RELOCATE: Register File Local Access Pattern Redistribution Mechanism for Power and Thermal Management in Out-of-Order Embedded Processor

Houman Homayoun[1], Aseem Gupta[2], Alex Veidenbaum[1],
Avesta Sasan (M.A. Makhzan)[3], Fadi Kurdahi[3], and Nikil Dutt[1]

[1] Department of Computer Science, University of California, Irvine, CA, USA
[2] Freescale Semiconductor Inc. Austin, TX 78729, USA
[3] Department of Electrical Engineering and Computer Science, University of California, Irvine, CA 92697-3435, USA
{hhomayou,aseemg,mmakhzan,alex.veidenbaum,kurdahi,dutt}@uci.edu

Abstract. In order to reduce register file's peak temperature in an embedded processor we propose RELOCATE: an architectural solution which redistributes the access pattern to physical registers through a novel register allocation mechanism. RELOCATE regionalizes the register file such that even though accesses within a region are uniformly distributed, the activity levels are spread over the entire register file in a deterministic pattern. It partitions the register file and uses a micro-architectural mechanism to concentrate the accesses to a single or a subset of such partitions through a novel register allocation mechanism. The goal is to keep some partitions unused (idle) and cooling down. The temperature of idle partitions is further reduced by power gating them into destructive sleep mode to reduce their leakage power. The *redistribution mechanism* changes the active region periodically to modulate the activity within the register file and prevent the active region from heating up excessively. Our approach resulted in an average reduction of 8.3°C in the register file's peak temperature for standard benchmarks.

Keywords: Register file, Temperature, Power, Local Activity Redistribution, Out of Order Embedded Processor.

1 Introduction

Continued CMOS process technology scaling has led to designs of much more complex embedded processors with significantly higher computational power. The high level of integration in SoC designs today has, however, led to correspondingly higher power densities (Watt per mm2) which in turn lead to higher operating temperatures. High operating temperatures have many unfavorable consequences: (i) Increased probability of timing violations because of higher signal propagation delay and switching time, (ii) Reduced lifetime because of phenomena such as electromigration, (iii) Lower clock frequencies of designs because of higher device and interconnect delays, (iv) Increased

Y.N. Patt et al. (Eds.): HiPEAC 2010, LNCS 5952, pp. 216–231, 2010.

leakage power due to super-linear relationship with temperature, (v) need for expensive cooling mechanisms, such as fans, and (vi) Overall increase in design effort and cost.

Components within a processor operate at different temperatures as function of their circuit design and activity levels. One of the hottest components is the integer register file. Recent embedded processors such as MIPS-74K or IBM PowerPC 750FX, use a large register file to support out-of-order execution. The register file is accessed every cycle, unless the processor is stalled, for both reads and writes. In addition, multiple instruction issue further increases the number of register file accesses per cycle. Thus in these processors the register file is one of the most active components which also makes it the hottest unit. Numerous academic and industrial papers have pointed out this fact [1,15,19]. For example, Koren et al. [19] show that the register file is hotter by as much as 20°C than any other block in a processor. This peak temperature determines the "design temperature", i.e., the reference temperature, which is used to characterize the performance of the design. Therefore, there is a critical need to reduce the peak operating temperature of the register file.

One general approach to reduce temperature of a given processor unit, including the register file, is to reduce its activity level by activity migration. This approach requires a replicated unit to be available in the system [16]. Once the unit reaches a critical temperature, its activity is migrated to the replicated unit and the hot unit becomes idle, allowing it to cool down.

While activity migration is effective in reducing the temperature of the register file it requires a replicated register file [16], which is expensive and complex (30% area overhead [16]). It can also lead to performance degradation (3~12% [16]), because of migration overhead. For instance, in [16] copying registers from a register file to its replica requires the pipeline to be drained and multiple additional reads and writes to be performed.

This paper introduces the idea of local activity migration to manage register file temperature in embedded out-of-order processors. It proposes a REgister file LOCal Access paTtern rEdistribution mechanism (or, in brief, RELOCATE). RELOCATE redistributes the access to physical registers through a novel register allocation mechanism. By "local" we mean that a replicated register file is not required; instead the register file access activity is "migrated" or redistributed from one part of the register file to another. This is accomplished without a noticeable impact on register usage and has no performance degradation.

RELOCATE uses a register file partitioned into multiple regions. This partitioning allows the RF access activity to be distributed in a non-uniform pattern over the regions. This pattern is such that the regions accessed are spatially and temporally apart allowing the opportunity for other regions to cool-down. The goal is to keep some regions unused and cooling down while other regions are active. This requires a new micro-architecture because in current micro-architectures the accesses are fairly uniformly distributed over the RF.

RELOCATE is based on the observation that only a small subset of physical registers is used (mapped) at any given time during the course of program execution. Therefore there is room for the migration (redistribution) within the RF itself instead of migrating the activity to a replicated unit. The micro-architectural solution to redistribute the physical registers and their access pattern is based on a novel register renaming mechanism. The new register renamer attempts to allocate new registers

from a given RF region and thus to concentrate the accesses in this region. The renamer partitions the free list to correspond to regions in the register file. After a partition is used for a period of N clocks, the renamer switches to a new partition. Choosing a large enough N (10K cycles) allows the RF regions to cool down if they can be kept idle (not accessed).

Successful local activity migration keeps some RF regions idle and this lack of activity is the reason why these regions can cool down. However, such idle partitions still dissipate leakage power, slowing down their cooling. Our approach further reduces the temperature by power-gating an idle region. This should be doable since an idle region has no accesses.

The experimental evaluation of the proposed mechanism is performed using an integrated architectural/temperature simulator. The results show that a 64-entry physical register file with 4 partitions performs best and achieves an average peak temperature reduction of 8.3°C and 6.9°C for SPEK2K and MiBench benchmarks respectively. The temperature is reduced without any impact on performance with minimal area and hardware overhead.

2 Background

2.1 On-Chip Thermal Behavior

Today's chips are operating at very high temperatures due to high power densities. Within a processor, regions operate at different temperatures due to their varying activity levels. The temperature of a region in a VLSI chip does not depend only on its own power dissipation. At any given time there are two phenomena, that determine the temperature of a region: *spatial* and *temporal*. Due to thermal diffusion, the temperature of a region also depends on the temperature of neighboring regions. This is the spatial phenomenon. Unlike power dissipation, the temperature of a region cannot change very quickly. It can take up to several milliseconds for a region's temperature to rise or cool down. This temporal phenomenon is characterized by the thermal resistance and thermal capacitance of the material. For any thermal management solution to be effective, it must consider both the spatial and temporal phenomena.

2.2 Conventional Register File Organization

An out-of-order embedded processor uses a larger register file with logical to physical register renaming and a dynamic physical register allocation policy. The same approach has been used in high performance superscalar processors such as Alpha 21264 [5] and MIPS R10000 [6].

The pipeline of an out-of-order embedded processors is capable of fetching, decoding, renaming several instructions per processor clock cycle. The processor can also execute and later commit up to as many instructions in each cycle as the issue width. This type of out-of-order multiple-issue processor accesses the register file very frequently. Up to 2*N reads and N writes can be issued to the register file per clock cycle, where N is processor issue width. Thus the register file is one of the most active components in a processor.

Due to frequent accesses the register file is one of the hottest units in an embedded processor. The physical register file is typically designed as a SRAM structure with as many write and twice as many read ports as the maximum number of instructions the processor can issue in each cycle.

The register renamer in these processors is implemented either as a CAM (IBM POWER4 [22] and Alpha 21264[5]) or a RAM (MIPS R10K[6]). This paper discusses issues assuming a RAM based register renaming mechanism similar to the one used in the MIPS R10K processor [6]. However, the techniques proposed in this work also can be applied to a CAM based renamer.

2.3 Activity Migration

One general approach to reduce the temperature of a given processor block is to modulate (or vary) its activity level. One such method is activity migration, where the heat is spread by moving an activity to another block with the same functionality. This technique requires availability of redundant blocks in the system [16]. Once a block reaches a critical temperature, its activity is migrated and it becomes idle allowing it to cool down and reduce its temperature as shown in Figure 1(a). Notice that both temperature increase and decrease are non-linear functions of time.

Activity migration was used in [16] for different units in a processor. It was shown that to benefit the most from this technique the migration period should be significantly smaller than the time constant of the equivalent thermal RC circuit (shown in Figure 1(a)). While short migration periods can result in larger temperature decrease, they incurred large power/performance overhead, as reported in [16].

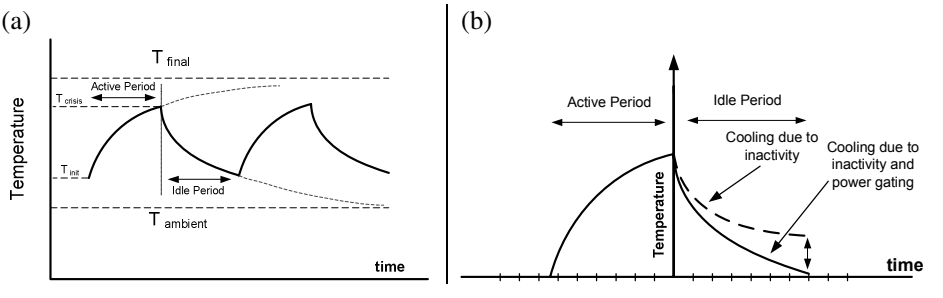

Fig. 1. Thermal benefit of (a) activity migration and (b) combination of activity migration and power gating

2.3.1 AM and Power Gating

Once the activity is migrated completely to a redundant block the base block can be put into a low-power mode to further reduce its temperature. The main reason for this is the reduction in idle leakage power, as described below. The decrease in temperature ΔT is given by:

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

where Δt is the time interval, P is total power dissipation ($P_{total} = P_{dynamic} + P_{leakage}$), T_{old} is the original temperature, and R_{th} and C_{th} are the thermal resistance and capacitance, respectively. From EQ.1 the rate of cooling is faster if P_{total} becomes smaller. This is shown in Figure 1(b) where cooling accelerates due to base block inactivity as well as due to turning off the unused regions. The difference between the two curves (marked *d*) is the additional temperature reduction on account of leakage power saving.

3 Analysis of Register File Operation

In this section we examine the register file access pattern and register file occupancy. This analysis allows us to propose a solution which reduces register file temperature without any performance degradation and at a minimal cost.

3.1 Register File Occupancy

Figure 2 shows the register file occupancy results for the MiBench and SPEC2K integer benchmarks. We observe that for nearly 60% of the time only half of all register file entries are occupied across the MiBench benchmarks. For SPEC2K only half of all register file entries are occupied for about 80% of the time. For 35% of the time only a quarter of the register file is occupied in MiBench benchmarks. This ratio is 60% for SPEC2K benchmarks. Such low register file occupancy raises the question: *why do we need such a large register file in the first place?* To answer this question, consider Figure 3 which shows the performance degradation as a result of using a smaller register file --with 16, 32, and 48 registers, instead of the original size of 64. We observe significant performance degradation across most SPEC2K and MiBench benchmarks. The performance impact is up to 36% and 19% in SPEC2K and MiBench benchmarks respectively when a register file with half its size (32 entries) is used. An even larger performance impact is observed as we shrink the size of the register file further.

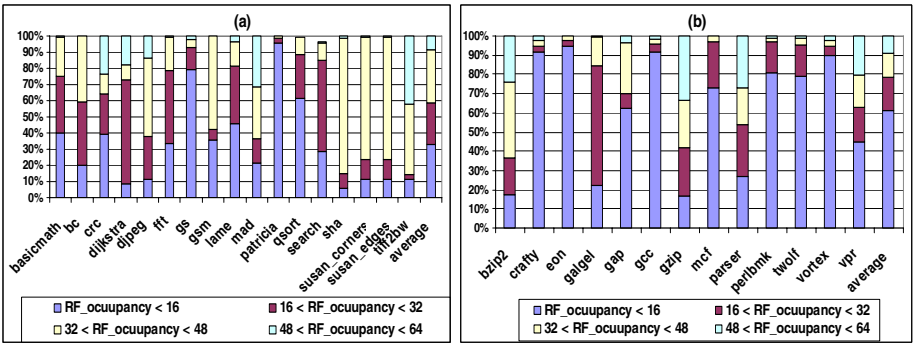

Fig. 2. Register file occupancy results for (a) MiBench and (b) SPEC2K integer benchmarks

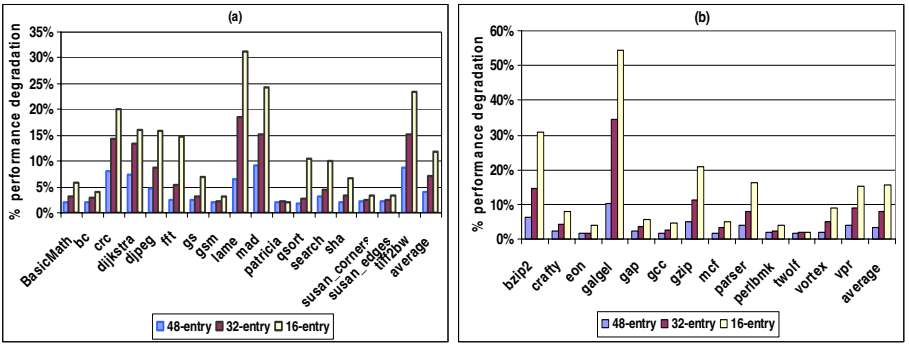

Fig. 3. Performance degradation as a result of using a smaller register file -with 16, 32, and 48 registers for (a) MiBench and (b) SPEC2K integer benchmarks

Thus in spite of low average occupancy, using a smaller register file degrades the performance noticeably. In fact, a smaller register file fills up faster when a long latency operation occurs. For instance, after a load instruction miss in L2 cache it stays on top of the *ROB* and doesn't allow the subsequent instructions to be committed. Therefore the dependent instruction's corresponding physical registers can not be released until the cache miss is serviced. During every cycle the processor fills up the register file with up to 2 physical registers whereas it releases registers at a slower rate. Consequently, the register file occupancy grows until it gets filled completely.

3.2 Register File Access Pattern

In this section we study how accesses to physical registers are distributed. We used coefficient of variation (CV) as a metric to indicate the distribution of accesses to physical registers. The coefficient of variation of accesses to the physical registers is a normalized measure of dispersion of registers access distribution. It is defined as the ratio of the standard deviation (σ) to the mean (μ).

$$CV_{access} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{n}(na_i - \overline{na})^2}}{\overline{na}} \tag{2}$$

where na_i is the number of accesses (read or write) to the physical registers i during a specific period (10K cycles). N is the total number of physical registers.

As shown in Figure 4, most benchmarks have a uniformly distributed access pattern to register file. It was also observed that register file occupancy is low for a large portion of program execution time. *Put together, while only a small number of registers are occupied at any given time, the total accesses are uniformly distributed over the entire physical register file during the course of execution.*

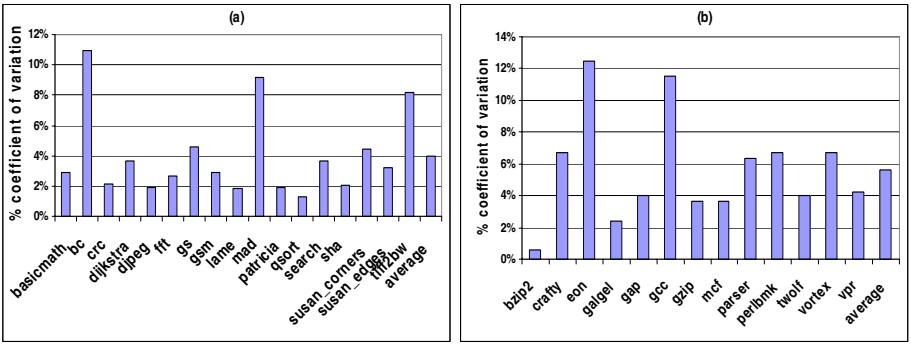

Fig. 4. Coefficient of variation of accesses to physical register file for (a) MiBench and (b) SPEC2K benchmarks

4 RELOCATE: Local Activity Redistribution within a Register File

Activity migration is effective in reducing temperature but it typically requires replicated units. As shown above, only a small subset of physical registers is in use at any given time during program execution. Therefore, instead of migrating the activity of this set of active registers to a replicated register file, one can do it within a single register file (activity redistribution). This paper proposes to partition the (single) register file into multiple regions and to spread the register allocation and therefore their access activity in a deterministic pattern over these regions. Activity of a region will be migrated after a certain amount of time to limit its temperature rise as such idle regions (or partitions) will be cooling off. To further improve the temperature reduction benefit of the proposed activity redistribution, one can power gate the idle regions of the register file.

Let us assume that, on average, 16 registers are being used at any given time. Results in the previous section showed that a small subset of active registers is distributed fairly uniformly over the entire register file during any time interval. The logical view of the baseline register file activity shown in Figure 5(a) represents 16 active registers distributed almost uniformly during a specific timing interval ($4*\tau_c$). This makes it impossible to perform any activity redistribution.

Now let us assume a register file is partitioned into four equal sized regions (partitions) with 16 entries each. Assuming that the 16 active registers are allocated such that they are concentrated in one partition for a certain period (we refer to this as a convergence period or τ_c as shown in Figure 5(b)), other register regions can be kept idle and cooling off. The activity needs to be moved to another partition after τ_c. There are a number of ways to modulate the activity within the register file, e.g., in a round-robin in-order pattern (AP1—AP2 —AP3 —AP4 as shown in Figure 5(b)). The activity is modulated to another region after every convergence period. Note that within a convergence period the activities are uniformly distributed within the active region, similar to the baseline register file. Once the activity is modulated to a new

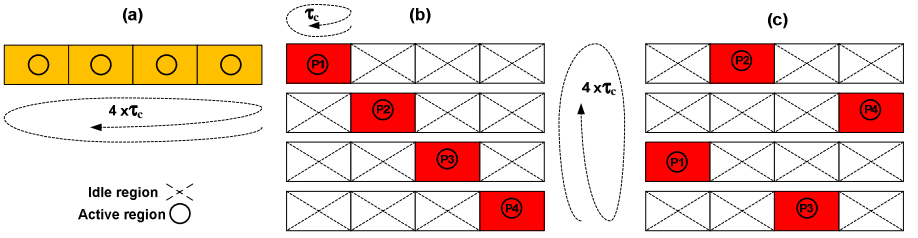

Fig. 5. Examples of register activity distribution (a) baseline, (b) in-order (c) distant patterns

region (active region) all other regions (idle regions) are cooling down. The in-order pattern spaces accesses in time, but there is not enough spatial separation (See Section 2 on Background). An access pattern with spatial as well as temporal separation between active regions would further reduce the temperature. The following redistribution pattern shown in Figure 5(c) can be used to achieve this: AP2 — AP4 — AP1 — AP3 (round-robin distant pattern). When a region becomes active and dissipates power, other regions get an opportunity to cool down. For instance, when AP1 is active, regions AP2 and AP4 are cooling down. AP4 is spatially distant from AP1 while AP2 is temporally distant from AP1.

4.1 The Architectural Mechanism to Support Activity Redistribution

This section introduces an architectural mechanism that attempts to concentrate active physical registers (live registers) in one register file region. This is accomplished through a novel register allocation mechanism that "concentrates" all register allocations during renaming in a given time period (convergence period) in one partition. It partitions the free register list into multiple consecutive partitions and allocates all new physical registers from one partition. Each partition of the free register list corresponds to a certain region in the register file. If there are no more registers in a given partition then the next partition is "activated", even if the time period (convergence period) is not over yet. We refer to the partition(s) currently participating in register renaming as *active partitions* and the rest as *idle partitions*.

The following terms will be used in the rest of the paper:

- Active partition: a register renamer partition which participates in register renaming.
- Idle partition: a register renamer partition which does not participate in renaming.
- Active region: a region of the register file corresponding to a register renamer partition (whether active partition or idle) which has live registers.
- Idle region: a region of the register file corresponding to a register renamer partition (whether active partition or idle) which has no live registers.

The activity concentration and redistribution in the register file occur via two techniques described next.

The concentration mechanism: At any given time registers are allocated from only one partition, referred as the default active partition (DAP), for instance P1 in the example of Figure 6. While our goal is to concentrate all live physical registers in one region, the default partition may run out of free registers before the convergence period is over. Once the free list of the DAP (P1) is empty, the next partition (according to some algorithm) is activated (referred to as additional active partitions or AAP) and is used for register renaming along with the default active partition. In Figure 6 the second and fourth partitions are idle (and power gated) and are activated only when the first and third partitions' free lists are empty, in that order. To facilitate physical register concentration in DAP, if two or more partitions are active and have free registers, allocation is performed in the same order in which partitions were activated. By doing this the default active partition gets the highest priority to allocate physical registers if it has any free registers, thus further concentrating the accesses in the DAP. For instance, if AAPs P2 and P3 were activated in this order, a new register will be allocated from P2.

The redistribution mechanism: The default active partition is changed once every N cycles (we used N=10K) to redistribute the activity within the register file (according to some algorithm). For instance, one can use a round-robin distant pattern algorithm (P2 — P4 — P1 — P3) to maximize the distance between regions. Once a new default partition (NDP) is selected, all active partitions (DAP+AAP) become idle. While these idle partitions do not participate anymore in register renaming, their corresponding regions in the register file are kept active (powered up) until their active list becomes empty. At this time the corresponding physical registers become idle and are power gated as well. Recall that an idle partition does not participate in register renaming. However, it is possible that a physical register belonging to an idle partition may need to be read as a source register of a scheduled instruction. An active region of the RF corresponding to an idle register renaming partition can be powered down provided that all of its physical registers have been released (no live registers left).

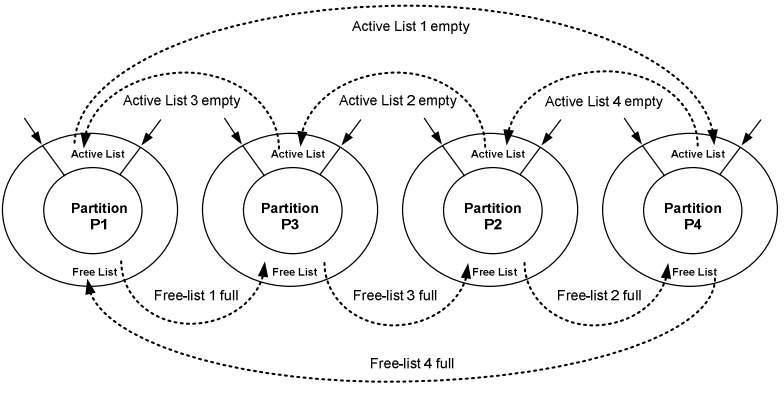

Fig. 6. Partitioned register renaming

We assume a two-cycle delay to wakeup a power gated physical register (the detailed wakeup power/delay overhead will presented later). It should be noted that after this wakeup delay is paid, there is no further effect on the register file access. Thus our technique has no performance penalty in this case. This can be explained as follows: the register renaming occurs in the front end of the microprocessor pipeline whereas the register access occurs in the back end. There is a delay of at least two pipeline stages between renaming and accessing a physical register file. These two cycles allow us to wake up the physical register's region without incurring any performance penalty at the time of access.

The mechanism described above can be implemented by partitioning the circular FIFO free list into multiple smaller size circular FIFOs dynamically adjusting the circular FIFO size. A design proposed in [21] can be used for this, which has no impact on queue performance. A CAM-based renamer [22] can also be similarly partitioned.

5 Experimental Setup

We used the following experimental setup for evaluating this work. We used an extensively modified version of MASE (SimpleScalar 4.0) [13] to model an architecture similar to the MIPS-74K embedded processor [14]. Table 1 describes the processor architecture in detail, which operates at 800 MHz frequency. MiBench and SPEC2K benchmarks were compiled with the O4 flag using the Compaq compiler and executed with reference data sets. The benchmarks were simulated for 1 billion instructions or until completion.

Table 1. Processor Architecture

L1 I-cache	8KB, ,4 way, 2 cycles
L1 D-cache	8KB, 4 way, 2 cycles
L2-cache	128KB, 15 cycles
Fetch, dispatch	2 wide
Register file	64 entry
Memory	50 cycles
Instruction fetch queue	2
Load/store queue	16 entry
Arithmetic units	2 integer
Complex unit	2 INT
Pipeline	12 stages
Processor speed	800 MHz
Issue	Out-of-order

Table 2. RF Design specification

Process	45nm-CMOS 9 metal layers	Register file layout area	0.009mm^2
Operating Modes	Active:R/W Sleep: no data retention	Operating Voltage	0.6V~1.1V
Read Access Cycle	200MHz to 1.1GHz	Access time typical corner (0.9V, 45°)	0.32ns
Active Power (Total) in typical corner (0.9V, 45°)	66mW @ 800MHz	Active Leakage Power typical corner (0.9V, 45°)	15mW
Sleep Leakage Power in typical corner (0.9V, 45°)	2mW	Wakeup Delay	0.42ns
Wakeup Energy per register file row (64bits)	0.42nJ		

To accurately model the register file, an industrial memory compiler was used to generate a dual read and single write port, 64-entry, 64bit single-ended SRAM memory in TSMC 45nm technology. The design including the wordline drivers, the wordline pulse generator circuit, the memory bit-cells and the output drivers is then scaled to model a 4-read, 2-write port SRAM. The register file operates in two modes: an active mode where it can be accessed, and a deep sleep mode where it does not keep the bit-cell data and can not be accessed. Table 2 shows the design specification of this 6-port 64x64 bits SRAM memory. All measurements are done using Spice simulation. The register file access time is 0.32 ns for a typical corner (0.9V and 45°C). The total power in typical corner is 66mw while the active leakage power is 15mw (for the entire register file). The deep sleep data-destructive state leakage is 2mw, almost 86% lower than the active leakage.

The power and delay overhead of transition from low leakage sleep mode to active mode are presented in Table 2. The area overhead for implementing the power gating technique is fairly small, almost 1% of the RF size (using one sleep transistor per register file entry). Total dynamic power of the register file was computed as N*Eaccess/Texec, where N is the total number of accesses (obtained from simulation) and Eaccess is the single-access energy (from Table 2). Leakage power computations are similar, but leakage energy is dissipated on every cycle. If the RF entry is put into sleep mode the sleep leakage power is dissipated, otherwise the active leakage power is dissipated. We used HotSpot [15] to estimate thermal profiles for the register file. We integrated HotSpot into our simulator. The temperature trace is obtained every 10K cycles. Once the temperature is calculated it is reported back to the simulator for the next interval leakage power computation. Since the leakage power is a function of temperature, the power simulator includes a lookup table for leakage power dissipation as a function of temperature in the range from 45°C to 120°C in increments of 5°C.

6 Experimental Results

Figure 7 shows the average register file power reduction over the course of execution of different benchmarks as a result of applying RELOCATE and for different number of RF partitions. We used the experimental setup described in Section 5. We observe that on average there is a reduction of 15% and 25% in the total power of MiBench and SPEC2K benchmarks, respectively. We also observe that many benchmarks, such as patricia (MiBench), eon and vortex (SPEC2K) have a power reduction of about 40%. However, it should be noted that the goal of our work is to reduce the peak temperature of the register file. Usually, the peak temperature is attained as a result of sustained peak power dissipation over a period of time. A workload can have a low overall average power but a very high peak temperature or vice versa due to variation in activity levels. The reduction in the average power does not have a direct correlation with the reduction in the peak temperature of register file. Overall, increasing the number of RF partitions provides more opportunity to capture and cluster unmapped registers to a partition, indicating that the wakeup overhead is amortized for larger number of partitions. There are some benchmarks (highlighted in Figure 7) in which increasing the number of partitions results in smaller power

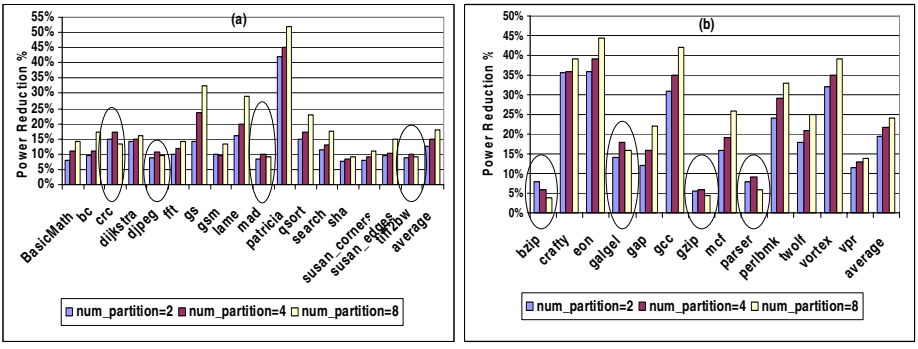

Fig. 7. Register file power reduction for (a) MiBench and (b) SPEC2K integer benchmark

Table 3. Peak temperature reduction for MiBench benchmarks

	base temperature (C°)	temperature reduction for different number of partition (C°)		
		2P	4P	8P
basicMath	94.3	3.6	4.8	5.0
bc	95.4	3.8	4.4	5.2
crc	92.8	5.3	6.0	6.0
dijkstra	98.4	6.3	6.8	6.4
djpeg	96.3	2.8	3.5	2.4
fft	94.5	6.8	7.4	7.6
gs	89.8	6.5	7.4	9.7
gsm	92.3	5.8	6.7	6.9
lame	90.6	6.2	8.5	11.3
mad	93.3	3.8	4.3	2.2
patricia	79.2	11.0	12.4	13.2
qsort	88.3	10.1	11.6	11.9
search	93.8	8.7	9.3	9.1
sha	90.1	5.1	5.4	4.5
susan_corners	92.7	4.7	5.3	5.1
susan_edges	91.9	3.7	5.8	6.3
tiff2bw	98.5	4.5	5.9	4.1
average	92.5	5.6	6.8	6.9

Table 4. Peak temperature reduction for SPEC2K integer benchmarks

	base temperature (C°)	temperature reduction for different number of partition (C°)		
		2P	4P	8P
bzip2	92.7	4.8	3.9	3.1
crafty	83.6	9.5	11	10.4
eon	77.3	10.6	12.4	12.5
galgel	89.4	6.9	7.2	5.8
gap	86.7	4.8	5.9	7.1
gcc	79.8	7.9	9.4	10.1
gzip	95.4	3.2	3.8	3.9
mcf	85.8	6.9	8.7	9.4
parser	97.8	4.3	5.8	4.8
perlbmk	85.8	10.6	12.3	12.6
twolf	86.2	8.8	10.2	10.5
vortex	81.7	11.3	12.5	12.9
vpr	94.6	4.9	5.2	4.4
average	87.4	7.2	8.3	8.2

reduction. In fact in these benchmarks the overall power overhead associated with waking up an idle region is become larger as the number of partition increases. This is in fact due to frequent but ineffective power gating and its overhead as the number of partition increases. Table 3 and Table 4 show the peak temperature reduction result.

We observe that benchmarks from both MiBench and Spec2K show a noticeable reduction in the register file's peak temperature. While increasing the number of partitions in all benchmarks provides more opportunity to capture and cluster unmapped registers, it does not always result in additional temperature reduction. This is especially noticeable in djpeg, mad and tiff2wb (MiBench) and galgel and parser (SPEC2K). In these cases increasing the number of partitions results in larger power density in each partition because RF access activity is concentrated in a smaller partition. While capturing more idle partitions and power gating them *may potentially* result in higher power reduction, larger power density due to smaller partition size results in overall higher temperature. The average reduction in the register file's peak temperature across all the benchmarks is 6.9 °C for MiBench benchmark and 8.2 °C for SPEC2K benchmark. This is very significant in light of the fact that the register file is the hottest block in an embedded processor. The peak temperature of register file determines the design temperature for which the embedded system is designed. Thus the proposed technique can reduce the design temperature by 8°C.

6.1 Additional Benefits of Temperature Reduction

Let us try to quantify the design gains as a result of reduction in the design temperature by 8°C:

(i) The Mean Time To Failure (MTTF) of an electrical interconnect depends on temperature because of electromigration. Depending on the base temperature, a 8°C decrease in the operating temperature can increase the MTTF of an interconnect by up to 2 years.

(ii) A reduction of 8°C in the design temperature means a lower switching delay of transistors. The rated frequency of the design is increased. Based on [20] we estimate that at 45nm technology, a circuit's rated frequency can be increased from 800 MHz to 880 MHz because of the 8°C reduction in peak operating temperature.

(iii) The leakage power has a super linear dependency on temperature. Depending on the process parameters, the leakage power of a cell can be lower by as much as 18% as a result of lowering of temperature by 8°C.

A 8°C reduction also delivers additional power savings since the fan can be run slower by reducing its duty cycle. However, these are difficult to estimate.

7 Related Work

Processor thermal characteristics at the architectural level have been studied extensively in recent years [15].

Several techniques have been proposed to reduce chip temperature. Many of these techniques are reactive in nature in response to a thermal emergency detected by temperature sensors. These techniques either migrate the processor activity [16] or adapt processor resources to reduce temperature [15]. Brooks et al. [3] introduced dynamic thermal management (DTM) in reaction to thermal measurements. They applied techniques such as stalling execution or migrating activity to reduce temperature. Among DTM techniques, clock gating was shown to be effective in

reducing temperature across the chip in response to a thermal emergency. This technique has been used in many processors including Intel's Pentium M [25].

Leveraging the redundancy in a processor pipeline, several techniques have been proposed for temperature reduction. In [15] the power density is controlled by balancing the utilization of register file, issue queues, and functional units. Fetch throttling was also shown to be effective in reducing the temperature [15].

Dynamic voltage and frequency scaling in response to thermal emergency has been studied in [15, 24]. Temperature-aware task scheduling has been investigated at both architectural level and operating system levels for multiprocessors [11]. Ku et al. [23] proposed techniques for reducing cache temperature through power density minimization. They introduced a cache block permutation to maximize the distance between blocks with consecutive addresses. Several thermal management techniques for multi-core architectures are explored in [27]. Various core throttling policies were applied at core and processor level for chip thermal management. Heo et al. [16] have introduced a power density minimization through computational activity migration. They applied this technique to many processor blocks including the register file. This technique is effective, but incurs into a large area overhead since it requires replicating processor blocks.

Many recent works have focused mainly on reducing the power density and peak temperature of a processor. They specifically target the register file as it has been shown to be one of the hottest units in a processor [1, 15, 17, 19]. Previous work on the register file's power has mainly attempted to reduce the number of access to the register file, reduce the number of ports [10], or reduce the number of entries [8,9]. The algorithm we proposed in this work can be combined with these algorithms for further power and a potentially larger temperature reduction. Replication or banking register file has been studied in [2, 4, 12, 16,17, 26]. This work does not rely on either register file replication or banking, and as a result no significant area overhead is incurred except for region power down. However the benefit of our proposed approach can be improved in presence of replicated register file. Register assignment algorithm for low-power, low temperature VLIW register files were also introduced in [7]. These algorithms are applied at compiler level to an architecture where no renaming exists.

8 Conclusion

The register file is the most active and the hottest unit in an embedded processor. In this paper we proposed RELOCATE, an architectural solution to reduce the peak temperature of the register file. We analyzed the register file accesses and observed that while only a small number of physical registers are occupied at any given time, the total accesses are uniformly distributed over the entire physical register file during the course of execution. Our solution redistributes the access pattern to physical registers through a novel register allocation mechanism. We regionalize the register file such that even though accesses within a region are uniformly distributed, the activity levels are spread over the entire register file in a deterministic pattern. This allows us to power gate the unused regions of the register file. This resulted in a reduction of an average of 8.3°C in register file's peak temperature for standard benchmarks.

References

1. Mesa-Martinez, F.J., Nayfach-Battilana, J., Renau, J.: Power model validation through thermal measurements. In: International Symposium on Computer Architecture (2007)
2. Homayoun, H., Pasricha, S., Makhzan, M.A., Veidenbaum, A.: Dynamic register file resizing and frequency scaling to improve embedded processor performance and energy-delay efficiency. In: Design Automation Conference (2008)
3. Brooks, D., Martonosi, M.: Dynamic thermal management for high-performance microprocessors. In: High-Performance Computer Architecture (2001)
4. Tseng, J.H., Asanović, K.: Banked Multiported Register Files for High-Frequency Superscalar Microprocessors. In: International Symposium on Computer Architecture (2003)
5. Kessler, R.: The Alpha 21264 Microprocessor. IEEE Micro (March/April 1999)
6. Yeager, K.: The MIPS R10000 Superscalar Microprocessor. IEEE Micro (April 1996)
7. Zhou, X., Yu, C., Petrov, P.: Compiler-driven register re-assignment for register file power-density and temperature reduction. In: Design Automation Conference (2008)
8. Balasubramonian, R., Dwarkadas, S., Albonesi, D.H.: Reducing the complexity of the register file in dynamic superscalar processors. Micro (2001)
9. Borch, E., Tune, E., Manne, S., Emer, J.: Loose loops sink chips. In: HPCA (2002)
10. Park, I., Powell, M.D., Vijaykumar, T.N.: Reducing register ports for higher speed and lower energy. In: International Symposium on Microarchitecture (2002)
11. Murali, S., Mutapcic, A., Atienza, D., Gupta, R., Boyd, S., Benini, L., Micheli, G.D.: Temperature control of high-performance multicore platforms using convex optimization. In: Design, Automation and Test in Europe (2008)
12. Chaparro, P., Magklis, G., Gonzalez, J., Gonzalez, A.: Distributing the Frontend for Temperature Reduction. In: High-Performance Computer Architecture (2005)
13. SimpleScalar4 tutorial, SimpleScalar LLC, `http://www.simplescalar.com/tutorial.html`
14. MIPS Technologies MIPS32® 74K™ Licensable Processor Core, `http://www.mips.com/media/files/74k/FINAL_BDTI_MIPS_74k.pdf`
15. Skadron, K., Stan, M.R., Huang, W., Velusamy, S., Sankaranarayanan, K., Tarjan, D.: Temperature-aware microarchitecture. In: ISCA 2003 (2003)
16. Heo, S., Barr, K., Asanović, K.: Reducing Power Density through Activity Migration. In: International Symposium on Low Power Electronics and Design (2003)
17. Patel, K., Lee, W., Pedram, M.: Active Bank Switching for Temperature Control of the Register File. In: GLSVLSI 2007 (2007)
18. Powell, M., Yang, S., Falsafi, B., Roy, K., Vijaykumar, T.N.: Gated Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In: International Symposium on Low Power Electronics and Design (2000)
19. Han, Y., Koren, I., Moritz, C.A.: Temperature Aware Floorplanning. In: Workshop on Temperature Aware Computer Systems (June 2005)
20. Kumar, R., Kursun, V.: Impact of temperature fluctuations on circuit characteristics in 180 nm and 65nm CMOS technologies. In: International Symposium on Circuits and Systems 2006 (2006)
21. Dynamically adjustable load-sharing circular queues, US patent 6782461
22. Buti, T.N., McDonald, R.G., Khwaja, Z., Amdedkar, A., Le, H.Q., Burky, W.E., Williams, B.: Organization and implementation of the register-renaming mapper for out-of-order IBM POWER4 processors. IBM Journal of Research and Development (2005)

23. Ku, J.C., Ozdemir, S., Memik, G., Ismail, Y.: Thermal Management of On-Chip Caches Through Power Density Minimization. In: International Symposium on Microarchitecture 2005 (2005)
24. Donald, J., Martonosi, M.: Techniques for multicore thermal management: Classification and new exploration. In: International Symposium on Computer Architecture (2006)
25. Rotem, E., Naveh, A., Moffie, M., Mendelson, A.: Analysis of Thermal Monitor Features of the Intel Pentium M Processor. In: TACS Workshop at ISCA-31 (June 2004)
26. Homayoun, H., Pasricha, S., Makhzan, M.A., Veidenbaum, A.V.: Improving performance and reducing energy-delay with adaptive resource resizing for out-of-order embedded processors. In: Conference on Languages, Compilers and Tools for Embedded Systems (2008)
27. Donald, J., Martonosi, M.: Leveraging simultaneous multithreading for adaptive thermal control. In: Second Workshop on Temperature-Aware Computer Systems (2005)