This is due to the fact that as we increased the oil temperature, the leakage power also increased. The plot shows that on average for a 13°C increase, total power consumption increases by 5.3%.

Figures 5-(b) and 5-(c) show the average temperature for several floorplan blocks for the respective execution of applu and apsi. The floorplan blocks, ordered from highest to lowest average temperature, are the integer scheduler (Sched), data cache (DL1), floating point unit scheduler (FPSched), clock generator (Clock), memory controller (MC), and instruction cache (IL1). For most of the applications, the scheduler is close to 10°C hotter than any other floorplan block. For both applications, the temperature across blocks is somewhat correlated. However, after a 1.5 second execution for applu (Figure 5-(b)) and 4 second execution for apsi (Figure 5-(c)), the integer scheduler decreases temperature while the floating point scheduler temperature increases. These phase changes correspond to periods of time when the application increases the percentage of floating point instructions being executed. The genetic algorithm correctly finds a phase where the access to the power consumption on the integer scheduler decrease while the power consumption on floating point scheduler increase.

The genetic algorithm (Section 2.5) uses the temperature and power traces to find the power equation (Equation 4) parameters for each floorplan block. Table 2 shows the maximum, minimum, and average total power, dynamic power, and leakage power for each floorplan block.

For the running applications, the minimum total power consumption (6.09W) (2nd column) is lower than the minimum total power consumption measured by the multimeter when the processor is idle at low oil temperature [3] (8.56W). Similarly, the maximum total power consumption (93.58W) (3rd column) is higher than the maximum power consumption measured (47W). However, this discrepancy is due to the fact that floorplan blocks never exhibit minimum and maximum activity simultaneously. In the event that this were to happen, the multimeter empirical and GA value would more closely match. We do not observe such behavior for the SPEC applications. On average, the AMD processor consumes 31.92W total power (4th column).

The minimum dynamic power (5th column) is the smallest value found by the GA across all the frames. Due to the fact that several frames include an inactive/idle processor activity, the minimum dynamic power consumption corresponds to the standby or idle dynamic power. The highest maximum dynamic power consumption block is the Fetch 0 block (12W) closely followed by the L2 cache block (11.87W). The fetch block includes the branch prediction and the instruction decode from x86 to microcode. The next highest blocks are the level one caches (L1D and L1I), the memory controller, and the scheduler.

The maximum dynamic power (6th column) reports the maximum dynamic power found for any frame, while the average dynamic power (7th column) takes into consideration the average floorplan block activity found on the SPEC applications executed. A very interesting case is the memory controller where the maximum dynamic power is 4.98W, while the average dynamic power is just 0.43W. The reason is that requests to the memory are infrequent but very costly. A similar conclusion can be drawn for the L2 cache. Other blocks like the integer scheduler (Sched) or the integer register file (IRF) have a smaller difference between maximum and average because these structures are more frequently utilized.

The difference between the minimum leakage power (8th column) and the maximum leakage power (9th column) is not as substantial as the difference found on the dynamic power. Due to the fact that the differential in leakage power is due to temperature changes. For a given floorplan block the same frame that has minimum dynamic power may not be the same frame that has min-

[3]Lower oil temperature decreases leakage power

Block	Total Power			Dynamic Power			Leakage Power			Leakage Constants		
	min	max	avg	min	max	avg	min	max	avg	$10^6 x P_{leak0}$	P_{leak1}	P_{leak2}
Memory Controller	0.18	5.26	0.64	0.03	4.98	0.43	0.12	0.33	0.22	5459	-2361	-232
L2	2.69	15.59	6.36	0.26	11.87	2.88	2.39	4.67	3.48	20771	-1869	-276
L1D	0.79	9.12	2.39	0.02	7.96	1.23	0.68	1.67	1.17	6291	-1881	-249
L1I	0.26	8.23	0.76	0.02	7.71	0.39	0.21	0.55	0.37	4117	-2178	-365
Inst. Pick	0.02	1.43	0.46	0.01	1.39	0.44	0.01	0.04	0.02	2878	-3012	-319
DTLB 1	0.03	2.70	1.71	0.01	2.64	1.67	0.01	0.06	0.03	1906	-2773	-319
DTLB 0	0.04	3.07	0.85	0.01	2.99	0.78	0.03	0.12	0.07	8095	-3012	-349
LSQ	0.06	3.07	0.70	0.01	2.94	0.62	0.03	0.14	0.08	2720	-2544	-271
Fetch 0	0.04	12.03	6.93	0.03	12.00	6.91	0.01	0.03	0.02	1961	-3012	-291
Fetch 1	0.08	3.15	0.92	0.01	3.01	0.81	0.06	0.16	0.11	397	-1807	-275
Fetch 2	0.01	2.04	0.34	0.01	2.03	0.34	0.00	0.01	0.00	417	-2964	-322
Sched	0.06	3.25	2.25	0.01	3.08	2.15	0.04	0.18	0.10	6296	-2795	-267
IRF	0.30	2.68	1.77	0.01	1.85	1.24	0.25	0.86	0.53	3778	-2082	-299
ROB	0.02	1.02	0.43	0.01	0.99	0.41	0.01	0.03	0.02	2015	-3012	-318
ALUs	0.17	2.03	0.68	0.01	1.70	0.38	0.14	0.47	0.29	4281	-2324	-371
Bus	0.33	4.76	0.89	0.01	4.29	0.43	0.28	0.67	0.46	2105	-1807	-232
FPSched	0.33	2.65	0.83	0.01	1.81	0.23	0.32	0.87	0.60	3778	-1982	-305
FRF	0.06	2.93	0.33	0.01	2.83	0.23	0.05	0.15	0.10	4046	-2538	-232
SSE	0.06	1.22	0.27	0.01	1.11	0.18	0.05	0.14	0.09	402	-1844	-302
FP0	0.17	1.97	0.47	0.01	1.71	0.21	0.15	0.36	0.26	1278	-1909	-373
Clock	0.15	1.36	0.54	0.01	1.07	0.32	0.13	0.32	0.22	1304	-1943	-297
Clock Distrib.	0.24	4.01	1.40	0.03	3.49	1.03	0.18	0.58	0.37	3522	-2632	-288
	6.09	93.58	31.92	0.57	83.46	23.31	5.14	12.40	8.61	3953	-2398	-297

Table 2: Power values obtained from the genetic algorithm.

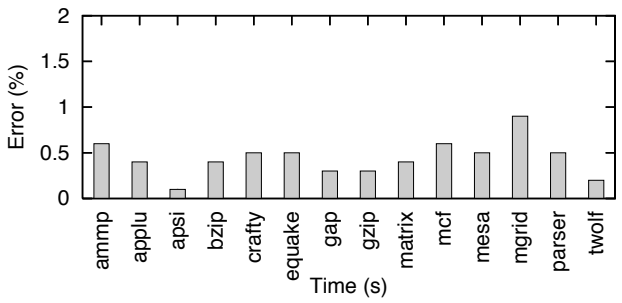

Figure 6: Power prediction error per application.

imum leakage power. Thus, Adding the minimum leakage and dynamic power does not correspond to the minimum total power. The same observation holds true for maximum power. Leakage represents 27% of the average power consumed by the applications in the evaluation.

To measure the accuracy of the predicted model, Figure 6 presents the overall error in power consumption per application. For all the benchmarks, the error is always lower than 1%. The error is defined to be the difference between the power found by the genetic algorithm and the power measured by the multimeter. The multimeter measures the current going from the power supply into the processor, however the voltage regulator module (VRM) between the power supply and the processor dissipates over 10% of the power during the voltage conversion. The efficiency rating for the voltage regulator is taken into account in order to correct the power measured by the multimeter.

A more detailed view of the source of errors is presented in Figure 7-(a), which plots the power discrepancy between the output generated by the genetic algorithm and the measurements from the multimeter. This plot summarizes the accuracy for the power prediction by joining all the application frames with distinct activity rates together. The frames in the plot are ordered from low to high power consumption. We observe that the genetic algorithm is able to find activity rates and technology parameters to fit the power values. The highest discrepancy has a 75% error. For this frame, the total power for low and high oil temperatures measured by the multimeter have a significant discrepancy (error of 14W vs 30W). This is evidence that the data acquisition process is not perfect. Dropped frames from the camera, while infrequent, may generate erroneous data. We feel that errors induced when capturing the frame, or changes in activity rate due to the temperature difference are the most likely causes for the discrepancies.

Figure 7-(b) also joins all the distinct frames into a single plot ordered from lower to higher power consumption. The plot shows the power breakdown between dynamic and leakage for each frame. Leakage dominates on the frames with the lowest power consumption because the activity rate is very low on these frames. As expected, leakage increases as dynamic power increases. The reason is that a higher power consumption increases the temperature which also increases leakage power.

4.2 Thermal Imaging

This section shows the raw thermal images and provides additional insights on the image processing performed on this paper.

As section 2.2 states, the IR camera does not have the same accuracy over all the pixels. To compensate for this error, we perform a different linear correction for each pixel. Figure 8-(a) shows the corrected thermal image for a single frame, including a floorplan overlay. A secondary correction filter also accounts for the registration of the camera against the plane of the processor.

The overlay on Figure 8-(a) does not cover the whole picture. The upper part of the figure shows part of the L2 cache. The picture seems to indicate that the pixels outside the die visible on the left and lower part are as hot as the die itself. The measurements on these areas have two artifacts. First, the emissivity is different outside the die area. Second, the fluid has turbulence outside the die. This turbulence creates fluctuations in the thermal measurements. As a result, measurements outside the die area are not considered accurate in our setup. Thus, they are ignored by the genetic algorithm because they are outside the processor's core block.

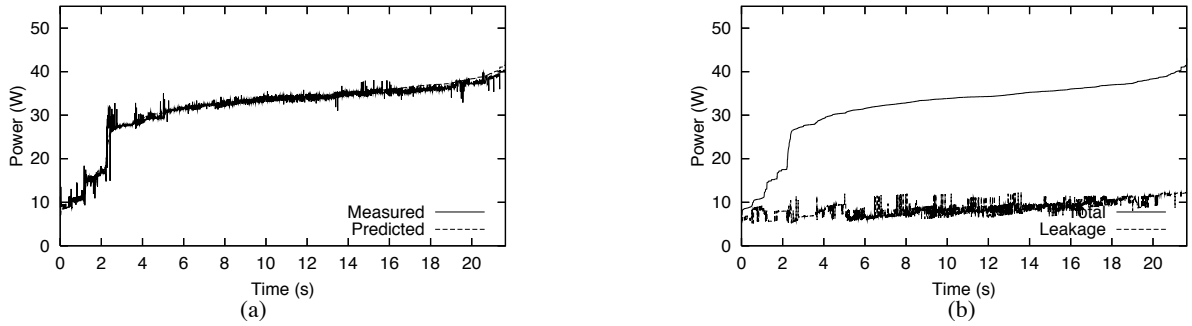

Figure 7: Power prediction accuracy summary for all the applications (a); power breakdown summary for all the applications (b).

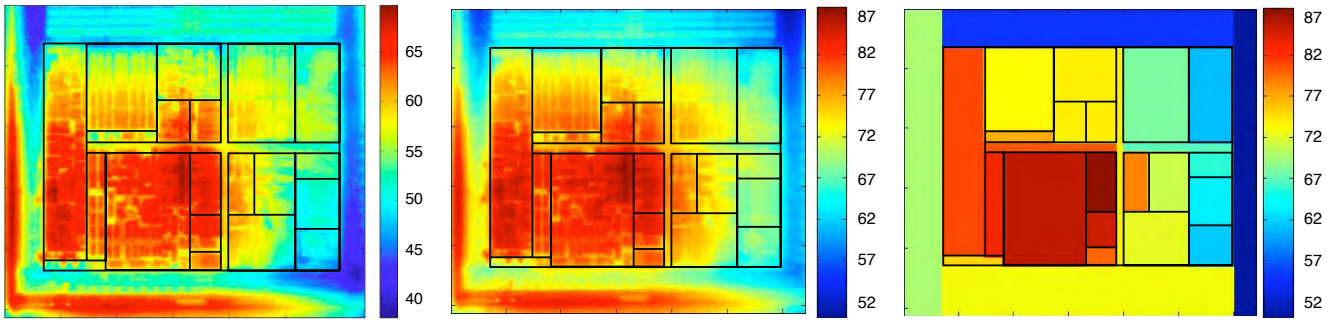

Figure 8: Temperature with overlapped floorplan (a); Hottest captured frame (b); and its average temperature per block (c).

Figures 8-(a) and 8-(b) show that there is temperature variability inside the floorplan blocks. It is this variability that prompted an extension to our thermal model so that each floorplan block could be modeled with fine granularity. Therefore, even though we report the average temperature for each floorplan block, our thermal model internally computes multiple temperature points for each block as explained on Section 2.3.

Figures 8-(b) and Figure 8-(c) show the frame from crafty with the maximum temperature measured. On this frame the register file reaches 84°C. Although the thermal model has a finer granularity, the genetic algorithm uses the average temperature per floorplan block to reduce the computing requirements as shown on Figure 8-(c).

4.3 Thermal Modeling Extensions

A possible use of the proposed thermal infrastructure is to provide a detailed thermal evaluation of existing models. To do so, a performance model and power model must be created which can be coupled with a thermal model. However, we are unable to perform this type of verification because our simulation infrastructure is unable to perform adequate performance modeling of the AMD Athlon processor used through the thermal measurements. Nevertheless, we are able to observe the thermal behavior of the AMD processor and suggest extensions for existing thermal simulation infrastructures.

Figure 9-(a) shows the thermal delta map [4] when a matrix multiply is executed. In this image, the activity is centered on the integer scheduler. As the scheduler gets warmer, the heat is propagated through the silicon substrate. The clock distribution network also gets warmer. Since the clock distribution is fairly independent on processor activity [5], a major source for heat in the clock distribution is heat generated from the scheduler. The heat propagates faster through the clock distribution network because it has a higher metal density. As a result, we feel that higher accuracy thermal models not only need to model metal layers but they also need to model different transistor/metal densities for each floorplan block.

The architecture research community integrates thermal models like Hotspot with architectural simulators. In most evaluations, researchers model less than one billion instructions. For several applications this corresponds to 1 second or less. Figure 9-(b) and Figure 9-(c) show that this can problematic. Both figures show the measurement results for the first 10 seconds of execution (1250 frames at 125fps). The figures include the average temperature for the whole core, the temperature for the register file, and the temperature for the data cache. The plots also include the power consumption for the whole CPU. Figure 9-(b) shows that the temperature decreases just 3°C (from 77°C to 74°C) when the power consumption decreases from 45W to 40W for 0.5 seconds. This means that substantial simulation time (several billions of instructions) is required to provide any interesting temperature oscillation like the one observed on wupwise. Both wupwise and apsi require over 2 seconds to warm up the chip.

The implication of Figure 9-(b) and Figure 9-(c) measurements is that to have realistic thermal simulations architects should model several seconds of simulated execution. It is important to consider

[4] A delta map is the temperature difference between the current and the previous infrared frame.

[5] ACPI power saving techniques are deactivated.

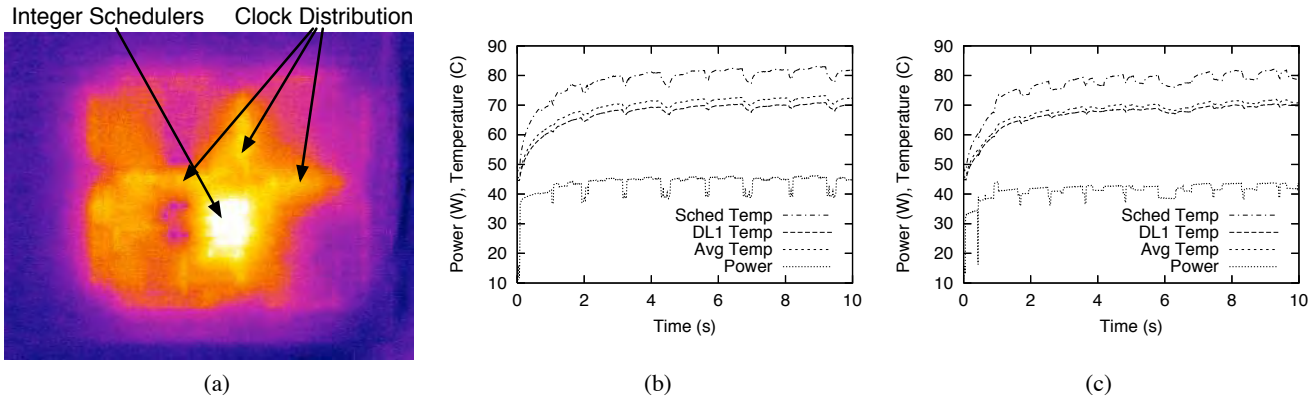

Figure 9: Measured thermal delta map when starting matrix multiply (a); Thermal and power measurements for the first 10 seconds of apsi (125fps) (b) and wupwise (c).

that our simulations use a liquid cooling instead of a more traditional heat sink. A traditional heat sink configuration may require additional simulation time. The reason is that metal heat sink has a higher potential for heat storage. In our configuration, the coolant is kept at a nearly constant temperature and little heat is transfered back to the silicon substrate.

4.4 CACTI Comparison

HotLeakage [13] and CACTI [11] are the two most popular tools used by architectural simulators to model leakage and dynamic power respectively. In this section, we compare our measured results from the AMD processor with results obtained from HotLeakage and CACTI.

We use the latest version of CACTI available (4.2) to generate the power for the AMD Athlon data cache. The latest CACTI has several improvements and also reports leakage power. The Athlon data cache is an 8 bank 64KByte 2 way set associative cache with 64byte cache lines. The L1D can provide two load and/or stores per cycle as long as there is no bank conflict. Once the frequency is adjusted to the frequency of the AMD processor (1.6GHz) and the voltage (1.4V), CACTI reports a 1.5W maximum dynamic read power. The maximum power consumption for the data cache (L1D from Table 2) is 7.96W. This corresponds to the frame with the maximum activity. From the thousands of frames evaluated, we assume that one frame had the maximum L1 data cache that the Athlon can sustain. Since the Athlon can perform a load and a store to the data cache every cycle, the power per access for the data cache is 3.48W. This means that there is a 2.3 times variation. We believe that this discrepancy is primarily due to differences in technology parameters and a custom AMD cache design that can handle two operations per cycle without being dual ported.

As previously said, CACTI 4.2 also reports the leakage consumption. For the analyzed data cache, CACTI reports 0.144W at 100°C. To compute the leakage at 100°C (373°Kelvin), we fill the leakage equation with the data from Table 2 ($6291x10^{-6} * 373^2 * e^{(\frac{-1881}{373})} * (1 - e^{(\frac{-249}{373})}) = 2.75W$). The leakage estimation by the genetic algorithm is approximately 19 times higher than the leakage estimated by CACTI. Due to the big difference in leakage modeling, it may be interesting to measure different processor caches to gain further insight.

The leakage component from equation 4 is derived from a BSIM3 equation. This is the same type of equation used by tools like HotLekage and CACTI 4.2. To better understand the discrepancy between CACTI and our measurements, we compute the average P_{leak1} and P_{leak2} used by CACTI for a 130nm. Since CACTI models several transistor types, we compute the average transistor for a cache like the Athlon cache. The results are $P_{leak1} = -455$ and $P_{leak2} = -15136$. These values are different to the average values that the genetic algorithm finds $P_{leak1} = -2398$ and $P_{leak2} = -297$ (Table 2).

Our proposed model finds different constants for each floorplan block. HotLeakage and CACTI keep the same technology parameters for all the processor floorplan blocks. Since modern processors have multiple voltages and threshold domains, adding multiple threshold voltages to the existing tools may be a feasible task. Analyzing several processors and technologies with our proposed infrastructure can find "typical" changes between floorplan blocks. These changes can be incorporated to existing models to improve accuracy.

5. RELATED WORK

HotSpot [10] is the most popular thermal model used by the computer architecture community. We build on top of it to extend the thermal models. In this work, we do not propose a new thermal model – just extensions on HotSpot model to capture additional characteristics found in our measurements.

Real power consumption measurements are a very useful tool. The original work by Isci et al [7] and later extended by Wu et al [12] measure the overall power consumption with a multimeter. Together with the activity rate captured from the processor performance counters, they provide the total power breakdown for each processor floorplan area. Our measuring setup builds on top of them as we also gather overall power consumption. The key differences is that our setup provides a detailed power breakdown and therefore provides dynamic and leakage power for each floorplan block.

Chung et al [4] build on top of models that use performance counters [7] to generate detailed thermal map. Their work compares a less compute intensive regression model against a HotSpot thermal map result.

Translating from temperature to power is known as an inverse heat transfer problem. Due to the potential uses, there is a significant effort by the research community to provide different models [8]. The most related work is done by Hamann et al [6]. This

work measures the temperature on a chip with a infrared camera. Their setup is similar to ours but they do not provide enough details on the materials/components used. Nevertheless, the key difference is that their setup only performs conversions from temperature to power for steady-state thermal maps. They do not perform temperature to power conversion for transient thermal maps. In addition, their power model does not provide a breakdown of power consumption by dynamic and leakage power.

Hamann setup only performs conversions from temperature to power for steady-state thermal maps. They do not perform temperature to power conversion for transient thermal maps. Steady-state can be viewed as a special transient case. While the steady-state has a 1-to-1 mapping (a temperature map has a unique power map), the transient has a n-to-1 mapping (a temperature map can be generated from multiple power maps). There are two fundamental reasons for the n-to-1 mapping: First, not all the boundary conditions are known (condition needed to have a 1-to-1 mapping) because the infrared camera only can measure the temperature for the transistor/bulk layer. It can not measure the temperature for pins, package, and other chip layers. As a result, the boundary conditions are not known. Second, the equation to solve (power equation) is being adapted for all the frames. If the equation to solve changes (a dynamic system), the solution also can have a n-to-1 mapping.

Genetic algorithms [5] have multiple applications for optimization problems. Raudensky [9] showed that GAs can perform the conversion from temperature for power for uni-dimensional topologies. This work extends the model to modern three dimensional processors.

6. CONCLUSIONS

This paper provides a method to obtain a breakdown of the power dissipation of modern processors into its leakage and dynamic components based on thermal measurements. The data obtained from this type of work can greatly benefit the computer architecture community.

As the evaluation shows, the measurement setup can help to improve energy and thermal models. After observing the temperature profile from a working processor, we suggested several improvements to existing simulation infrastructures. The results show situations where the heat propagates faster through floorplan blocks with higher metal density. This seem to imply that detailed simulations should include metal density. More important for the architecture community is the simulation time. Our measurements imply that architects should execute several seconds of simulated time in order to have interesting thermal oscillations.

In addition, the paper provides power consumption breakdowns from a real processor (AMD Athlon 64). The proposed measuring setup (Section 2) details all the steps, tools, and processes required to perform additional measurements on additional processors. The evaluation shows that the difference between the power consumption estimated and that the one predicted by the genetic algorithm is less than 1% on average. We feel that measuring temperature directly from processors and estimating the power consumption has a great potential for the computer architecture community.

Acknowledgments

We like to thank the reviewers for their feedback on the paper and Olivier Temam for his insights. Special thanks to James Christofferson, Ali Shakouri, Michael Huang, Luigi Capodieci, and Matthew Guthaus for their feedback on the thermal measurement infrastructure. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was supported in part by the National Science Foundation under grants 0546819; Special Research Grant from the University of California, Santa Cruz; and gifts from SUN.

7. REFERENCES

[1] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a Framework for Architectural-Level Power Analysis and Optimizations. In *International Symposium on Computer Architecture*, pages 83–94, Jun 2000.

[2] Y. Cheng and C. Hu. *MOSFET Modeling and Bsim3 User's Guide*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[3] Y.K. Cheng, P. Raha, C.C. Teng, E. Rosenbaum, and S.M. Kang. ILLIADS-T: An Electrothermal Timing Simulator for Temperature-Sensitive Reliability Diagnosis of CMOS VLSI Chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(8):1434–1445, Aug 1998.

[4] S.-W. Chung and K. Skadron. Using On-Chip Event Counters For High-Resolution, Real-Time Temperature Measurement. In *Thermal and Thermomechanical Phenomena in Electronics Systems, 2006*, pages 114–120. IEEE Computer Society, May 2006.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[6] H.F. 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, 2006*, pages 121–125. IEEE Computer Society, May 2006.

[7] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2003.

[8] M. N. Ozisik. *Inverse Heat Transfer*. Taylor and Francis, 2000.

[9] M. Raudensky, K.A. Woodbury, J. Kral, and T. Brezina. Genetic Algorithm in Solution of Inverse Heat Conduction Problems. pages 293–306, 1995.

[10] 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*, pages 2–13, Jun 2003.

[11] S. Wilton and N. Jouppi. CACTI: An Enhanced Cache Access and Cycle Time Model. *IEEE Journal on Solid-State Circuits*, 31(5):677–688, May 1996.

[12] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan. A systematic method for functional unit power estimation in microprocessors. In *DAC '06: Proceedings of the 43nd annual conference on Design automation*, New York, NY, USA, 2006. ACM Press.

[13] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. Technical Report CS-2003-05, Univ. of Virginia Dept. of Computer Science, March 2003.