Analytical Model for Sensor Placement on Microprocessors

Kyeong-Jae Lee, Kevin Skadron, and Wei Huang§
Departments of Computer Science, and Electrical and Computer Engineering§
University of Virginia
kl2z@alumni.virginia.edu, skadron@cs.virginia.edu, wh6p@virginia.edu

Abstract

Thermal management in microprocessors has become a major design challenge in recent years. Thermal monitoring through hardware sensors is important, and these sensors must be carefully placed on the chip to account for thermal gradients. In this paper, we present an analytical model that describes the maximum temperature differential between a hot spot and a region of interest based on their distance and processor packaging information. We also use a runtime thermal model, as an illustration of virtual sensors, and examine two benchmarks that exhibit highly concentrated thermal stress. We then use our analytical model to demonstrate the safety margins of the chip. Ultimately, the mathematical expression allows designers to obtain worst-case behavior of thermal heatup and select the optimal location of additional sensors.

1. Introduction

As an effort to reduce costs associated with increasing heat dissipation in processors, researchers have developed various dynamic thermal management (DTM) techniques. Ideally, a chip's temperature would be monitored at a fine granularity using many sensors. However, most sensors are based on analog CMOS circuit designs. Precise temperature measurement typically requires matched transistors, and as process variations grow in severity, accurate sensors may require large device sizes to compensate. This in turn increases their size and power requirements. On-chip sensors can also be difficult to calibrate, and adding sensors may therefore increase testing costs. To date, CPU designers have used at most a few sensors. (One notable exception is the recently announced IBM POWER5 [1], with 24 on-chip sensors.) The limited number of sensors makes their placement on the chip important for reliability. It might become possible to create a program that targets a unit far away from the sensor for overheating. The localized overheating may not be visible to distant sensors, and cause permanent damage to the chip. The choice of thermal safety margins will also dictate sensor number and placement; smaller margins are less wasteful but require more sensors.

This paper proposes a framework to examine the maximum temperature gradient in processors and place sensors accordingly to account for cross-chip temperature variations. Virtual sensors, which are modeled in software, are explored as an alternative to hardware sensors. We also present a simple case study that examines thermal security risks and implications for sensor placement.

The paper is structured as follows. Section 2 gives an overview of the expression. Section 3 explains the use of virtual sensors. Section 4 presents a case study using the expression to examine thermal security risks. Section 5 concludes the paper.

2. Modeling Temperature Differentials

2.1. Deriving the Equation

Understanding the various hot spots on the chip is one of the most important factors to consider when placing a hardware sensor on a chip. Hot regions can be effectively located through simulations or runtime measurements. But Lee and Skadron [5] have also demonstrated that hot spots can move around the chip during the execution of the program. Hence, locating a sensor near one well known hot spot is often not enough to ensure that that is the maximum temperature on the chip. Another important factor to consider is the maximum possible temperature differential between a hot spot and a potential sensor location. This information can then be used to determine various safety margins and the most effective spacing between hardware sensors as well as perhaps the optimal number of sensors.

The following expression describes the maximum temperature differential from a given location. The temperature at the hot spot (source) –where $r = 0$– is the highest. At a point of interest, located at a distance r from the hot spot, the temperature difference between the two points is given

In ICCD 2005, October 2-5, 2005, San Jose, California, USA.

by $T_{src}(r)$; i.e., the temperature at r is $T_{src}(r)$ degrees cooler than that at the source.

$$T_{src}(r) = T_{src-max} \times \left(1 - e^{-\frac{2 \cdot r}{K}}\right) \quad (1)$$

The constant K denotes the thickness of the processor package –die, heat spreader, and thermal interface material– in terms of silicon. The thickness of each packaging material is multiplied by a factor based on the thermal resistivity of the material and that of silicon. After obtaining the equivalent thickness of the material in terms of silicon, these values are summed to obtain the value of K. T_{max} can be derived for two different cases: one where there is a single power source, and the other where there are multiple power sources. When there is a single power source, the temperature drops off exponentially as the distance from the source increases. When the source is at its local maximum power density, the value of T_{max} can then be derived as the difference between the maximum and minimum temperature value.

However, there is limited use for the equation for a single power source. In practice, each functional unit can be modeled as a power source, and many of these will be active simultaneously. The complexity of having multiple power sources interacting does not easily permit us to find an exact equation to fit the temperature-distance curve. Nonetheless, we can derive T_{max} to be an upper limit. The effects of different power sources are assumed to be independent of each other and follow a linear relation allowing superposition. Thus, for multiple power sources, $T_{src-max}$ can be derived as follows:

$$T_{src-max} = \sum_{i \in Units} (T_{i-max} - f_a \times T_i(r_{src-i})) \quad (2)$$

The T_{i-max} contribution for each different unit is summed up, then subtracted by a contribution equal to the temperature that each unit adds to the source. Since the T_i values are derived from maximum power cases, the latter part is multiplied by an activity factor in the range of 0.0 to 1.0. If f_a equals 1.0, then all the power sources are active and outputting maximum power. Hence, $T_{src-max}$ is minimized and the difference between different heat sources is smaller. This is consistent with our intuition, since the temperature *difference* between two points would actually be smaller if both points were acting as a power source. A smaller f_a value indicates that most units are less active while one or more units has the potential to be highly active. This results in a larger $T_{src-max}$ value, which implies that a larger temperature gradient is possible. A thermal stressmark or virus takes this behavior to an extreme, creating a very hot region potentially far away from the sensor. Section 4 explains this behavior and how to use the equation in more detail.

2.2. Using the Temperature Curve

In this section we explain two possible ways to use the temperature curve, both illustrated in Figure 1. First, the curve tells us the maximum temperature differential between a hot spot and a thermal sensor. For example, let the temperature at the hottest spot –$r = 0$– be T_0. Assume that the sensor is located at a distance of r_s. The curve will indicate that the maximum possible temperature difference is, for example, $T_\Delta = T_{src}(r_s)$. Thus, if the the sensor reads a temperature value of T_s, then $T_0 - T_s < T_\Delta$, or $T_0 < T_s + T_\Delta$. If the hottest spot starts to heat up rapidly, as in a thermal virus, T_Δ indicates the maximum temperature gradient between the two points no matter how high the temperature rises.

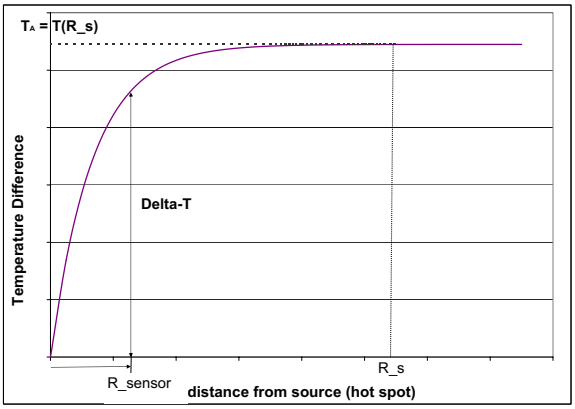

Figure 1. Maximum temperature differential

In addition, if a sensor is already placed near a hot spot, the temperature curve can be used to place a second sensor so that the temperature difference between two them cannot be any larger ΔT. Solving for r_{sensor} in equation(1) gives:

$$r_{sensor} = 0.5 \cdot K \cdot ln\left(\frac{T_{src-max}}{T_{src-max} - \Delta T}\right) \quad (3)$$

3. Virtual Sensors

Placing as many sensors as possible would be ideal, however, design constraints may limit the number of sensors that can be implemented on a chip. Nonetheless, not all sensors have to be physical CMOS circuits. Virtual sensors –modeled through software– can be created to supplement the existing hardware sensors. The equations described in Section 2 do not distinguish between physical sensors and virtual sensors.

Skadron et al. proposed a thermal model called *HotSpot*, which computes the temperature for each microarchitecture block on the chip [6]. In prior work, we have extended HotSpot to interface with performance counters [5] based

on a power model by Isci and Martonosi [4]. This creates a dynamic online model that estimates temperature readings based on real workload at runtime. This creates virtual sensors, one for each functional unit.

One important area for future work is to develop a very simple thermal computation based on performance counters to model virtual sensors. While HotSpot has the advantage of providing a very detailed thermal profile of the processor, it may be desirable to trade off some accuracy to obtain a radically simpler algorithm.

4. Case Study

4.1. BPU Stressmarks

The main purpose of this section is to demonstrate the use of our analytical expression as a sensible method for examining sensor placement. This case study examines two artificial benchmarks that were created to emulate the characteristics of thermal viruses; highly concentrated thermal stress in one unit, and minimal activity in other regions. We use the extended HotSpot model as virtual sensors [5]. Although the extended Pentium 4 HotSpot model has not yet been fully validated for accuracy, the resulting thermal stress patterns nevertheless provide valuable information to demonstrate our equation for effective sensor placement and for studying localized heating effects on the chip.

Both benchmarks attempt to stress the branch prediction units. The first experiment uses a program that contains approximately 90 if-statements within a large loop. During the program's peak execution, the results show that the temperature of the L2_BPU increases by roughly 12^{o}C. While the L2_BPU does heat up more than usual, the integer units still remain the hottest units on the chip. This is consistent with most applications; the hot spots are located in the lower region of the chip where the integer units and the rename/queue/scheduling units are located.

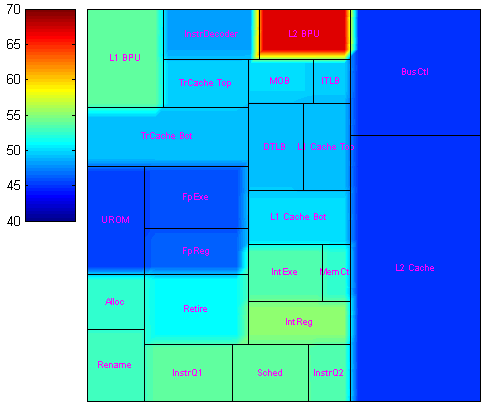

Figure 2. BPU stressmark #2

The second benchmark slightly modifies the first benchmark program. Within the compiled assembly code of the first program, nearly all cmpl and movl instructions are removed, thus creating a series of conditional branch instructions –jg or jle– inside the main loop. The results show that the temperature of the L2_BPU increases above that of the integer execution core. Figure 2 shows the thermal map at a particular instance in time. Exact temperatures are uncertain due to uncertainties in our model.

4.2. Analysis

Intel has carefully considered every possible thermal gradient in its sensor placement and design margin. The Pentium 4 also has advanced thermal management techniques [2, 3]. We use the Pentium 4 in our study not because we can induce damage, but believe it is safe to experiment with given all its built-in thermal protection. Despite the experimental limitations of the BPU stressmark characterization, it gives us a hypothetical scenario in which a unit heats up remote from any sensor while the units near the sensor remain colder. This lets us test our analysis using the framework developed in Section 2.

In order to find the $T_{src-max}$ when there are multiple power source, we need to first find the single-source T_{max} value for every unit of the Pentium 4 processor. For each unit, we artificially assign each unit's local maximum power and assign zero power to all other units. We then obtain the steady-state temperature readings and calculate T_{max}. Since the floorplan dimensions are known, we can plot this temperature-distance relation. The distance is approximated as the distance between the center of two units.

To analyze the BPU stressmarks in Figure 2, note that the IntReg and the L2_BPU are the hottest units on the chip respectively. Table 1 shows the results for all of the key parameters derived using equation(2) . The activity factor was derived using the two BPU stressmarks in Section 4.1. This factor could be adjusted for average or worst-case scenarios.

	K (mm)	f_a	$T_{max}(multi)$	$T_{max}(single)$
L2_BPU	2.69	0.4	36.37 ^oC	10.9 ^oC
IntReg	2.69	0.5	27.77 ^oC	1.34 ^oC

Table 1. Parameters for the temperature equation

Using these data, we can obtain the equations that describe the maximum temperature differential when the IntReg and the L2_BPU are the hottest units respectively. Figure 3 shows these curves as well as the actual observed temperature data. The actual data is taken from the same temperature sample seen in Figure 2. When IntReg is the hottest

unit (not shown in Figure 2), all the data points lie under the $T_{ireg}(r)$ curve. In Figure 2, L2_BPU is the hottest unit, and all the data points in Figure 3 fit well within the $T_{bpu}(r)$ curve.

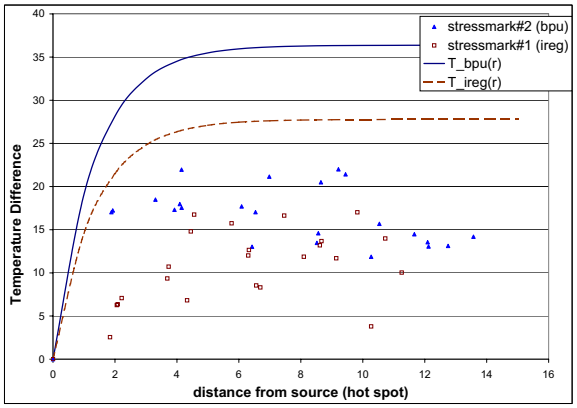

Figure 3. Benchmark Temperature Curves

If most functional units are below normal operating temperatures, heating could be too isolated and the thermal monitor may not operate or may respond too late when the chip has already been permanently damaged. The results in Section 4.1 show early work indicating the conceptual possibility of thermal security attacks through thermal viruses. However, as seen in Figure 3, these benchmarks do not exhibit thermal behavior near the thermal limits described by the temperature curves. The analysis further supports the fact that the Pentium 4 is not vulnerable in our case study.

Hardware designers can gain a better understanding of the thermal gradient bounds using the analytical method presented in this paper. By effectively identifying thermal safety margins through average or worst-case thermal behavior, designers may be able to recognize the need for more sensors as well as improve sensor placement on the chip. Furthermore, not all processors have on-chip sensors, or place them in the package but not in the chip itself. Those chips may be vulnerable to thermal attacks unless they use very large design margins.

5. Conclusions

Uneven activity associated with each functional unit within a processor results in localized hot spots. If this is not factored into the design or monitored by sensors –real or virtual– then reliable operation may be at risk. Thus, it is important to be able to efficiently place hardware sensors to monitor the temperature of the processor. The equation described in Section 2 provides a qualitative limit in terms of the thermal gradient that is achievable.

Ultimately, hardware sensors and virtual sensors can be used in conjunction to provide detailed information about the thermal distribution of the processor. Design constraints, including cost, may limit the number of hardware sensors. Creating efficient algorithms for virtual sensors is an important area of future work. Our analytical method can guide designers to effectively use such sensors to provide more thermal protection. Our work illustrates the need for on-chip sensors and their careful placement to avoid thermal risk. While we used the Pentium 4 in our case study, its thermal design is robust. The POWER5 is even more aggressive, with 24 on-chip sensors [1]. Chips without on-chip sensors are either incurring excessive design margin or may be vulnerable to thermal attacks.

Acknowledgments

This work is supported in part by the National Science Foundation under grant no. CCR-0133634, ARO under grant no. W911NF-04-1-0288, a grant from Intel MRL, the University of Virginia Fund for Excellence in Science and Technology, and the University of Virginia Summer Science and Engineering Scholars Program. We would also like to thank Mircea Stan and the anonymous reviewers for their constructive feedback.

References

[1] J. Clabes et al. Design and implementation of the power5 microprocessor. In *ISSCC Digest of Technical Papers*, pp.56-57, Feb. 2004

[2] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Tech. J.*, Q1 2001

[3] Intel Corporation. IA-32 Intel Arch. Software Developers Manual, Vol. 3: System Programming Guide, 2004

[4] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proc. MICRO-36*, Dec. 2003

[5] K.-J. Lee and K. Skadron. Using performance counters for runtime temperature sensing in high-performance processors. In *Proc. Workshop on High-Performance, Power-Aware Computing in conjunction with IPDPS*, Apr. 2005

[6] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proc. ISCA-30*, Apr. 2003