Measuring Performance, Power, and Temperature from Real Processors

Francisco J. Mesa-Martinez Michael Brown Joseph Nayfach-Battilana Jose Renau

Dept. of Computer Engineering, University of California Santa Cruz

http://masc.soe.ucsc.edu

ABSTRACT

The modeling of power and thermal behavior of processors requires challenging validation processes, which may be complex and undependable. In order to ameliorate some of the difficulties associated with the validation of power and thermal models, this paper describes an infrared measurement setup that simultaneously captures run-time power consumption, thermal characteristics, and performance activity counters from modern processors. We use infrared cameras with high spatial resolution (10x10μm) and high frame rate (125Hz) to capture thermal maps. Power measurements are obtained with a multimeter, while performance counters are obtained after modifying the operating system (Linux), both at a sampling rate of $1KHz$. The synchronized traces can then be used in the validation process of possible thermal, power, and processor activity models.

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

1. INTRODUCTION

Temperature and power consumption have become first order design parameters for most modern, high performance architectures. Elevated operational temperature and power consumption present possible limits to performance and manufacturability. Due to the importance of this data, modern architects have extended their performance-centric processor simulation infrastructures to accommodate models of power consumption [2, 12] and thermal behavior [4, 11].

Wattch [2] and several Wattch-like tools are used routinely in the modeling of dynamic power consumption in modern processors. Wattch builds on top of CACTI [12], a very popular power model for SRAM-like structures. For the modeling of static and leakage-associated power consumption, designers employ packages such as HotLeakage [13], which builds on top of the HotSpot [11] thermal model and the Wattch [2] power model.

Each of these tools has been individually validated to a varying degree, but the validation of a final integrated model is not an easy process. This is mainly due to the fact that modern processors do not provide a sufficient (if any) means to validate such models. This limitation stems from the nature of the verification process for these kinds of tools.

Validation of real-time processor metrics demands the measurement of real-time responses 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. However, adding a sufficient number of power counters to obtain the needed level of granularity would consume a significant amount of area, and impact power consumption 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. Without the measurement of real-time responses 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 cumu-

ExpCS, 13-14 June 2007, San Diego, CA

lative impact of many 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 the thermal or power behavior.

This paper proposes an infrastructure to directly measure temperature, power consumption, and performance on modern processor. The proposed measuring system uses infrared cameras to capture transient temperature fluctuations. Power consumption is gathered by isolating the current used by the processor during run time. Furthermore, performance statistics are gathered by sampling internal processor-specific performance counters. The data gathered by our proposed system can be used to validate the accuracy of many power and thermal models.

The measuring setup described in this paper captures thermal, power, and activity rates from modern processors. These three techniques have been proposed separately, but this is the first time that they are proposed as an integrated infrastructure. As a result, this paper proposes a unified measuring setup that captures temperatures on modern high-performance processors; develops image processing filters to increase the accuracy of thermal images; measures floorplan temperature on a real chip; and proposes techniques to synchronizes power, temperature, and performance data. Compared with other works [9], this papers focuses on how to build such a measurement setup and provides insights on failed setups.

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

2. RELATED WORK

Real power consumption measurements are a very useful tool. Isci et al [8] 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. A major difference between the setup described by Isci and our power measurement setup is the use of a shunt instead of clamp amp meter to increase measurement accuracy. In addition, we also take into account the efficiency of the on-board voltage regulator. A bigger difference is that fact that our setup also captures temperature.

A related work by Sung et al [5] builds on top of models that use performance counters to generate detailed thermal maps. Their work compares a less compute intensive regression model against a HotSpot thermal map result.

The thermal measurement setup has similarities with Hamann et al [7]. 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 measurements for steady state thermal images and they do not provide a method to capture power and/or activity rates synchronously.

The most related work [10] employs a similar thermal and power measurement setup. However, the focus of that work is on building a novel model that relates thermal processor maps to fine-grain power consumption by using a genetic algorithm. This paper furthers the scope of the work done in [10] by proposing the capture of processor performance counter data concurrently with power and thermal information. This paper discusses in deeper detail the thermal imaging, power and performance counter data gathering issues. Furthermore, some of the failed approaches that we incurred while trying to develop the measurement setup described in this paper.

3. INFRASTRUCTURE

In this paper we propose a system capable of measuring power and temperature characteristics of modern high performance processors, all done with a very fine degree of granularity. The proposed setup measures processor temperature using an infrared camera. Power consumption data is obtained by directly isolating the current used by the CPU at run time. Performance counters available in most modern processors are used to obtain traces with execution statistics during run time. Figure 1 displays the major components of the measuring setup.

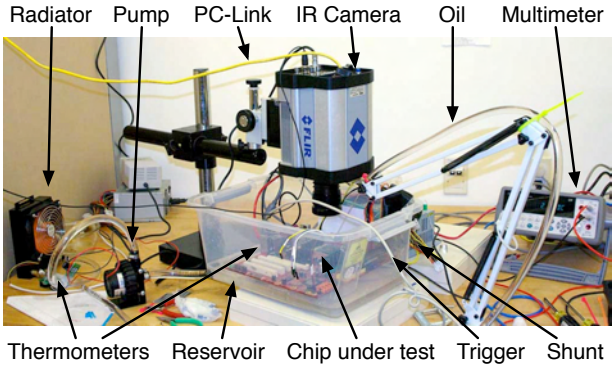


Figure 1: Measuring setup

First, the proposed **measuring setup** (Section 3.1) captures the chip temperature with an infrared (IR) camera. An infrared-transparent heat sink is used to allow the IR camera to obtain the processor die temperature. 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. The IR camera frame rate (125fps) can be increased up to 10KHz as long as the bandwidth of the camera stays under 1GB/s. E.g: If the measurement experiments require it, it is possible to capture

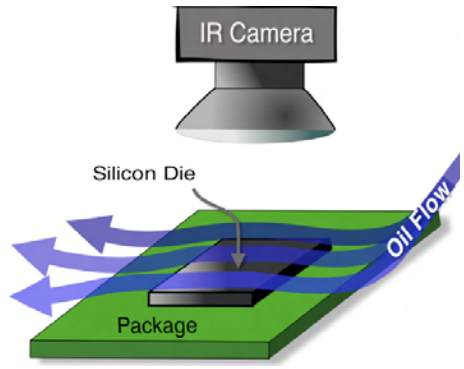
1000 thermal frames per second (1KHz) with a 100x80 resolution.

Second, the data must be processed to remove distortion. Modern IR cameras have a resolution of 320x200 or 640x400 pixels with a precision of 25mK error per pixel. Nevertheless, cameras suffer from a significant distortion of several degrees Kelvin error between pixels and need calibration for each specific lens, objective, and/or temperature range setup. To solve these problems and increase accuracy, the paper introduces a **thermal image processing** (Section 3.2) correction filter.

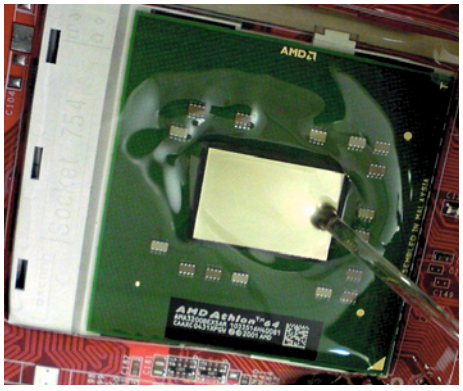
The **power measurement** setup is explained in Section 3.3, and the Linux kernel extensions to capture the performance counters are described in Section 3.4.

3.1 Measuring Setup

To generate an accurate thermal map for a given chip, we need to measure the temperature at multiple points. To do so, we use an infrared camera to measure temperature as close as possible to the transistor junction.



(a)



(b)

Figure 2: Oil-based heatsink with laminar flow (a), liquid heatsink in operation (b).

To keep the processor in operational conditions, an IR transparent heat sink must be implemented. To do so, we create a mineral oil (Fluka Mineral Oil 69808) flow on top of the silicon substrate (Figure 2). 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 the infrared spectrum. Several oils like olive oil are partially transparent on the infrared wavelengths. Fluka oil is designed for infrared spectrography and delivers excellent infrared pictures. Turbulent flow can remove more heat than laminar flow, however it is more complicated to correctly model. For that reason, on the L2 side of the processor, we add filling with the same height as the silicon. The oil impacts on the filling and generates a flow from L2 to the core. We keep the oil flowing as fast as possible to minimize heat transfer from L2 to the core. Our measurements show less than .1C 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 litters in the oil reservoir and connect a small radiator 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 spatial resolution. This camera operates on the 3-5μm wavelength (MWIR) a range of light where silicon is transparent. 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 chips – exposing the silicon substrate. Since the camera can measure temperatures through the silicon substrate, using flip-chips [1] greatly simplifies the task of measuring junction temperatures. Although the SC-4000 has 25mK sensitivity per pixel, to obtain accurate thermal measurements it requires extensive calibration. The following section explains it.

3.2 Thermal Image Processing

Due to their operational characteristics, infrared cameras need to be calibrated in order to compensate for different material emissivities, lens configurations, temperature range for the object/material to be measured, and a host of other factors. One approach to calibration is to have the infrared measuring device calibrated for the specific setup by the manufacturer. However, this tends to ignore 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 calibrate the camera, we perform two measurements: one with cold (289K) and one with hot (344K) min-

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

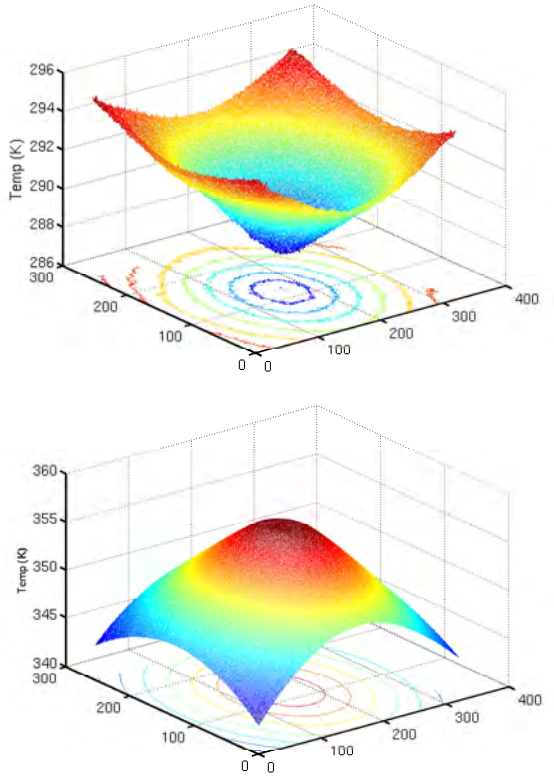


Figure 3: IR Camera error behavior, exposed with low (Top), and high temperature (Bottom) constant plane temperatures. (Temperature scale in Kelvin)

eral oil on top of the processor's silicon substrate. Figures 3-(a) and 3-(b) show the IR thermal measurements when the processor is powered off. We observe that for cold oil 3-(a) the center of the image closely resembles the measured temperature while the side pixels can have up to 6C error (288K vs 294K). The opposite effect is shown when the camera measures a uniformly hot mineral oil (345K vs 335K).

The camera is coupled with a set of lenses and extension rings to provide the desired magnification and focus function needed to differentiate between processor structures. The lenses and extension rings add a small distortion factor that increases the overall error in the thermal data. Although not as significant as the error in the IR sensor array itself, the compounding of the IR distortion and the differential error across the sensor array needs to be taken into account. To do so, we perform a two step filtering of the raw thermal imaging data.

First, to correct the lens distortion, a calibration grid provides a simple compensation function for both the lens and the ring extensions in the camera. This correction function is represented as a 2D vector field, which maps the actual camera image plane onto a transformed thermal image that takes into account both distortion and perspective correction.

Since both, the camera and test chip remain static during the measuring runs, we only need to produce a unique correction filter for the whole experiment. Accordingly, the camera needs to be registered just once. Figure 4 shows the experimental setup for performing the camera calibration and lens correction.

The distortion parameters are obtained by running an edge detection algorithm [3] on the sides of the processor die being measured, and estimating how distorted each of the edges of the processor core are. Distortion for each edge is measured as a simple least squares approximation of each edge assumed to be a 3D segment projected by a line [6]. The error is assumed to be the sum of squares of the least square approximation. In this model the "curvature" of a straight line is directly proportional to our defined error. The distortion factor is defined as $D = \sum_{i=0}^{3} E_i^2$, where E_i is the least squares approximation for each of the outer edges of the processor core. We assume the core to be a rectangle, thus we need to consider just 4 edges. The distortion factor is optimized by applying a simple linear minimization. The final lens correction function L is defined to be directly proportional to the optimized distortion factor D'.

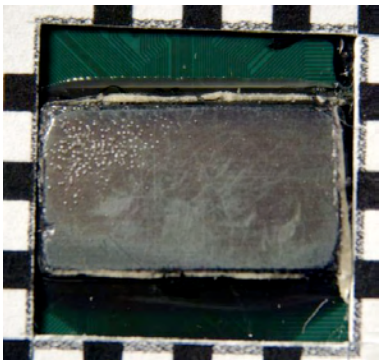
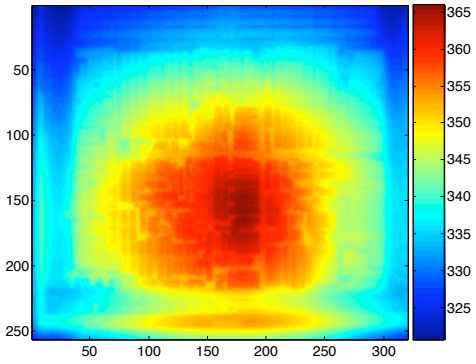


Figure 4: Test overlay chip used for camera registration and distortion calibration.

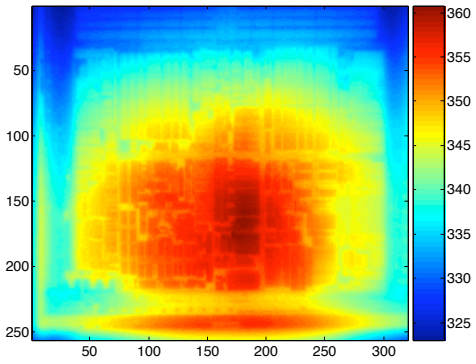
Second, we need to compensate for the intrinsic IR sensor error. The camera specifications indicate that a linear ($\texttt{"real temp"} = A * \texttt{"IR temp"} + B$) correction should be applied for each camera pixel. While the central pixels have a high accuracy, the side pixels need a compensation factor. Our image filter automatically generates a linear correction factor to compensate for the inaccuracies expected in the outer regions of the lens. A secondary filter is used to compensate for the optical distortion induced by the lens setup. After calibration the thermal error was reduced to 3%.

Figure 5-(a) shows the thermal map before the filter process is applied, and Figure 5-(b) shows the filtered thermal map. Further, Figure 6 shows the behavior of the IR filter when the thermal data is represented as a temperature elevation map across the core.

Using regular structures, such as the large L2 cache found in modern processors, we can estimate the performance of our correction filter. The uncalibrated thermal imaging data performs

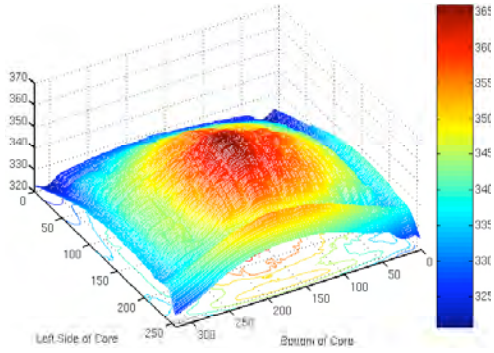


(a)

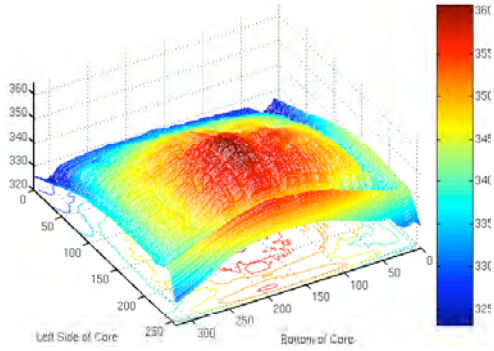


(b)

Figure 5: Effect of the IR correction filter on a sample image for an Athlon64 core during run-time. Thermal data unfiltered (a), and filtered (b). (Temperature scale in Kelvin)



(a)



(b)

Figure 6: Effect of the correction filter on a temperature elevation map for a sample thermal image for an Athlon64 core. Thermal data unfiltered (a), and filtered (b). (Temperature scale in Kelvin)

quite badly, sometimes reporting with as much as 50% error. A simple linear thermal filter reduces the expected error down to 3%.

The benefits of the IR correction filter are easily illustrated by considering the average temperatures for each of the major operational blocks for the Athlon 64 core being tested. Figure 7 shows the dramatic temperature differences for the same functional block between the raw and filtered IR data. Please refer to Figure 10 for an overview of the major blocks being itemized in the Athlon 64 floorplan.

3.3 Power Measurements

To measure the overall power consumption, we cut the 12V wires that provide power to the voltage regulators (VR) onboard. To have a low overhead and high accuracy measurement, we use a shunt (LTS 25-NP). A shunt provides higher accuracy than clamp current measurements at the cost of cutting the power supply cables to connect the shunt, but measuring the power supplied by the processor 12V cable is not enough. The reason is that the voltage regulators from the motherboard are not 100% efficient. To have accurate absolute power measurements, we discount the power wasted due to VR. Note that to know the efficiency, it is necessary to find the VR specifications.

The voltage reported by the shunt is measured with an Agilent 34410A multimeter. This multimeter is capable of sampling at 1KHz and storing over 50 seconds of execution. To read the power measurements, we use a TCP/IP ruby script.

3.4 Performance Measurements

By measuring the occurrence of performance-relevant events computer architects can gain added insight into the behavior of a processor under specific workloads. Modern processors offer hardware support to measure and gather performance-relevant events inside the processor's core. Performance counters are an example of such facilities. For example, one performance counter keeps track of the number of L1 cache misses, while another performance counter counts the number of retired instructions.

In this paper we consider an AMD Athlon 64 processor. The K8 core, in which our sample processor is based, provides four concurrent hardware performance counters, each counter is stored in a separate 48-bit wide register. The palette of performance events is much larger [1], for example K8 supports more than 48 different events to be monitored (Table 1). At run time, we can easily instruct each performance counter to monitor a spe-

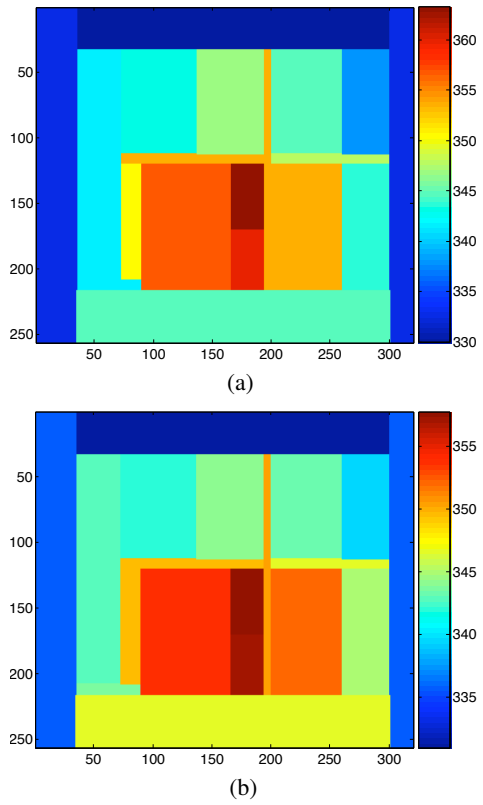


Figure 7: Effect of the IR correction filter on the average temperature for each of the major functional blocks for an Athlon64 core. Thermal data unfiltered (a), and filtered (b). (Temperature scale in Kelvin).

cific event. Depending on the type of event being tracked the counter will reflect two possible behaviors.

First, the counter can store the overall count of certain processor events. In this mode, whenever an instance of the event being tracked occurs, the processor increments the counter associated with this event. Examples of these types of events are L1 cache misses, number of instructions retired, number of replays, etc.

Second, the counter can be used to track the duration of specific events. The processor updates the counter with the number of cycles that were required to complete the action being tracked. An example of this would be a counter that keeps track of the number of cycles associated with servicing a cache miss.

A 64-bit wide time stamp counter (TSC) is also available within the K8 microarchitecture. The TSC operates concurrently with the four general counters in the K8, this counter keeps track of the overall number of cycles executed by the processor.

There are two main approaches to count events, once the mapping between a performance counter and a specific event has been formalized.

Efficiency	Instructions per cycle (IPC) Memory bandwidth
L1 Data Cache	Number of misses Miss ratio
L1 Instruction Cache	Number of misses Miss ratio
L2 Cache	Number of misses Miss ratio
Data TLB	Number of misses Miss ratio
Instruction TLB	Number of misses Miss ratio
Prediction Tables	Branch mispredictions Near return mispredicitons
Floating Point	Retired operations Exceptions
Load/Store Unit	Unaligned data accesses

Table 1: Types of events measured by the performance counters in K8-based cores.

Event counts can be read before and after a specific sampling period. These two counts can be subtracted to obtain the number of events that occurred. This approach is able to track the number of events, however there is no information regarding the distribution of such events through the sampling period.

A second approach is to load a performance counter with a value that is treated as a threshold or limit count. This value is passed to the performance measuring hardware inside the processor. Events will be tracked until the specified threshold is reached, then an interrupt is generated and the performance counter will reflect the event count when the interrupt was generated.

In order to get the required kHz resolution for the project, a modified Linux 2.6 kernel is used. We sample the performance counters during every scheduler tick, this approach has a significant overhead, so the system only runs a single process during the sampling period. Multiple runs are needed to obtain all performance-related events since only four counters and a TSC can be read during each scheduler cycle.

The counters are read in kernel-mode, not user-mode. However, the benchmark processes are run in user-mode. We modified the kernel to allow user-level messages to be trapped by the kernel to signify when the sampling should begin. The sampling period is fixed to a single kernel scheduler tick, and the sampling length is also fixed (generally 20 seconds). The kernel stores the sampled values in private pages. Once the sampling length timer expires, these kernel-level pages are passed down to the user-level through the /proc filesystem.

Running the sampling of performance counters in kernel-mode reduces the collection overhead associated with the elevated sampling rate for this project. The higher resolution results in a higher pollution of the caches as well as the predictor structures and the translation lookaside buffer (TLB). This pollution affects the memory behavior for the benchmarks being executed. Furthermore, multiple processes may be swapped during the sampling run, this contributes to the pollution of the counters

themselves, but it is a tradeoff that we consider necessary.

In order to further reduce the uncertainty in the data gathered, thermal throttling and other power and temperature-aware mechanisms inside the processor are disabled.

3.5 Failed Thermal Setups

The proposed thermal measurement setup has evolved over several iterations. Although most papers do not include failed experiments, we consider it important to include a quick summary so that other researchers do not fail again on the same problem.

Our first setup used thermal couples in contact with the processor substrate. Although this was an effective and fast method to measure temperature, it was very challenging to place a very thin thermal couple between the silicon substrate and the heatsink. More importantly, we could not devise any effective array of thermal couples to measure temperature at several points simultaneously.

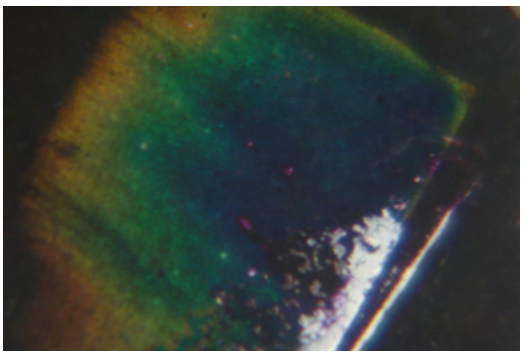


Figure 8: Detailed view of a K8 core coated with liquid crystal paint and the thermal response of the paint during run time.

To capture a more fine grain thermal measurements, we tried liquid crystal (LCs) which delivered significant resolution, but we found several challenges: the color response of LCs change over time; the temperature range (30-100C) is difficult to achieve with LCs because LCs tend to operate with lower temperature ranges (60-80C); the paint used to glue the LC degraded after several iterations; it was difficult to paint the chip without small irregularities (Figure 8); and paint is a good insulator which complicates the heatsink design. As a result, we decided to explore IR measurement setups. Figure 8 shows a LC painted Pentium processor after executing a matrix multiplication benchmark for several seconds. As it is easily observable by the picture, the paint is not regular due to decay and it is difficult to paint the chip homogeneously. Equally important, the color (temperature) measurements do not show the resolution observed by the IR camera. The reason is that the IR sees through the silicon substrate while the liquid crystals measure the temperature at the top of the chip substrate.

The major challenge in using an IR-camera was to develop an IR-transparent heatsink. The original setup used a air jet flow. This was enough to cool down up to 17W, common for mobile processors. The major issue was the complicated turbulent flow associated and more importantly, the low capacity to remove heat and the associated slow thermal response.

To cool the chip substrate faster, we explored several liquids. Water has a very high specific heat which makes it ideal for such experiments, but it is opaque to the IR camera. Some toxic fluoride materials are extremely transparent to IR with a high capacity to cool down modern (and future) processors. But, to avoid dangerous lab materials, we looked for safer products. The most transparent IR material was olive oil (enough to perform good measurements, but a bit to thick for the pump used).

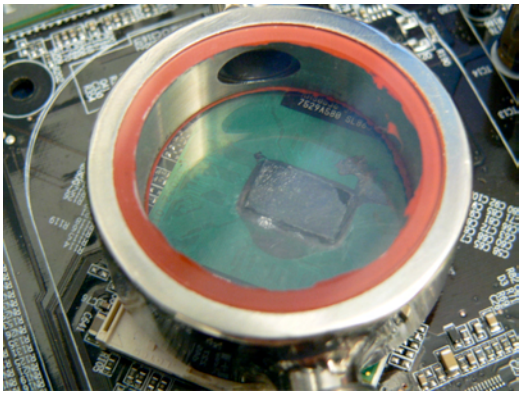


Figure 9: Transparent sapphire heat sink.

To have a clean setup, we designed several Sapphire heatsinks where the oil flowed through two layers of sapphire. This setup had a major problem: oil is transparent to IR, but it generates signals captured by the IR camera. As a result, to have accurate thermal measurements, the oil flow should be less than 2mm thick (Figure 9).

4. EVALUATION SETUP

While the measuring setup section (Section 3.1) explains the setup infrastructure required to measure any modern processor, this section explains the measured configuration. Table 2 summarizes the main processor and thermal parameters.

Parameter	Value	Parameter	Value
CPU	AMD/Athlon 64	Technology	130nm
CPU Model	AMN2800BIX5AR	Vdd	1.4v
Socket	754	Frequency	1.6GHz
Oil Spec. Heat	1.67kJ/Kg.K		

Table 2: Main processor and thermal model parameters.

Figure 10 shows floorplan blocks used in the evaluation. The infrared camera captures 320x200 pixels of resolution but we average the temperature for each block before performing further computation. The register file is marked in the middle of the floorplan with a question mark (RF?). We call this block register file because it looks like an SRAM-like structure and it

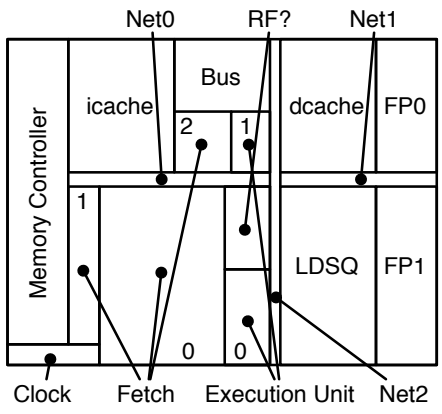


Figure 10: AMD Athlon 64 core floorplan blocks.

has a high power consumption. Nevertheless, we could not find any confirmation and it may be a different part of the execution units. Since only the caches are specifically used to validate the results, incorrectly naming a block does not affect the results. It is important to note that the floorplan is the flip version of the AMD floorplan commonly published. The reason is that we are measuring "through" the substrate, which flips the image.

4.1 Applications

The system under test boots Linux. For the evaluation, we gather thermal and power statistics for the first 20 seconds of each application measured. When the applications are launched, we use the serial port to trigger the multimeter. As a result, we have the measurements synchronized with the application initialization.

To have a diverse set of applications 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 performs matrix multiplications at 1Hz frequency (0.5 seconds matrix multiplication, 0.5 second idle).

5. EVALUATION

The main objective of the paper is to propose an infrastructure to capture temperature with high resolution and obtain the associated power and performance traces. We feel that such thermal, power, and activity rate traces are very valuable to the research community. Since these traces utilize several GBytes of data, it is impossible for us to include all the values on the paper. The evaluation focuses on the accuracy of the proposed measuring setup and highlights potential applications.

5.1 Main Results

Figure 11-(a) shows the temperature and power profile for the first 20 seconds of execution (2500 frames) of applu from SpecFP2000. The solid lines correspond to a run where the oil was heated to 33.5C. The dashed lines correspond to a run with a 22C oil temperature. As expected even though the initial oil temperature difference is just 11.5C the total temperature difference after 20 seconds is slightly higher (13C). The power consumption is higher on the high temperature (HT) run than on the low temperature (LT) run. The reason is that as we increased the oil temperature the leakage power also increased. The plot shows that on average for a 13C increase, total power consumption increased by 5.3%.

Figures 11-(b) and 11-(c) show the average temperature for several floorplan blocks as applu and apsi execute respectively. The floorplan blocks, starting from highest to lowest average temperature, are the register file, data cache (D$), floating point unit (FP0), clock generator, memory controller (MC), and instruction cache (I$). For both applications, the temperature across blocks is somewhat correlated. However, there are several situations where the data cache temperature increases while the register file temperature decreases (1.5s for applu (Figure 11-(b)), 4s for apsi (Figure 11-(c)). In both cases, we found a phase where the access to the register file decreased and the rate of memory operations increased. For most of the applications, the register file is close to 10C hotter than any other floorplan block.

Figure 12-(a) shows the number of retired instructions for 4 seconds of execution (4K samples) of applu and apsi. Since the retired number of instructions and sampling rate are known, it is easy to infer the IPC information for each benchmark. Since thermal throttling is disabled in our test processor, there is little deviation between the runs using different temperature oil for cooling.

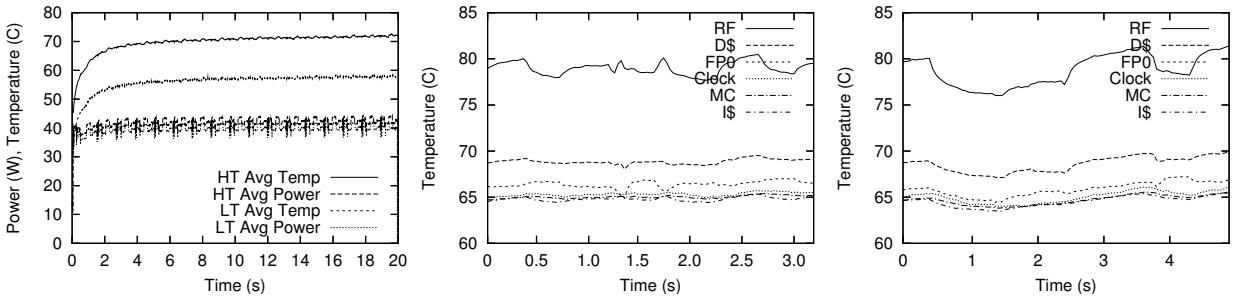
Figure 12-(b) shows in further detail statistics for the data cache behavior for both accesses and misses, as well as the number of FP operations dispatched during 4 seconds of execution (4K samples). Lastly, figure 12-(c) shows the complete behavior for instructions retired, replayed, and cache accesses during 20 seconds of sampling.

5.2 Thermal Imaging

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

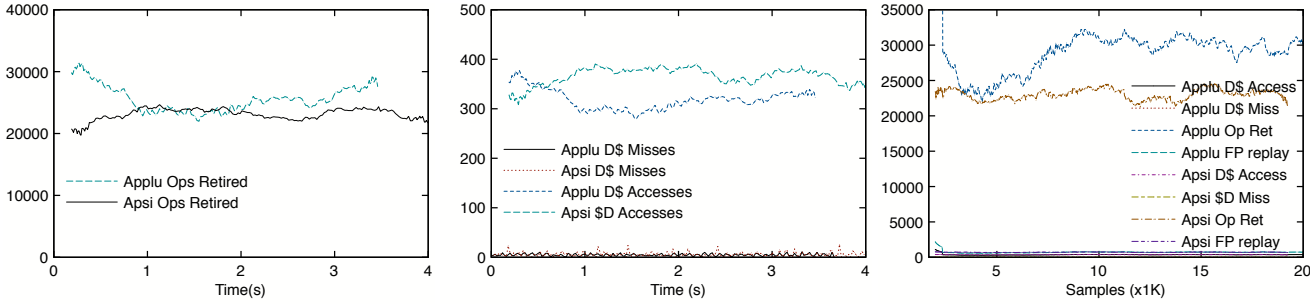
As section 3.2 states, the IR camera does not have the same accuracy over all the pixels. The central pixels have a higher accuracy than side pixels. To compensate for that error, we perform a different linear correction for each pixel. Figure 13-(a) shows the corrected thermal image for a single frame, including a floorplan overlay.

The overlay on Figure 13-(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 accurate for our setup.



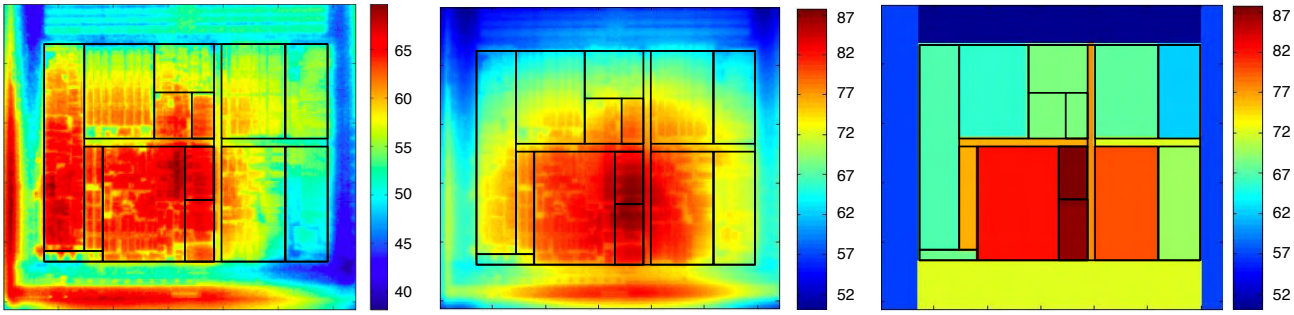
(a) (b) (c)

Figure 11: Thermal and power data for the first 20 seconds of applu (a); temperature profile for the memory controller (MC), register file (RF), instruction cache (I$), data cache (D$), and floating point unit for applu (b) and apsi (c).



(a) (b) (c)

Figure 12: Retired instruction statistics for 4 seconds of applu and apsi execution (a); detailed view for the D$ accesses and misses (b); execution behavior for 20K performance counter samples (c). Vertical axis represents number of operations/instructions.



(a) (b) (c)

Figure 13: Full-thermal image with overlapped floorplan (a); hottest captured image (b); and its average temperature per block (c).

Figures 13-(a) and 13-(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, although we report the average temperature for each floorplan block, our thermal model internally computes multiple temperature points for each block.

Figures 13-(b) and Figure 13-(c) show the frame from crafty (SpecINT2000) with the maximum temperature measured. On this frame the register file reached 84C. The average temperature per floorplan block is shown on Figure 13-(c).

6. CONCLUSIONS

In this paper we propose a system capable of simultaneously measuring the power, temperature, and performance characteristics of a modern high performance processor. This is all done with a very fine degree of granularity. We believe the development of such infrastructures is necessary in order to develop and validate advanced thermal and power models.

The main contributions of this paper are not only the development of a working measurement system, with the necessary hardware and software support and integration. But also the description of failed setups that may provide added insights to future developments in this area of experimentation.

The setup was described in detail, and further data and software is available publicly. Our goal is to allow other research groups to reproduce our approach, to improve on it, and to apply it to different research topics in the computer architecture community.

Acknowledgments

We like to thank the reviewers for their feedback on the paper. This work was supported in part by the National Science Foundation under grants 0546819; Special Research Grant from the University of California, Santa Cruz; Sun OpenSPARC Center of Excellence at UCSC; gifts from SUN, Altera, and ChipEDA.

7. REFERENCES

[1] *BIOS and Kernel Developer's Guide for AMD Athlon 64 and AMD Opteron Processors*. Advanced Micro Devices, Inc., Apr 2004.

[2] 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.

[3] C. Cafforio, E. Di Sciascio, C. Guaragnella, and G. Piscitelli. A simple and effective edge detector. In *ICIAP '97: Proceedings of the 9th International Conference on Image Analysis and Processing-Volume I*, pages 134–141, London, UK, 1997. Springer-Verlag.

[4] 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.

[5] Sung Woo 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.

[6] F. Devernay and O. Faugeras. Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments. *Machine Vision and Applications*, 13(1):14–24, 2001.

[7] 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.

[8] 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.

[9] F. Mesa-Martinez, J. Nayfach-Battilan, and J. Renau. Power Model Validation Through Thermal Measurements. In *International Symposium on Computer Architecture*, Jun 2007.

[10] F. Mesa-Martinez, J. Nayfach-Battilana, and J. Renau. Power model validation through thermal measurements. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, New York, NY, USA, June 2007. ACM Press.

[11] 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.

[12] 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.

[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.