A Novel Software Solution for Localized Thermal Problems

Sung Woo Chung[1,*] and Kevin Skadron[2]

[1] Division of Computer and Communication Engineering,
Korea University, Seoul 136-713, Korea
swchung@korea.ac.kr
[2] Department of Computer Science,
University of Virginia, Charlottesville 22904, USA
skadron@cs.virginia.edu

Abstract. In this paper, we propose a temperature-aware DFS (Dynamic Frequency Scaling) technique using the performance counters that is already embedded in the commercial microprocessors. By using performance counters and simple regression analysis, we can predict the localized temperature and efficiently schedule the tasks considering the temperature. The proposed technique is especially beneficial to potential localized thermal problems that are inevitable due to limited number of costly CMOS thermal sensors. When localized thermal problems that were not detected by thermal sensors are found after fabrication, the thermal problems can be avoided by the proposed software solution without re-fabrication costs. The evaluation results show that the proposed technique is comparable to the DFS technique using CMOS thermal sensors.

1 Introduction

Reducing energy consumption has been one of the most interesting research topics in the computer architecture field. As technology trends leads to packing transistors ever more tightly, power densities are increasing rapidly. The higher heat flux leads to higher cooling costs-otherwise high temperature might cause the unexpected functional errors or permanent damage of microprocessors, especially in high-performance microprocessors. Thus, it is important to control the temperature as well as the energy consumption. To control the temperature, a couple of techniques have been proposed. One is to use the cooling fan to lower the temperature of a chip and the other is to make a heat spreader more efficiently. For example, Intel's Pentium 4 already has a cooling fan and an efficient heat spreader [20][24] and PowerMac G5 has huge cooling pumps [18]. To solve the thermal problems, on the computer architectural level, pipeline throttling, DVS (Dynamic Voltage Scaling), and DFS (Dynamic Frequency Scaling) have been proposed [2][14][16].

To control the temperature, we need to know the actual temperature of the functional block that needs to be controlled. In the Pentium 4, there are two independent thermal sensors [19]. By using on-die temperature sensing circuit and a fast acting temperature control circuit, the processor can rapidly initiate thermal management

[*] Corresponding author.

control. The Pentium 4, however, only uses one of its sensors for thermal management; the other is for external use and is not located near any anticipated hotspots. In fact, hotspots may move over time, depending on which on-chip functional blocks (register fie, integer arithmetic, floating-point arithmetic, etc.) are most heavily used [8]. As technology scales down, power density increases which might lead to more localized hotspots. Temperature differences become exponentially larger with distance, so a single thermal sensor does not cover a large chip like the Pentium 4. In future high-performance microprocessors, more than ten thermal sensors are expected to be embedded in a microprocessor. However, the number of thermal sensors is limited, because they are too expensive to be placed in all the potential hotspots. When potential hotspots that do not have thermal sensors are found serious after fabrication, it is impossible to resolve the localized thermal problems without re-fabrication, using previous techniques.

We chose DFS instead of DVS for the scheduling policy. There are three reasons. 1) The frequency transition at the high Vcc is done within few microseconds, which takes much less, compared to the voltage transition [11]. 2) We found a linear proportional relation between the frequency and the temperature by using simple regression analysis. On the other hand, the voltage is not linearly proportional to the temperature, which makes it difficult to find a relation between them. 3) In terms of reliability, the supply voltage scaling reaches a plateau, since the difference between supply voltage and threshold voltage should be kept large enough [6]. Thus, this paper proposes a ***DFS technique using performance counters*** that efficiently controls the temperature of the localized hotspots. The localized thermal problems that were found after fabrication can be resolved by using the proposed technique.

The rest of this paper is organized as follows. Section 2 presents related works. Section 3 explains the temperature-aware DFS scheduling using performance counters. Section 4 describes the experiment methodology and Section 5 shows the efficiency of the proposed technique. Section 6 concludes the paper and describes some avenues for future works.

2 Related Works

Huang et al. [4] proposed a DVS-based technique for thermal control. Though they investigated the memory hierarchy, they did not examine other hot functional blocks such as register files. Brooks et al. [2] set a constant threshold power and they applied five thermal control techniques (clock frequency scaling, voltage and frequency scaling, decode throttling, speculation control, and I-cache toggling), when the threshold power was exceeded. They found DFS and DVS to be inefficient because of the invocation overhead. However, the inefficiency may be due to the short sampling period (10K cycles) and large invocation overhead (more than 10 ms). Skadron et al. [12] proposed formal control theory for dynamic thermal management. The previous studies used constant trigger temperature (or power) and fixed response. In contrast, they allow the fetch-toggling rate to be changed according to the thermal history that may need additional storage. There are some previous works [8][10] on thermal management in SMP systems, which schedules the tasks making use of the idle SMP nodes. Srinivasan et al. proposed the predictive dynamic thermal management by profiling multimedia applications [16]. Most of these researches are based on the thermal sensors to measure the temperature.

Though the number of thermal sensors is limited by design budget, localized hotspots are too serious to be ignored [8]. Alternative to the thermal sensor is the performance counter that was already embedded in microprocessors to evaluate the performance. There have been several studies on using performance counters. Brooks et al. proposed using performance counters to find activity factors [2], where details were not proposed. Bellosa et al. proposed formulas that correlate the activity factor to energy that is eventually correlated to temperature [1]. They tried to manage the temperature by controlling power consumption [1][16] . They only concentrated on the overall temperature (not on the localized hotspots). Lee et al. [6] also proposed runtime temperature sensing using performance counters, which is accurate but incurs some computational complexity, because they use full HotSpot [13][14].

In this paper, we present a software technique using performance counters that can investigate the localized hotspots. To estimate the temperature of functional blocks, we only have to calculate a simple linear formula with inputs from the activity factor (the number of accesses) of the functional block. The linear formula is established by simple regression analysis. The data (activity factor(X) and temperature(Y)) for regression analysis can be obtained from real measurement in laboratories or from accurate simulations. In this paper, the parameters for regression analysis are obtained from simulation using HotSpot [13][14]. Though the performance counters are read every 10 ms, the estimated temperature was shown to be accurate enough [3]. In addition, the frequency transition overhead that is done every 10 ms is negligible [11].

3 Temperature-Aware DFS Technique Using Performance Counters

We examine two methods to measure the temperature: One is using CMOS thermal sensors and the other is using performance counters. The former is more accurate but needs CMOS thermal sensors. In other words, the thermal sensors should be placed in the localized hotspots before fabrication. The latter is less accurate but does not need additional hardware, since performance counters are already embedded in commercial microprocessors. On-chip sensors are now widely used to measure the temperature but are believed by many designers to be too expensive to be placed in all the potential localized hotspots. To alleviate the cost of the thermal sensors, only very probable localized hotspots have the thermal sensors. After fabrication, there is a possibility that severe localized hotspots that were not detected at the time of validation, are found. For this case, we propose a temperature-aware DFS technique using performance counters for sensing the temperature of the possible localized hotspots. Originally, the performance counters are used to count specific micro-architectural events for debugging and performance measurements. However, we can examine lots of localized hotspots by utilizing performance counters. For example, in the Intel Pentium 4, there are 45 configurable events and 18 physical performance counters, which implies that we can estimate temperatures of the 45 functional blocks in the microprocessors [15][27] .

For the temperature-aware scheduling, simple *offline* regression analysis [3] is used to find a simple relation between selected values of activity factor and observed values of temperature. Please recall that the most probable value of Y can be predicted for any value of X by simple regression analysis. Temperature can be estimated using a simple formula (T=*a*x + *b*, where T is temperature, X is activity factor, and *a* and *b* are coefficients). We only have to consider only the activity factor of the functional unit that is investigated. The key observation is that the regression captures second-order contributions from other functional units. We did try multiple regression analysis with the current activity factor and the previous activity factor. Results were at best minimally improved compared to results from simple regression analysis, and in fact the accuracy with multiple regression analysis was sometimes worse.

At runtime, multiplying the activity factor by the regression coefficient is required for temperature measurement. Although it is feasible to re-compute temperature every cycle, this is wasteful, since even at the fine granularity of architectural units, it takes at least 100K cycles until the temperature rise by 0.1C [14]. We chose a sample period 10 ms, which is the scheduling granularity of commercial operating systems and creates a natural opportunity for software to read the performance counts. For our CPU clock rate of 2.6 GHz, this works out to be sampling period of 26 M cycles. This is in any case the minimum granularity at which software techniques could perform any kinds of thermal management. For example, to compute the temperatures of the integer register file, we only utilized the IIPC (Integer Instructions Per Cycle) statistic. Although the peak temperature estimation error was small, there were times when our technique under- or over-estimated temperatures by as much as 10 degree. These large differences only occurred when the performance counter technique responded faster than the actual temperature. The reason is that the proposed technique is linearly proportional to the IIPC so that the estimated temperature changes quickly, whereas the actual temperature changes gradually. We did not mediate these spikes and dips, since we may be able to schedule tasks more efficiently if we know the temperature tendency (increase/decrease) in advance.

In this paper, we compare the scheduling efficiency using the thermal sensors to that using the performance counters. In the conventional technique using thermal sensors, the frequency is lowered when the temperature is more than (or same as) the threshold temperature and the actual temperature is measured from the thermal sensors. On the contrary, in case of the proposed technique using performance counters, the temperature is estimated from the activity factor so that the frequency is lowered when the activity factor (instead of the actual temperature) is more than a threshold.

4 Experiment Methodology

The processor used for the experiments is a 2.6 GHz Pentium 4, 130 nm Northwood core. The typical power dissipation is 69.0 W, and the operating voltage is 1.6 V [23]. The processor supports hyper-threading technology, which allows the processor to run two threads simultaneously. This means that the task that regularly reads the performance counters and calculates the temperature interferes minimally with user tasks: not only does it consist of only a few instructions, but hyper-threading fits these few instructions into empty execution slots as instructions are issued within the processor.

The performance counters are used to count specific micro-architectural events for debugging and performance measurements [21]. Each counter is associated with one counter configurable control register (CCCR), which determines the specific counting scheme. The event selection control registers (ESCRs) determine which event is to be counted. A simplified device driver, adapted from the abyss device driver [27], is used to configure all the control registers and read the performance counters.

The temperature model requires the geometric specifications and the floorplan layout of the processor. We derived the configurations of Pentium 4 to configure HotSpot [13][14]. These parameters are based on design schematics found in [23]. We also use the floorplan layout that was adapted from the Northwood core die photo [22].

Though we are able to investigate the temperature of 45 functional blocks through performance counters, we concentrate on the register file which is known as one of the hottest functional blocks. In the simple regression analysis, IIPC is X (selected value) and the temperature is Y (observed value). The actual temperature is obtained from the HotSpot [13][14] that was proven to be accurate. To use the performance counters, the Hotspot was modified to be based on a model by Isci and Martonosi [5] for the Pentium 4.

We selected four benchmarks (bzip2, gap, gcc and parser) from the SPEC CPU2000 benchmark suite [26], since these benchmarks show more temperature differences than other benchmarks during the execution. Since running single benchmark of these four benchmarks does not increase the temperature so much, we would like to run two benchmarks at the same time. However, running two benchmarks on two threads sometimes defers reading the performance counters severely and incurs thermal throttling by the Pentium 4 processor, resulting in inefficient evaluation of scheduling techniques. To prevent the inefficiency, we schedule the tasks off-line instead of on-line. We ran two applications separately and obtained the trace of the activity factor of all functional blocks. After then, we utilize off-line task scheduling, by using activity factor of all functional blocks. When the proposed technique using performance counters is adopted in the real world, the access to the performance counter can be set to have a higher priority than the other tasks in order to allow periodic accesses to the performance counter.

By running applications, we can have the coefficients for the formula. For more accurate estimation, we only use the samples whose IIPC is more than 2.0. We set the confidence interval is 99% in order to cover as many cases as possible. The formula that we obtained from the simple regression analysis is $Y = 14.1*X + 58.4$, where the IIPC (X) corresponding to 95 Celsius (Y) is 2.59.

The DFS using thermal sensors lowers the frequency by 20% when the temperature is same as (or more than) 95 Celsius. It increases the frequency by 5% every 10 ms up to the 2.6 GHz when the temperature is lower than 95 Celsius. The DFS using performance counters lowers the frequency to (2.6 GHz * (2.59/previous IIPC)), when the IIPC is more than 2.59. When the IIPC is lower than 2.59, the frequency is 2.6 GHz.

5 Evaluations

We evaluate the proposed DFS scheduling technique in six cases: bzip2 + gap, bzip2 + gcc, bzip2 + parser, gap + gcc, gap + parser, and gcc + parser. According to [25], maximum temperatures are between 65~100 Celsius in commercial microprocessors, depending on the model. We set the threshold temperature to 95 Celsius. We also assume that the frequency can be freely set not to distort the experiment results by discrete frequency.

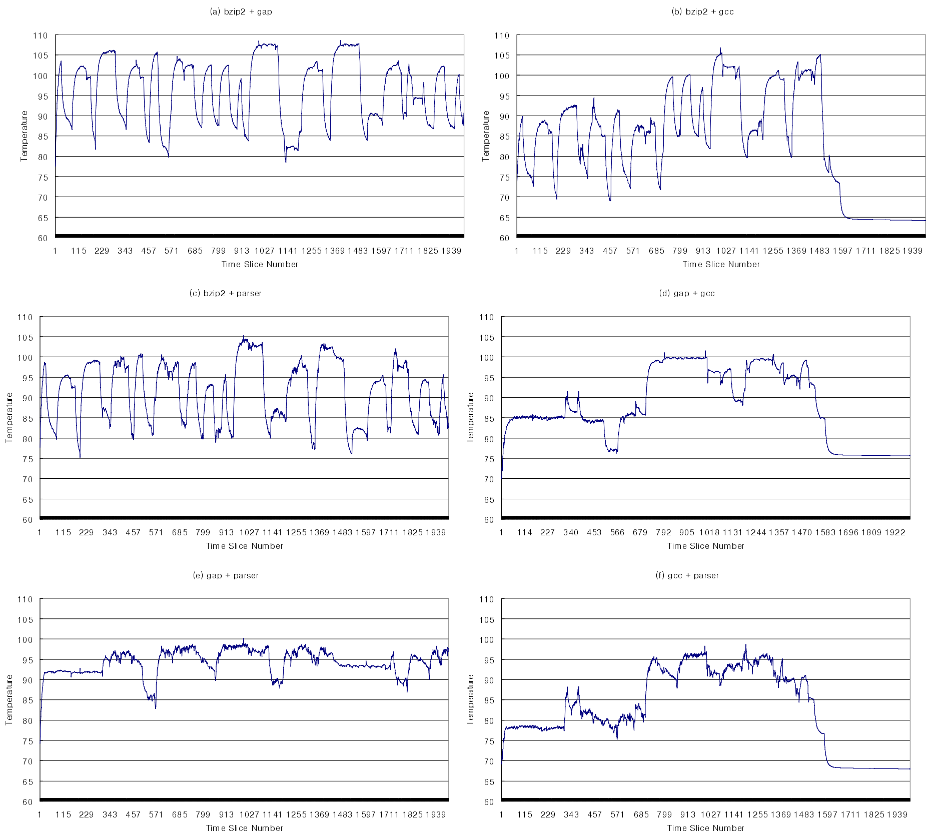

Fig. 1. Temperature changes (w/o DFS)

5.1 Scheduling Efficiency

Figure 1 shows the temperature changes when there is no consideration for temperature. In Figure 1, the temperature varies fast in (a), (b) and (c) due to the characteristic of *bzip2*, whereas the temperature does not vary so much and it is under 100 Celsius in (d), (e) and (f).

Figure 2 shows the temperature changes when DFS using CMOS thermal sensors is applied. As shown in the Figure 2, the temperature is varied significantly when the temperature is around 95 Celsius. The reason is that the frequency increases/decreases by a constant rate (20% for increase and 5% for decrease). If the frequency is decreased only by 10% or less, the temperature remains over 95 Celsius for longer time. When the frequency is increased more gradually, the performance loss will be severe. If the frequency is increased more than 5%, there are more temperature violations. Please note that there is no run-time information on how much the frequency should be changed. In fact, we tried to make use of the temperature history to find patterns of temperature variation in order to utilize the run-time information, which turned out not so helpful.

Figure 3 describes the temperature changes when the DFS is applied using performance counters. Different from Figure 2 where CMOS thermal sensors are used, Figure 3 does not show the spikes and dips of the temperature around 95 Celsius. In the proposed technique, the frequency is determined by referencing to the previous IIPC. When the previous IIPC is more than 2.59, the clock frequency is 2.6 GHz *

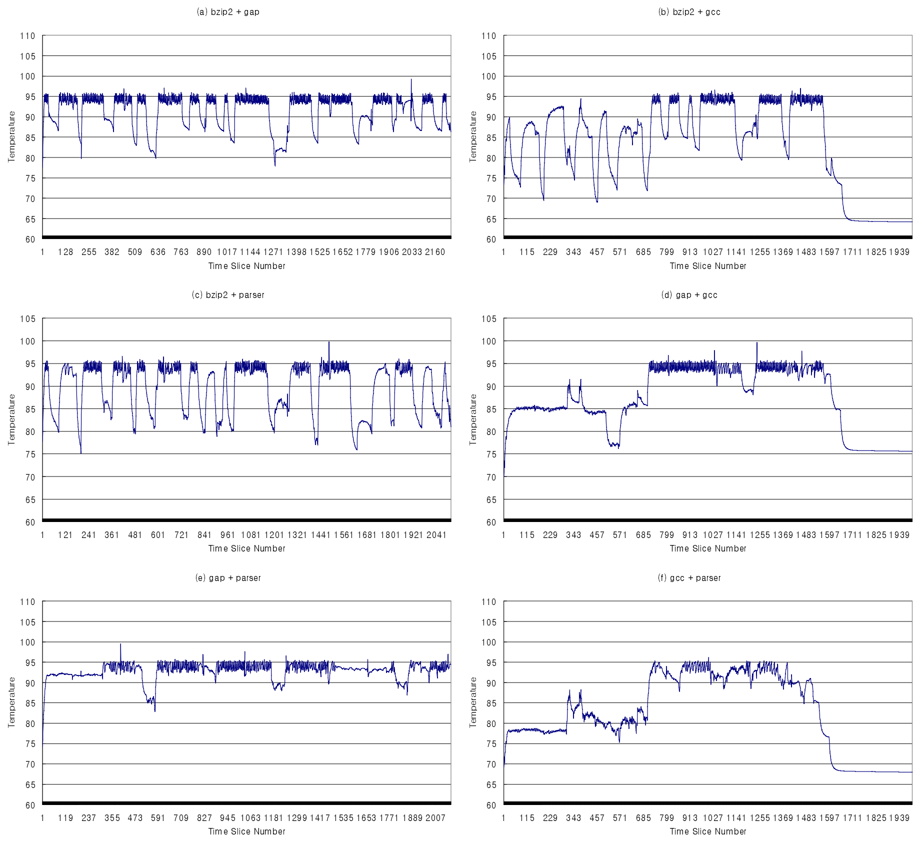

Fig. 2. Temperature changes (w/ DFS using thermal sensors)

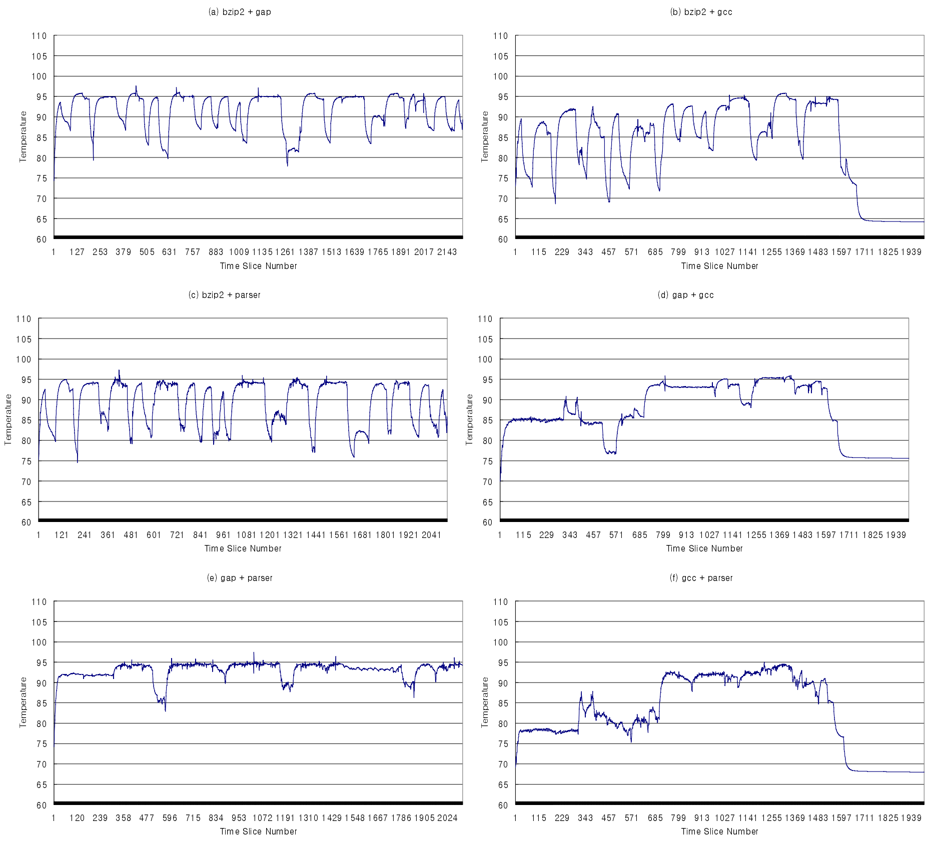

Fig. 3. Temperature changes (w/DFS using performance counters)

(2.59/(previous IIPC)). Otherwise, the frequency is 2.6 GHz (full speed). Thus, the fluctuation around 95 Celsius is less severe, compared to the DFS using thermal sensors.

As explained in the Section 3, using performance counters can make it possible to foresee the temperature tendency in advance. Accordingly, the proposed technique decreases the frequency early when the temperature goes up, which reduces the spikes around 95 Celsius.

5.2 More Details of Temperature Changes

Figure 4 presents the ratio of times when the actual temperature is over the threshold temperature. Both DFS techniques dramatically reduce the thermal violations. Sometimes the DFS using the performance counters performs better and sometimes does not. At least, we can say that the DFS using the performance counters is comparable to the DFS using the thermal sensors.

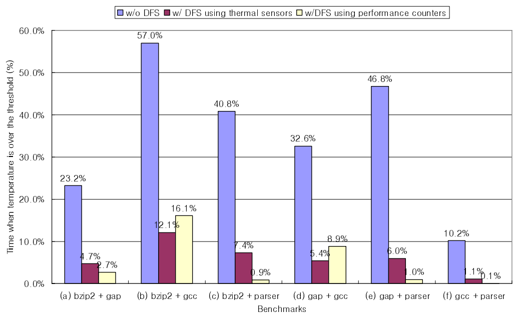

Fig. 4. Ratio of times when the actual temperature is over the threshold value (95 Celsius)

Figure 5 shows the average temperature difference between the actual temperature and the threshold value, when the actual temperature is over the threshold value. Though the temperature violation ratios in Figure 4 are not negligible, the average temperature excesses are significantly reduced. The average values of the temperature excesses in Figure 5 are 0.37 and 0.40 degree, on average, for the DFS using thermal sensors and the DFS using performance counters, respectively.

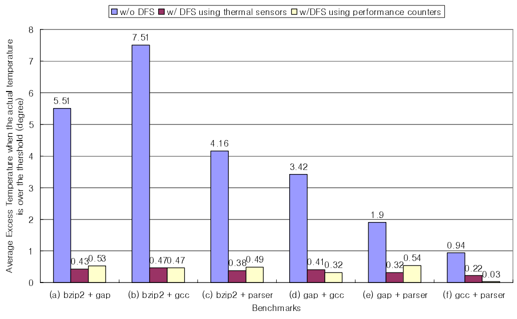

Fig. 5. Average temperature difference between the actual temperature and the threshold value (95 Celsius)

Figure 6 shows the maximum temperature when the actual temperature is over the threshold value. We can notice that the DFS using performance counters always outperforms the DFS using thermal sensors. The DFS using performance counters more accurately forecasts the temperature by referencing to the IIPC, which prevents the spikes. However, the DFS using thermal sensors can not predict future temperature. Thus, the temperature continues to go up even with the DFS, because the power

5.3 Performance

The tasks in this experiment are not periodic, in other words, which is not predictable. Thus, we should sacrifice the performance to sustain the temperature under the threshold value. If more aggressive DFS technique were adopted, the number of thermal violations would be decreased. As the number of thermal violations decreases, the performance is naturally degraded. For example, suppose that one technique sets the threshold value to 90 Celsius and the other sets it to 100 Celsius. The former has less thermal violation and more performance degradation. For a fair comparison, we should check that the both techniques are similarly aggressive. If the proposed DFS using performance counters performed much worse than the DFS using thermal sensors, the experiment would not be fair.

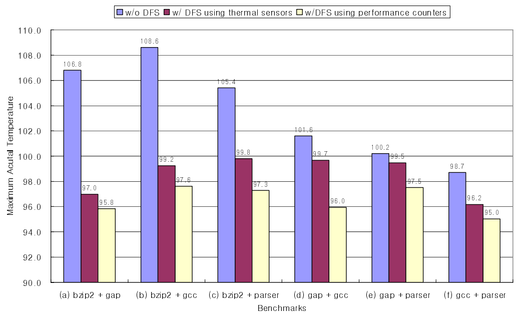

Fig. 6. Maximum temperature for each technique consumed in the past should be dissipated, resulting in higher maximum temperature

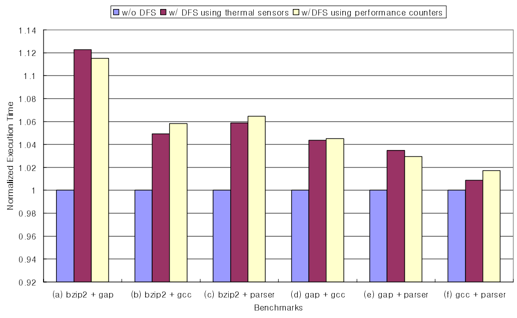

Fig. 7. Execution time normalized to the no DFS

Figure 7 shows the execution time normalized to the no DFS. The relative execution time, compared to the no DFS, only depends on the benchmarks' characteristics, themselves. The importance lies in the relative execution time between the DFS using thermal sensors and the DFS using performance counters. As shown in Figure 7, it is hard to say which technique is better in terms of performance, which implies two techniques are similarly aggressive, in the perspective of thermal control.

6 Conclusions and Future Works

Uneven activity from one functional block to another, results in localized hotspots that may move over time. Thus, accurate thermal monitoring therefore requires lots of thermal sensors. This may be too costly, because precise CMOS thermal sensors are expensive in terms of area and power. As an alternative, we can use performance counters and regression analysis.

In this paper, we show that the DFS using performance counters is comparable to (sometimes better than) the DFS using thermal sensors. The DFS using performance counters only have to utilize the performance counters that are already embedded in most commercial microprocessors. Especially, after fabrication, when a microprocessor or an SOC (System On Chip) turns out to have localized hotspots that are not covered by CMOS thermal sensors, the proposed technique using performance counters can be a cost-effective solution. Though we used the temperature from the Hotspot [13][14] for regression analysis, the temperature from more accurate circuit-level thermal simulations can be used for regression analysis, which leads to more efficiency.

We only concentrated on the integer register file. However multiple functional blocks can be monitored and controlled using performance counters, since different clock frequencies might be assigned to different functional blocks. In this paper, we freely change the frequency but experiments with discrete frequencies would be interesting. We only examined the scheduling efficiency only with the DFS, since the DVS is not so reliable due to technology scaling [6] and it has more timing overhead [11]. The alternative to the DFS is clock gating to cool down the localized hotspots.

Acknowledgements

This works was supported by a Korea University Grant.

References

1. F. Bellosa, A. Weissel, M. Waitz, and S. Kellner. Event-Driven Energy Accounting for Dynamic Thermal Management. In Proceedings of COLP 2003, Sep. 2003.
2. D. Brooks and M. Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. In Proceedings of HPCA’01, Jan. 2001.
3. S. W. Chung and K. Skadron. Using On-Chip Event Counters for High-Resolution, Real-Time Temperature Measurements. Proceedings of ITHERM’06, June 2006.
4. M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas. A Framework for Dynamic Energy Efficiency and Temperature Management. In Proceedings of Micro’00, 2000
5. C. Isci and M. Martonosi. Runtime Power Monitoring in High-End Processors: Methodology and Empirical data. In Proceedings of Microarchitecture (Micro’03), Dec. 2003.
6. V. Narayana and Y. Xie. Reliability Concerns in Embedded System Designs. IEEE Computer, vol. 39 no. 1, pp.118-120, Jan. 2006.
7. K.-J. Lee and K. Skadron. Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors. In Proceedings of the Workshop on High-Performance, Power-Aware Computing (HP-PAC), April 2005.
8. K.-J. Lee and K. Skadron. Analytical Model for Sensor Placement on Microprocessors. In Proceedings of the IEEE International Conference on Computer Design (ICCD’05), Oct. 2005.
9. Merkel, F. Bellosa, and A. Weissel. Event-Driven Thermal Management in SMP Systems, In Proceedings of the Second Workshop on Temperature-Aware Computer Systems (TACS’05), June 2005.

10. M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-Run : Leveraging SMT and CMP to Manage Power Density Through the Operating System. In Proceedings of International Conference on Architectural Support for Programming Language and Operating System (ASPLOS'04), Oct. 2004.
11. E. Rotem, A. Naveh, M. Moffie, and A. Mendelson. Analysis of Thermal Monitor Features of the Intel Pentium M Processor. In Proceedings of the Second Workshop on Temperature-Aware Computer Systems (TACS'04), June 2004.
12. K. Skadron, T. Abdelzaher, and M. R. Stan. Control-Theoretic and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management. In Proceedings of HPCA'02, Feb. 2002.
13. K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayana, and D. Tarjan. Temperature-Aware Microarchitecture. In Proceedings of the 30th International Symposium on Computer Architecture (ISCA'03), June 2003.
14. K. Skadron, M. Stan, K. Sankaranarayana, W. Huang, S. Velusamy, and D. Tarjan. Temperature-Aware Microarchitecture: Modeling and Implementation. ACM Transaction on Architecture and Code Optimization. Vol. 1, No. 1, March 2004, pp. 94-125.
15. B. Sprunt. Pentium 4 Performance-Monitoring Features. IEEE Micro, 22(4), Jul/Aug 2002.
16. J. Srinivasan and S. V. Adve. Predictive Dynamic Thermal Management for Multimedia Applications. In Proceedings of International Conference on Supercomputing (ICS'03), June 2003.
17. Weissel and F. Bellosa, Dynamic Thermal Management for Distributed Systems. In Proceedings of the First Workshop on Temperature-Aware Computer Systems (TACS'04), June 2004
18. Apple Computer. Quad G5 2.5Ghz Processors. Available in http://homepage.mac.com/ thunderaudio PhotoAlbum11.html.
19. J. Citaerlla. The Intel PIV's Thermal Diodes. Available in http://www.overclockers.com/ artocles 517.
20. HP Corporation, Intel Corporation, Microsoft Corporation, Phoenix Tech. Ltd., and Toshiba Corporation, "Advanced Configuration and Power Interface Specification",. Available in http://www.acpi.info /DOWNLOADS/ACPIspec30.pdf, September 2004
21. Intel Corportation. IA-32 Intel Architecture Software Developers Manual. Vol. 3: System Programming Guide, 2004.
22. Intel Pentium 4 Northwood Die Photo. Available in http://www.chip-architect.com/news/ 003_04_20_Looking_at_Intels_Prescott_part2.html
23. Intel Pentium 4 Technical Documents. Available in http://www.intel.com/design/Petium4/ documentation.html
24. Intel Corporation. Thermal Zone Information. Available in http://support.intel.com/ support/motherboards/desktop/sb/CS-12552.htm
25. RCN Corporation. Processor Electrical Specifications, Available in http://users.erols.com/ chare/ elec.htm
26. Standard Performance Evaluation Corp.. Available in http://www.specbench.org.
27. B. Sprunt. Brink and Abyss Pentium 4 Performance Counter Tools for Linux. Available in http://www.eg.bucknell.edu/bsprunt/emon /brink_abyss.