Active Bank Switching for Temperature Control of the Register File in a Microprocessor

Kimish Patel, Wonbok Lee, Massoud Pedram
Department of Electrical Engineering
University of Southern California
Los Angeles CA 90089
{Kimishpa, wonbokle, pedram}@usc.edu

ABSTRACT
An effective thermal management scheme, called active bank switching, for temperature control in the register file of a microprocessor is presented. The idea is to divide the physical register file into two equal-sized banks, and to alternate between the two banks when allocating new registers to the instruction operands. Experimental results show that this periodic active bank switching scheme achieves 3.4℃ of steady-state temperature reduction, with a mere 0.75% average performance penalty.

Categories and Subject Descriptors
B.7.2 [**Hardware**]: Design Aids
General Terms:
Design, Reliability
Keywords
Thermal model, Temperature-aware Design, Register File

1. INTRODUCTION

Peak power dissipation and the resulting temperature rise have become a limiting factor to microprocessor performance and a significant component of its cost. Expensive packaging and heat removal solutions are needed to achieve acceptable substrate and interconnect temperatures in high-performance microprocessors. Current thermal solutions are designed to limit the peak processor power dissipation to ensure its reliable operation under worst-case scenarios. However, the peak processor power and ensuing peak temperature are hardly ever observed. Dynamic thermal management (DTM) has been proposed as a class of micro-architectural solutions and software strategies to achieve the highest processor performance under a peak temperature limit.

Most DTM methods are reactive due to the complex nature of temperature variation in a processor; when a certain triggering temperature is reached, DTM mechanisms become operational. For example, in [1], Skadron et al. introduced a number of DTM methods such as temperature driven frequency scaling, localized toggling and computation migration to spare hardware units. The same authors presented a hybrid DTM technique that combines fetch gating and dynamic voltage scaling (DVS) in [2]. Reference [3] described a feedback control theory based DTM method, which determines the aggressiveness of the DTM methods based on the distance of triggering temperature from the emergency temperature. Recently, reference [4] introduced a predictive DTM method for multi-media applications whereby instruction window resizing and switching among active functional blocks were utilized to achieve the desired temperature control. All of these methods characterize and/or predict the thermal behavior of a processor typically on a functional block basis, calculate the power density of functional blocks within a fixed time period, and apply their temperature control policies as needed.

It is known that the register file is the hottest block in a modern microprocessor chip [1][4]. As such, full-chip DTM methods, such as fetch-toggling and instruction cache throttling (where the number of fetched instruction is reduced as needed), [5][6], have been utilized to control this register file temperature. A DTM method specifically targeted toward temperature control in the register file was presented in [7]. This method, called activity migration, is quite effective, albeit it has a large area overhead.

In this paper, we present a DTM method that targets and effectively reduces temperature of the register file. Our idea is based on the observation that the register file is not fully utilized over a program's execution, i.e., the lifetime of registers/operands are short such that we only need a rather small number of physical registers to be active during most of the cpu cycles. Therefore, by introducing two equal-sized banked structures in the physical register file (one active bank and another sleep bank) and alternately using these two banks, temperature of both banks can be reduced while little performance penalty is incurred. This is similar to what the authors proposed in [7] except that we do not introduce a redundant register file structure. Instead we divide the existing register file structure into two banks and alternate between the two while monitoring and respond to register file utilization of the application program. In addition to area savings, our method also avoids processor-wide performance penalty in the sense of IPC degradation.

2. ACTIVE BANK SWITCHING BASED DTM

2.1 Register File Utilization

Many 32-bit instruction set architectures (ISA) are designed to have 32 architectural registers although modern superscalar processors have more than 32 physical registers. This discrepancy is handled by register renaming, which assigns architecture registers to physical registers while considering data/control dependencies among the instructions. In practice, not all of the physical registers are used all the time. In [8], Tran et al. showed that physical register usages are typically in the range of 40% to 60%. This low utilization phenomenon arises mainly from the dependencies among instructions in the instruction window.

This work was sponsored in part by a grant from the CISE directorate of the National Science Foundation.

GLSVLSI'07, March 11-13, 2007, Stresa-Lago Maggiore, Italy.

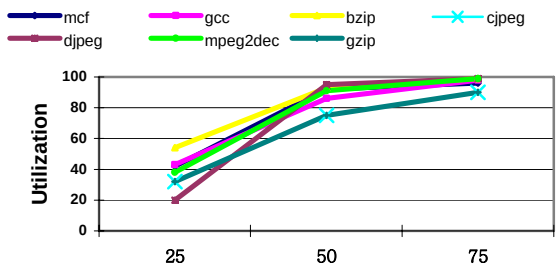

Figure 1 Physical Register File Utilization

Figure 1 shows the utilization of physical registers according to our own simulation data (the simulation methodology will be explained later). Here the x-axis represents the number of physical registers that are being utilized as a percentage of the original register file size (64), whereas the y-axis represents the utilization ratio as a percentage of total execution time. For example, for *mcf*, 25% of physical registers are actually in use during 42% of the execution time. Note that on average for about 90% the time, less than a half of the physical registers (32) are actually allocated.

Figure 2 shows the performance penalty if the register file size is cut in half (from 64 to 32). Notice that for *djpeg* even though for 25% of the execution time more than 32 registers are used, the respective performance penalty is only 3%. This is because although a new instruction is dispatched and allocated to a physical register, much of the time this instruction is not issued and executed due to the data dependencies among instructions.

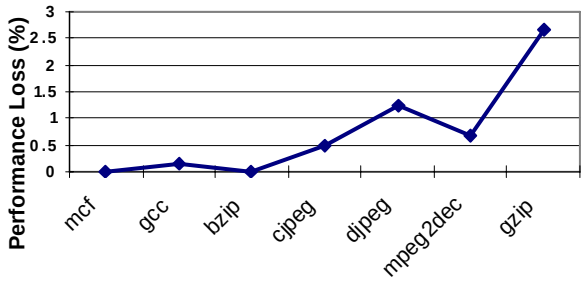

Figure 2 Performance Penalty When the Register Bank Size Is Cut in Half

2.2 Periodic Bank Switching: The Idea

Based on the above observation, we propose to divide the register file into two equal-sized banks and use only one bank at a time, i.e., the number of physical registers available at any time is one half of the original count and registers are allocated from one of these two banks. Here we designate the active bank as a *primary bank* and the other one as a *secondary bank*. Registers are allocated first from the primary bank and only if the primary bank is full, the allocation is done from the secondary bank. Since only a small number of physical registers are used during most of the execution time of typical programs, the duration for which the secondary bank will be in-use is relatively small. When bank switching occurs, there might still be some references to the non-active bank. However, these pending references will be relatively small compared to the number of references to the active bank.

2.3 Thermal Zones and Thermal Gradients

We have carried out detailed analysis of the temperature regions in terms of thermal gradients and classified them into two zones: 1) Fast Temperature Rise (FTR) zone: The rising thermal gradient is higher than the falling thermal gradient i.e., the temperature rises faster than it falls (when the chip is allowed to cool off). 2) Fast Temperature Fall (FTF) zone: The falling thermal gradient is equal to or higher than the rising thermal gradient i.e., the temperature drops faster than it rises. Note that the DTM methods are most effective in the FTF zone. Based on our simulations (cf. Figure 3), the FTF zone is above the FTR zone. This is fortunate because the temperature profile of a microprocessor chip is such that DTM techniques become more effective as the chip temperature rises.

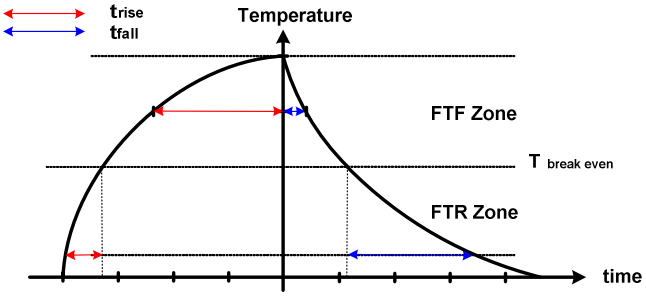

Figure 3 Thermal Gradients and Temperature Zones

Depending on the type of packaging and cooling solutions, the chip's *critical temperature* (CT, the temperature beyond which chip may not function correctly or may even get damaged) may lie in any of these two zones. In the absence of a DTM technique, any application program running on a microprocessor chip will give rise to a *steady-state temperature* (ST) depending on the program behavior e.g., in terms of its CPI. The goal of our proposed DTM method is to *minimize the chip ST while meeting a performance loss constraint*. If the ST lies in the FTF zone, then the DTM methods tend to work very well and the new ST of the chip will be significantly lower. Otherwise, the DTM techniques are expected to be less effective.

2.4 Thermal Model

To mathematically support the periodic active bank switching idea, we use a thermal model developed by Skadron et al. in [3]. Based on this model, the temperature increase in the processor is represented by:

$$\Delta T = \left(\frac{P}{C_{th}} - \frac{T_{old}}{R_{th} \cdot C_{th}}\right) \cdot \Delta t \qquad (1)$$

where Δt is a time interval, P is the average power dissipated in an interval, R_{th} is a thermal resistance, C_{th} is a thermal capacitance and T_{old} is the initial temperature of a time period, respectively. After a time interval, the new temperature becomes:

$$T_{new} = T_{old} + \Delta T \qquad (2)$$

Let $t_{initial}$ and t_{final} denote two instances of time (and their difference be denoted by Δt), respectively. Then, the rising thermal gradient with respect to time is represented as:

$$\frac{\Delta T_r}{\Delta t} = \left(\frac{P}{C_{th}} - \frac{T_{old}}{R_{th} \cdot C_{th}}\right) \qquad (3)$$

Hence when the active bank is switched, the new active bank's temperature rises according to equation (3). Whereas the other bank experiences a temperature drop, and this temperature drop

follows:

$$\frac{\Delta T_f}{\Delta t} = (-\frac{T_{old}}{R_{th} \cdot C_{th}}) \quad (4)$$

If $\frac{\Delta T_f}{\Delta t} \geq \frac{\Delta T_r}{\Delta t}$ (i.e., we are operating in the FTF zone), then active bank switching will be quite effective in reducing the temperature. The *breakeven temperature* (BT or T_{BE}) above which the active bank switching will be beneficial is obtained by solving the following equation:

$$\frac{\Delta T_f}{\Delta t} = \frac{\Delta T_r}{\Delta t} \Rightarrow (\frac{P}{C_{th}} - \frac{T_{BE}}{R_{th} \cdot C_{th}}) = (\frac{T_{BE}}{R_{th} \cdot C_{th}}) \Rightarrow T_{BE} = \frac{1}{2} \cdot P \cdot R_{th} \quad (5)$$

Consider the case where the ST is above the BT. Conceptually, we would like to identify a *trigger temperature* (TT) such that BT ≤ TT ≤ ST and then switch between the two banks as soon as the temperature of the active bank is about to go above the TT. However, in practice, we have found it to be unnecessary to identify such a trigger temperature level. More precisely, a simple DTM policy where we regularly (i.e., at fixed timing intervals) switch between the primary and secondary banks is sufficient. We have found that a fixed interval of 10M CPU cycles is adequate for our purposes and that the overall reduction in ST is not sensitive to the exact length of this interval.

Note that the actual falling thermal gradient in the sleep bank is smaller than equation (4) since some of the registers previously mapped to this bank are alive for certain cycles even after switching. Similarly, the actual rising thermal gradient in the newly active bank is smaller than equation (3) since some of the registers previously mapped to the sleep bank are alive and accessed from that bank for certain cycles. However, the idea is still the same.

2.5 Overhead

It is expected that the banked structure in physical register file needs extra control logics and the renaming logic need to be changed to allocate new registers from the active bank only. However, these area penalties are much smaller than those for the activity migration method, which duplicates the entire register file. Furthermore, the periodic active bank switching scheme does not have self-producing performance penalty as is the case for the activity migration method since we do not need to transfer the content of registers from one bank to the other.

3. EXPERIMENTAL RESULTS

3.1 Micro-architecture Simulation Data

Table 1 reports the architectural configuration that was assumed in our simulations.

Table 1 Micro-Architecture Parameters

Main Memory Latency	32 cycles
L1 I/D Cache	32KB 32-way 32Byte block
I/D-TLB	4-way 1K entries 32 cycles miss latency
Branch Predictor	Bimodal 128 Table
Functional Units	4 INT. ALU, 1 INT. MULT/DIV 4 FP ALU, 1 FP MULT/DIV
RUU/LSQ size	8/8
Instruction Fetch Queue	8

3.2 Methodology

For the experiments, we integrate SimpleScalar [9], Wattch [10] and Hotspot [11] into one simulator The temperature data is generated every 50K cycles and the initial/ambient temperatures are set by 60/45℃, respectively. For the floor-plan in our thermal simulation, we obtain a 2.6GHz Pentium IV 130nm floor-plan from [15], estimate/extract the area information for each of the functional unit, and provide this information to our integrated simulator.

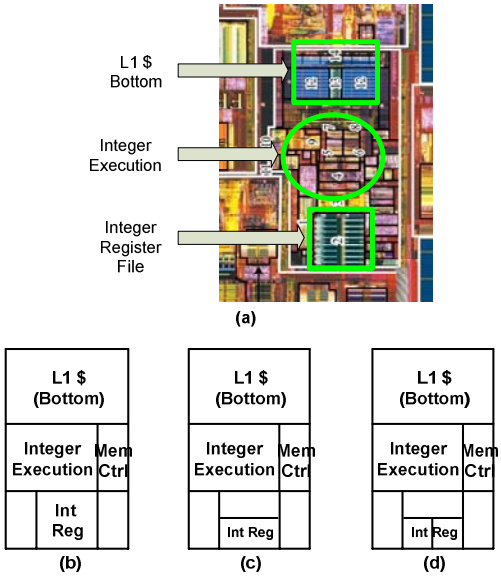

Figure 4 Detailed Floor-plan for the Register File

Figure 4 (a) shows the 'integer execution core' part of the tagged die-photo obtained from [15]. As shown, the register file area is in reality smaller than in the original floor-plan and is roughly half of the original size. Hence, we divide the original register file area into half to match our floor-plan with more detailed description (cf. Figure 4 (a)). We position this half sized register file in the center of the register file area and the surrounding area is kept empty (cf. Figure 4 (b)). Since the original register file area corresponds to the size for 128 registers whereas in our experiments the register file has a size of 64, we further divide this area into half (cf. Figure 4 (c)). For the banked structure, we further cut the original register file area (cf. Figure 4 (c)) into half to denote two banks of size 32 (cf. Figure 4 (d)).

Our simulation setup is as follows. For the first 200K cycles of each benchmark program run, we obtain the typical power figure for the register file (along with other functional units). Next, we use this power figure to mimic the thermal simulation without actually simulating the application by continuously feeding this typical power value to each functional unit. The thermal simulation is carried out in order to find the steady-state temperature for the register file. Once the steady-state is found, we resume the actual thermal simulation of the application.

For the test applications, we used SPEC2000INT benchmarks [12] with reference/train input files, Mediabench program [13] and MPEG-2 decoder program [14]. Input files for mediabench are custom made, input file for the MPEG-2 decoder program is obtained from [14] and the input files of all programs are shown in Table 3. Each program is compiled with the PISA compiler using default optimization option. For the test platforms, two Linux machines were used: Intel Pentium IV 2.8GHz with

512MB memory and Intel Pentium IV 1.8GHz with 2GB memory.

3.3 Experimental Results

At first, we ran each application in a monolithic physical register file of size 64 and record the ST. Next, we ran the application with a banked register file with active bank switching. In a banked register file, the total number of physical registers is 64 but they are divided into two banks, each of size 32.

Table 2 Steady-State Temperature and IPC

Program	Steady-state Temp (℃)		Thermal reduction (℃)	IPC
	Monolithic RF (64)	Banked RF (2*32)		
mcf - inp.in in train	68.0	66.7	1.3	0.7707
gcc - input.source in ref	76.5	73.7	2.8	1.2748
bzip - input.log in ref	78.2	75.0	3.2	1.5022
gzip - input.log in ref	81.7	77.5	4.2	2.1069
cjpeg - custom.gif	83.0	79.0	4.0	2.2553
mpeg2dec-hhilong.m2v	82.0	77.7	4.3	2.2729
djpeg - custom.jpg	82.0	77.0	5.0	2.3825

In Table 2, the difference of steady-state temperatures between the monolithic and the banked register file is shown (cf. the thermal reduction column). The average steady-state temperature reduction of the active bank switching scheme is 3.4℃. Note the relationship between the steady-state temperature and the IPC of each program: As a program workload increases, its steady-state temperature increases as well.

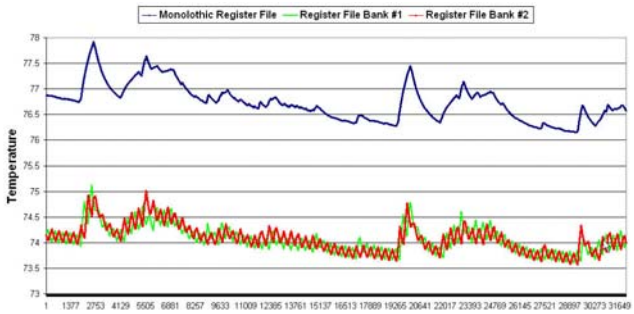

Figure 5 An Example of Thermal Behaviors in gcc

Figure 5 shows the steady-state temperature behavior of the gcc program. The upper thermal curve corresponds to the monolithic register file and the lower two thermal curves correspond to each bank in the banked register file. Compared to the upper curve, note that (a) the application program's thermal behavior is maintained in the lower curves, and (b) the periodic active bank switching is observed between the two lower curves. Note also that two lower curves lay one upon another with very small thermal differences. Each point in the x-axis corresponds to 10M cycles.

Table 3 shows the register file utilization in terms of percentage of total execution cycles spent using 1/4, 1/2, 3/4 of the register file, respectively. The performance penalties reported correspond half sized (32) register file. Note that low performance penalty is due to the lower utilization of register file.

Table 3 Register File Utilization and Performance

Program	Register File Utilization (%)			Performance Penalty (%)
	25%	50%	75%	
mcf	42	92	96	0
gcc	43	86	98	0.16
bzip	54	93	99	0
djpeg	20	95	99	0.47
cjpeg	32	75	90	1.25
mpeg2de	38	91	99	0.69
gzip	32	75	90	2.68

4. CONCLUSION

We presented an effective steady-state temperature reduction method by adopting a banked structure in the register file. In our scheme, only one bank is active at a time and we keep switching between the two available banks. With banking, we achieve a sizeable steady-state temperature reduction with a small performance penalty.

5. REFERENCES

[1] K. Skadron et al., "Temperature-Aware Micro-architecture," Proc. of *Int'l Symp. on Computer Architecture*, Jun. 2003.

[2] K. Skadron, "Hybrid Architectural Dynamic Thermal Management," Proc. of *the Design Automation and Test in Europe*, 2004.

[3] K. Skadron et al., "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," Proc. of *the Int'l Symp. on High-Performance Computer Architecture*, 2002.

[4] J. Srinivasan, S. V. Adve, "Predictive Dynamic Thermal Management for Multimedia Application," Proc. of *Int'l Conference on Supercomputing*, Jun. 2003.

[5] D. Brooks et al., "Dynamic Thermal Management for High-Performance Microprocessors," Proc. of *Int'l Symp. on High-Performance Computer Architecture*, 2001.

[6] H. Sanchez et al., "Thermal Management System for High Performance PowerPC Microprocessor," Proc. of *IEEE Computer Society Int'l Conference*, 1997.

[7] S. Heo, K. Barr, K. Asanovic, "Reducing Power Density through Activity Migration," Proceedings of *Int'l Symp. on Low Power Electronics and Design*, Aug. 2003.

[8] L. Tran et al., "Dynamically Reducing Pressure on the Physical Register File through Simple Register Sharing," Proc. of *the Int'l Symp. on Performance Analysis of Systems and Software*, 2004.

[9] Simplescalar at: http://www.simplescalar.com

[10] D. Brooks et al., "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," Proc. of *the Int'l Symp. on Computer Architecture*, Jun. 2000.

[11] HotSpot at: http://lava.cs.virginia.edu/HotSopt/

[12] SPEC2000INT benchmark at: http://www.spec.org/cpu

[13] Mediabench at: http://euler.sluedu/~fritts/mediabench

[14] MPEG-2 Programs at: http://www.mpeg2.de/video/

[15] Pentium IV floor-plan at: http://www.chip-architect.com