A Power Model for Register-Sharing Structures

Balaji V. Iyer and Thomas M. Conte

Abstract Register files (RF) are known to consume about 20% of the power inside a processor. Embedded systems, due to area and timing constraints, generally have small register files, which can cause significant register pressure. This work explores how having a map-table or a map-vector can decrease the power dissipation in the processor. The distribution of register writes and sharing of commonly occurring values such as '0' is investigated. It is shown that systems with small register files obtain a greater power reduction than larger register files when these sharing structures are used. Finally, the proposed power model comes within 95% accuracy when compared using benchmarks on a synthesized Verilog softcore processor.

1 Introduction

Registers play a significant role to improve the instruction-level-parallelism (ILP), in modern systems [2, 5, and 11]. Large register files (RF), with the help of an optimal register allocation scheme, can greatly reduce the number of spill-code inserted in the program [1]. This can reduce the memory traffic, thus reducing the number of execution cycles necessary.

To remove false dependences in dynamically-scheduled processors, designers implement rename-map tables that map the architectural registers to physical registers [2, 3]. In statically scheduled systems, these false dependencies are resolved by using tighter register allocation schemes and/or large RF. In either case, there can be a huge amount of pressure exerted on RF [1].

Even though the idea of implementing a large RF is attractive for performance (figure-of-merit for performance is IPC), there can be setbacks in terms of energy

Balaji V. Iyer · Thomas M. Conte
Center of Efficient, Scalable and Reliable Computing, North Carolina State University, Raleigh, NC 27695
e-mail: bviyer, conte@ncsu.edu

or power dissipation, access time and chip area [12]. It is known that RF energy consumption accounts for about 10-20% of the overall energy consumption [3, 5]. For example, in the Motorola M.CORE architecture, the RF energy consumption accounts for 16% of the total processor power and 42% of the dual-path power.

Current embedded systems are required to achieve high performance, but many still must run on batteries [4, 13]. Battery technology significantly lags behind the processor's power consumption [13]. New technology processes currently allow higher integration density and larger chips, which leads to higher power consumption and heat radiation. High heat in chips can cause glitches, races and frequent errors [1].

To combat the performance degradation, researchers are exploring several hardware and software optimization techniques. Some of the software techniques include reducing value lifetimes [6, 10], content-based value storage [5], packing instructions into pairs [9], and value based register sharing [2, 11]. There are also several hardware based solutions to reduce the access time and power dissipation such as distributing the registers among clusters [10, 14] and gating certain unused registers in the RF [1].

The majority of the techniques mentioned above take advantage of value locality. The granularity of a value can be the whole word or even certain patterns in a word. These techniques exploit the fact that a large number of values written to registers are already present inside the RF. To do such value sharing between registers, some hardware addition is necessary.

The main aspect of this paper is to understand the power overhead added by these structures that aid register/value sharing. In addition, we point out when such structures are useful and help reduce the RF power dissipation. Moreover, we try to show how the patterns of register-writes can affect the power consumption of the RF. Finally, we validate our power-model using standard embedded benchmarks.

2 Related Work

Optimizing register-usage for performance improvement has been studied for the past two decades. The problems concerning power and heat dissipation in processors became a problem only in the nineties. Zyuban and Kogge in [16] study the power dissipation of an integer RF. Their models express the power consumption of a register in terms of the number of read-write ports and issue width. Similarly, Xao and Ye in [15] also provide models for finding power dissipation in RF.

Hu and Martonosi in [6] find that most read and write operations occur within a few cycles. They introduce a value aging buffer that saves recently-produced values so that the instructions requiring these values need not access them from RF. They received a power reduction of 30% with a less than 5% performance loss.

Kim and Mudge in [8] observe that only 0.1% of the cycles fully utilize a 16-bit read port of the RF. The main aim of their work was to reduce the number of read-ports, not the number of registers. They use a delay-writeback queue, an operation

pre-fetch buffer and request queues. They show 22% reduction in energy per register access.

Gonzalez et al. in [5] explain ways to share partial values between registers inside a RF. They find a 50% reduction in power consumption with 1.7% IPC loss. Ayala, Veidenbaum and Lopez-Vallejo in [1] propose ways to statically find registers that are not used during certain times and turn-off these registers to reduce power. They show 46% energy reduction in the entire MiBench benchmark suite.

Seznec, Toullec and Rouchecouste in [10] propose that restricting certain function units to write and read only a subset of registers (clustering the processor) can reduce the access time by 33% and power by 50%. Jain et al. in [7] evaluate the RF for an ASIP using ARM7TDMI as a test processor. It is shown that there is a high correlation between performance improvement and energy reduction. They further prove that slight increase in number of registers will give a large amount of power reduction in ASIP (˜50%).

Balakrishnan and Sohi in [2] discussed using a map-table for relieving register pressure by sharing values such as '0'. Tran et al. in [11] proposed a way to mark Reorder-buffers with one bit to indicate if the instruction's result from the ALU is a zero. [11] also discusses using a map-table as a possibility. These two papers are quoted extensively for value sharing inside the RF. In this work, we find the power contributions of these two types of structures for different configurations and percentage of zeros-writes in RF.

3 Experimental Frameworks

In order to view the register-value patterns, we picked four machines with different register-file sizes: ARM (thumb), OpenRISC, Simplescalar (PISA), and IA64. Table 1 below shows the register configuration of these four machines. The benchmark-set consists of 10 benchmarks from the EEMBC workload [20]. Table 2 explains these benchmarks. EEMBC is considered one of the most representative benchmarks in the industry today. Secondly, we modeled different RF configurations along with the appropriate sharing structures using Verilog. The original RF was extracted from the Verilog model of the OpenRISC 1000 processor [17]. The RF contains 2 read ports and 1 write port. These models were synthesized (0.18 μm IIT/OSU-standard-cell library) using Synopsys Design Analyzer and simulated using the Cadence NC-Verilog simulator to generate the VCD waveform files.

Table 1 Register file sizes of different architectures

Processor/Architecture	Number of Registers
ARM (thumb mode)	16
OpenRISC 1000 Processor	32
Simplescalar 2.0 Simulator (PISA) (using hard float)	64
IA-64 (using software floating point)	128
IA-64 (using hardware floating point)	256

Table 2 EEMBC Benchmarks Description

Benchmarks	Description
aifir01	FIR Filter
conven00	Convolutional encoder
Dither	Floyd-Steinberg error diffusion Dithering Algorithm
Ospf	Open-shortest path first/Dijkistra's Algorithm
Puwmod	Pulse Width Modulation Algorithm
rotate01	Image Rotation Algorithm
Routelookup	IP Datagram forwarding Algorithm
rspeed01	Road Speed Calculation
ttsprk01	Tooth-to-Spark tests in automobiles
viterb00	Viterbi Decoder

The synthesized register files along with the sharing structures are placed-and-routed using Cadence Design Encounter. The parasitic information is extracted during this process. Power analysis was done by Synopsys Primepower software using the VCD files, parasitic information and the synthesized gate-level verilog model. RF Inputs are discussed in section 5. Primepower is considered one of the most accurate power measurement tools, second only to SPICE [19].

4 Preliminary Analysis

To benefit from register sharing it is necessary to see if there is a great deal of duplicate values and constant values written into the registers. It was found by experimentation that '0' is the most frequent value written in the register-file. Table 3 shows the percentage of zero-writes and duplicate-writes (dupl. writes) in the ten benchmarks used in this work. Zero-writes are not necessarily a subset of duplicate writes, since in the life-time of a program, certain values can be re-written several times but not duplicated in the RF.

Table 3 Percentage of Zero-Writes and Duplicate-Writes for Different Architectures.

	ARM (Thumb Mode)		**OpenRISC 1000**		**Simplescalar 2.0**		**IA-64 (Soft Float)**		**IA-64 (Hard Float)**	
Benchmark	Zero-Write	Dupl. Write	Zero-Write	Dupl. Write	Zero-Write	Dupl. Write	Zero-Write	Dupl. Write	Zero-Write	Dupl. Write
aifirf01	24%	43%	13%	46%	20%	67%	1%	5%	1%	5%
conven00	15%	48%	30%	49%	25%	48%	1%	7%	1%	7%
dither	8%	14%	8%	21%	10%	15%	2%	10%	2%	10%
puwmod	3%	25%	6%	39%	3%	30%	1%	9%	1%	10%
rotate	3%	17%	3%	23%	3%	17%	1%	10%	1%	11%
routelookup	5%	40%	7%	53%	5%	44%	1%	5%	1%	5%
rspeed01	4%	20%	21%	45%	3%	23%	1%	7%	1%	7%
ttsprk01	5%	28%	25%	53%	3%	39%	1%	8%	1%	8%
viterbi	11%	31%	12%	40%	7%	42%	1%	6%	1%	6%
ospf	6%	35%	19%	41%	3%	32%	1%	5%	1%	5%

There are a large number of duplicate and zero-writes occurring inside a RF. The distribution of these writes also varies across benchmarks. In some benchmarks such

as dither, the zero-writes occur in a pattern, but in benchmarks such as routelookup, they are very bursty. In the remaining benchmarks they are more or less random.

5 Register Sharing Techniques

Power dissipation of the RF is a popular research area. Many register optimization techniques can greatly help in improving the performance of the program. Most register-sharing techniques typically encompass using either a map-table [2] or a map-vector [11]. Figure 1 explains the top-level block diagram of these two structures.

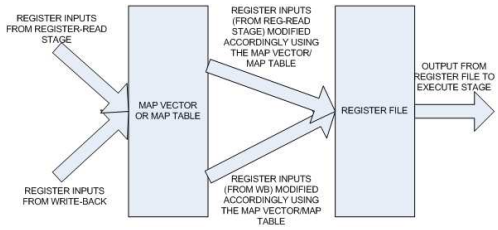

Fig. 1 Top-level block diagram of the map-table or the map-vector.

A register map-table is used to map certain architectural registers to physical registers that hold the certain values. Since there is a significant amount of '0' written into the RF, we choose one architectural register (r0) that is permanently grounded to zero. Any register whose value is zero is mapped to r0. The primary advantage of this scheme is that we do not access the RF for zero-writes, thus saving power.

A Second approach is to use a map-vector to indicate which registers hold the zero value. Each register is assigned a bit in the vector to indicate if its result is zero. If the corresponding bit is set, then the register-file is not accessed. In our experiments, map-vectors generally consume about 30-40% less power than a map-table. As soon as we reach the write-back stage, we know the register value along with the result to be written. If the value written is zero then a certain bit is set in a map-table and the RF is not accessed. Otherwise the value is forwarded to the RF and written to the appropriate register. Figure 2 present flow-charts for the steps in these stages. These structures were designed such that the processor's clock-cycle remains unaffected. The base processor's clock-period remains unaffected by using these structures.

To accurately portray register writes, 1-million register writes and 2-million register reads were generated. In each run we increased the number of zeros by a set percentage. Throughout the paper, the number of zeros in the stream is given in terms of percentage. It is worth mentioning that we only read registers that have already been written (with the exception of the stack pointer and the return value

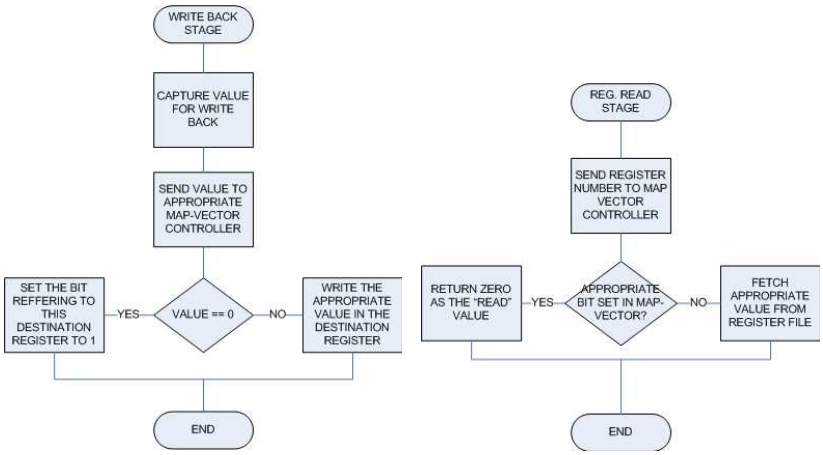

Fig. 2 Flow-diagram for the write-back stage (a) and reg-read stage (b) of the processor that uses a map-vector.

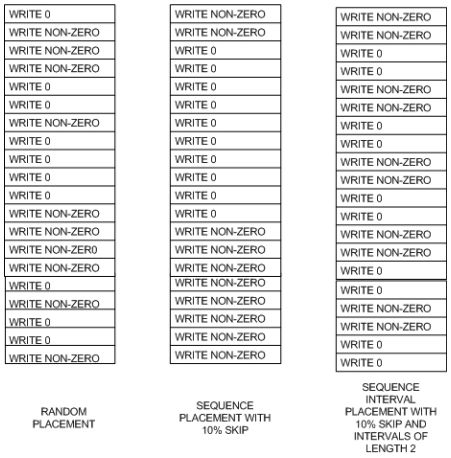

Fig. 3 Example of Different placement of zero-writes in our experiments (50% zero-writes) for 20 register-writes

register[1]). Figure 3 explains different test-input schemes. In Figure 3, the number of writes was reduced to 20 for the ease of explanation.

In addition, we also created sequences of zeros writes into the RF. These sequences of writes are placed in different regions. For example the sequence 40-10 implies that the first 10% of the register writes are non-zero values. Next 40% of the writes are zeros. Then, the remaining 50% of the values are non-zero writes.

[1] Registers r1 and r9 are designated as stack pointer and return register as described in [17].

We take this model further and break this zero sequence into intervals to see their effects. For example, 40-40-10 implies that the first 40% of the writes are non-zeros, and then in the next 60%, the 40% zeros (bold) are divided into intervals of 10%. In the next section we explain the results of these distributions.

6 Results

To see the impact of the RF size on power dissipation a 32-bit RF of size 16, 32, 64, 128 and 256 registers is modeled. The percentage of zero-writes (distributed randomly) is varied from 0-100% in 5% intervals. Figure 4-8 show our findings.

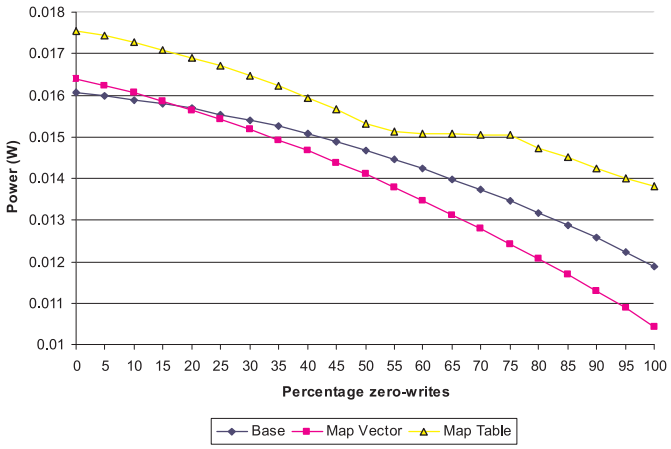

Fig. 4 Power dissipation (PD) for random reg-write for (RF size: 16)

In all cases, using a RF with a map-table consumed more power than using just the RF without any value sharing (the "base" case). The map-vector gives a power advantage when we have 20% and 45% of zeros for the RF size of 16 and 32, respectively. The map vector fails to provide a power-reduction for the 128 and 256-size RF. This is because the internal power of the cell dominates significantly for larger RF. For 64, the break-even point is after 95%.

In this work, leakage power is not a major factor. Static power, however, is a problem in the memory hierarchy [18]. Inside the processor, the dynamic power is a major contributor (˜90-95%). In addition, the static power is not activity-based. The only way to reduce static power is through turning-off certain units, which is beyond the scope of this work [18].

Next the impact of writing zeros into RF in burst sequences placed at different parts of the trace is examined. Specifically, we wanted to see if scheduling a chunk of zero-writes in the beginning, middle or end would be most beneficial. Zero-writes were placed at 10%, 40% and 80% of the trace to see their impact. The chunk-size was modeled from 10-80% (whenever applicable). Table 4 shows our findings.

Positive values in tables 4 and 5 indicate a power reduction, while negative values indicate a power increase. Since the map-table failed to provide any power reduction for the overall system, we do not show its results for the rest of the paper.

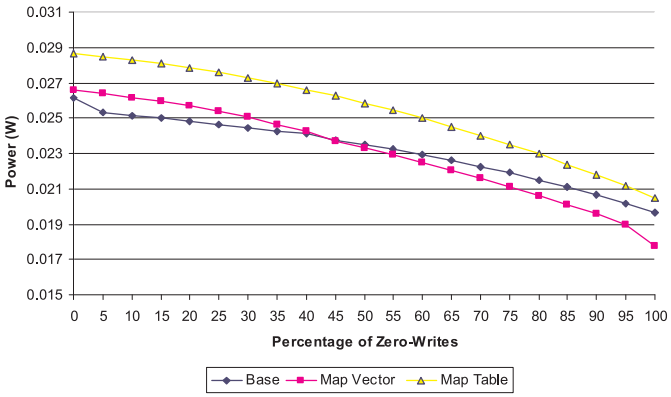

Fig. 5 Power dissipation for random reg-write (RF Size: 32)

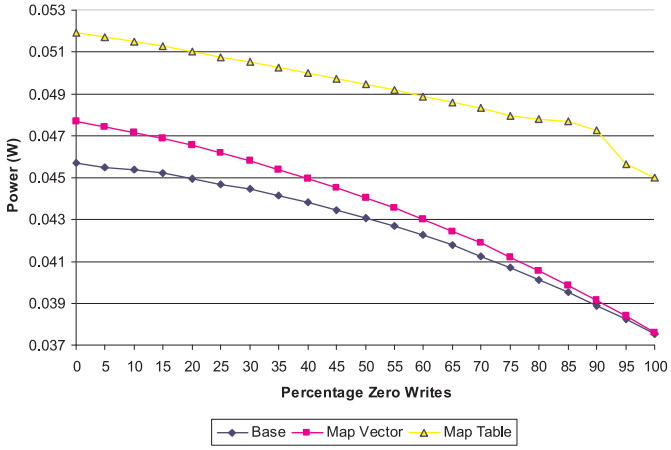

Fig. 6 Power-dissipation for random reg-write (RF size: 64)

It can be seen that for smaller register files, there is a power reduction even when zero-writes are not significant. As the register-file size increases, there must be a significant burst of zeros to get a power reduction. To understand why, we converted the register file into an appropriate SPICE model using the Synopsis Virtuoso toolset ("icfb") to see the transistor layout. We noticed that on large register files, the map-vectors created a significant amount of latches, which consumed a non-negligible amount of power. In addition, the wire-lengths between these map-vectors and the

register-file interface were also huge. This increase in length caused an increase in wire-capacitance (verified using design encounter's parasitic values, and the capacitance using SPICE), which increased the dynamic power.

Now, we extend our previous results further and divide these sequences into burst interval chains. Typically in a program, the compiler will have an easier time to distribute 5-2% zero-write chains as supposed to a one single 10% chain. The values of intervals where chosen as 2%, 5%, and 10% respectively. These values are chosen because they are common divisors of 10, 40 and 80, thus making a fair comparison. Table 5 displays the results of this experiment. The trends noticed in this experiment are similar to the ones given in Table 4.

According to Table 4, small register-files greatly benefited with such structures when there were significant number of bursts. For small bursts, the map-vector contributed negatively to the power dissipation. One odd trend in Table 4 and 5 is that for the same set of sequences, a RF of size 64 did slightly worse than 128. Registers were chosen to write and read based on a random number generator. For a 128 RF, the probability of picking the same register to be written twice is significantly less than that of a 64 RF. Thus, there was more switching inside the RF of 64 than that of 128, thus we find a 0.3-0.5% difference. This phenomenon did not affect the 256 RF.

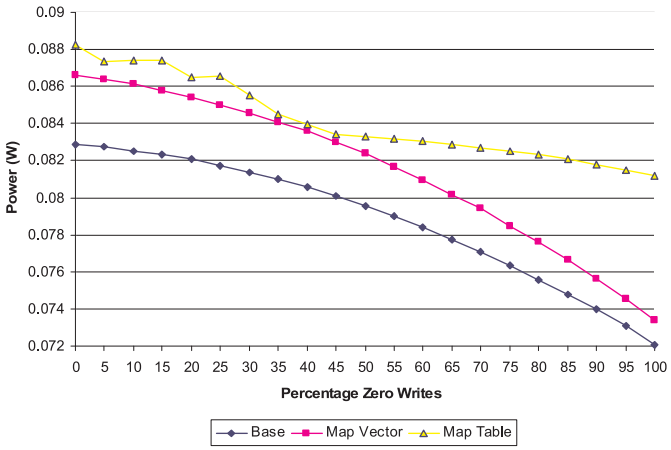

Fig. 7 Power Dissipation for random reg-write (RF size: 128)

The next step is to validate our power-model. For this work, we use the OpenRISC 1000 (OR32) processor. This processor is considered a valid representation of modern embedded systems [17]. The ten benchmarks mentioned in section 3 were executed on a Verilog-core of OR32 and the power values of each processor unit are captured separately. For this work, we only present the power-savings of the register-file. Table 6 displays our results. It can be seen that the power savings depicted using our synthetic benchmarks matched very closely to the values obtained using representative benchmarks. For example, the benchmark *conven00* had 30% zero-writes, and exhibited a 1.44% power reduction. Value obtained from the syn-

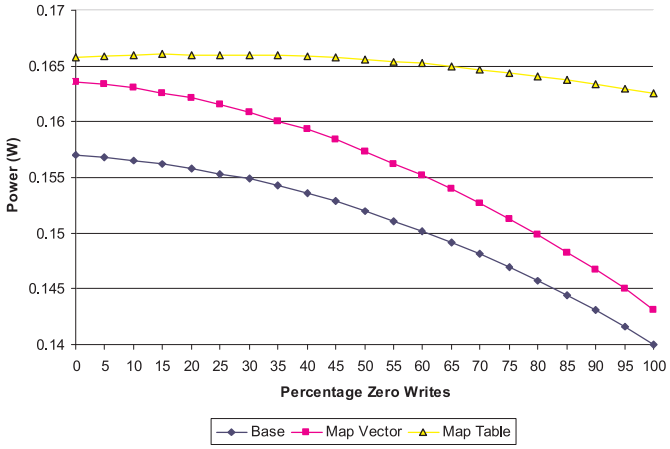

Fig. 8 Power Dissipation for random reg-write (RF size: 256)

thetic benchmarks in Figure 5 show a 1.47% reduction in power. The difference is mainly due to pipeline stalls and the differences in values that are written into the register. Similarly, *dither*, even though had approximately the same number of zero-writes as *routelookup*, exhibited lower power dissipation due to the bursty nature of the zero-writes. The rest of the benchmarks, even though they did not fall into the granularity that was studied in this paper, had power dissipations that fell within the correct range.

Table 4 Percentage of Zero-Writes in Burst Sequences for Different Register-file sizes

	16 Regs.	**32 Regs.**	**64 Regs**	**128 Regs**	**256 Regs.**
10--10	-0,30%	-4,10%	-4,10%	-4,30%	-4,10%
10--20	-0,30%	-4,10%	-4,10%	-4,30%	-4,00%
10--30	-0,40%	-4,20%	-4,10%	-4,30%	-4,00%
10--40	-0,40%	-4,20%	-4,10%	-4,30%	-4,00%
10--60	-0,50%	-4,30%	-4,10%	-4,30%	-4,00%
10--80	-0,60%	-4,40%	-4,20%	-4,30%	-4,00%
40--10	2,60%	0,40%	-3,00%	-3,60%	-3,60%
40--20	2,40%	0,30%	-3,10%	-3,70%	-3,60%
40--30	2,00%	0,30%	-3,20%	-3,80%	-3,60%
40--40	1,80%	0,40%	-3,40%	-3,80%	-3,70%
40--50	1,60%	0,50%	-3,40%	-3,90%	-3,70%
80--10	7,40%	6,90%	-1,60%	-2,70%	-3,00%
80--20	6,50%	7,40%	-1,80%	-2,90%	-3,10%

Table 5 Percentage of Zero-Writes in Sequence-intervals for Different Register-file sizes

	16 Regs.	32 Regs.	64 Regs	128 Regs	256 Regs.
10--10--2	-25,30%	0,40%	-2,70%	-1,70%	-5%
10--10--5	-25,40%	0,40%	-2,80%	-1,70%	-5%
10--10--10	-25,40%	0,40%	-2,80%	-1,70%	-5,00%
10--40--2	-25,30%	0,40%	-2,70%	-1,70%	-5,10%
10--40--5	-25,30%	0,40%	-2,80%	-1,70%	-5,00%
10--40--10	-25,30%	0,40%	-2,70%	-1,70%	-5%
10--80--2	-25,40%	0,40%	-2,70%	-1,70%	-5,00%
10--80--5	-25,30%	0,40%	-2,70%	-1,70%	-5,00%
10--80--10	-25,40%	0,40%	-2,80%	-1,70%	-5,00%
40--10--2	4,00%	2,30%	-1,90%	-1,10%	-3,60%
40--10--5	3,80%	2,10%	-2,10%	-1,30%	-3,50%
40--10--10	3,80%	2,10%	-2,20%	-1,40%	-3,60%
40--40--2	3,10%	1,70%	-2,20%	-1,30%	-4,90%
40--40--5	3,00%	1,50%	-2,40%	-1,50%	-4,80%
40--40--10	3,00%	1,50%	-2,40%	-1,50%	-4,60%
80--10--2	10,10%	2,70%	-2,00%	-1,00%	-1,70%
80--10--5	9,90%	2,50%	-2,10%	-1,30%	-1,70%
80--10--10	9,00%	2,70%	-2,10%	-1,30%	-2,10%

Table 6 Power Reduction using Map-vector on EEMBC Benchmarks

	Percent Zero Writes	Power Savings
aifirf01	13%	-1,06%
conven00	30%	1,44%
dither	8%	-0,78%
puwmod	6%	-1,33%
rotate	3%	-1,33%
routelookup	7%	-1,33%
rspeed01	21%	0,28%
ttsprk01	25%	0,74%
viterbi	12%	-0,95%
ospf	19%	0,23%

7 Conclusion

This study reveals several power dissipation patterns of the RF. First, adding a map-vector can cause a power reduction only when there is a significant amount of zero-writes present in the workload. Similarly, scheduling multiple zero-writes together, regardless of the destination registers, can give some power reduction for a small RF. Some power-reduction can also be achieved if it is able to divide the register write into intervals rather than just placing them at random. Finally, the power difference obtained when using such structures is at least 95% accurate when verified using real benchmarks.

These techniques can be extended to a physical or an architectural register file. The impact of zero-writes on power dissipation can be useful in several ways. For example, a compiler can use this information and schedule instructions that potentially have a zero-write together and form chunks. In addition, the processor can

gate a map-vector when the compiler or a profiler can predict and communicate that the number of zero-writes in the system is low. Another option is to run a similar workload in a simulator to predict the amount of zero-writes and have the compiler schedule specialized instructions that enables or disables the register sharing structure based on the workload.

References

1. J. L. Ayala, A. Veidenbaum, M. Lopez-Vallejo, "Power-Aware Compilation for Register file energy reduction," *International Journal of Parallel Programming,* Vol. 31, No. 6, 2003
2. S. Balakrishnan, G. S. Sohi, "Exploiting Value Locality in Physical Register Files," *Intl. Symposium on Microarchitecture,* 2003
3. R. Balasubramonian, S. Dwarkadas, D. H. Albonesi, "Reducing the Complexity of the Register File in Dynamic Superscalar Processors," *Intl. Symposium on Microarchitecture, 2001*
4. A. Bechini, T. M. Conte, C. A. Prete, "Opportunities and Challenges in Embedded Systems," *Proc. of the Intl. Symposium on Microarchitecture*, August 2004.
5. R. Gonzalez, et al., "A Content Aware Integer Register File Organization," *ISCA*, 2004
6. Z. Hu, M. Martonosi, "Reducing Register File Power Consumption by Exploiting Value Lifetime Characteristics," *Workshop on Complexity Effective Design, 2000*
7. M. K. Jain, et al., "Evaluating Register File Size in ASIP Design," *Proc. of 9th Intl. Symposium on Hardware-Software Codesign,* 2001
8. N. S. Kim, T. Mudge, "The Microarchitecture of a Low Power Register File," *ISLPED,* 2003
9. M. T, Lee, et al., "Power Analysis and Minimization Techniques for Embedded DSP Software," *IEEE Trans. on VLSI Systems, Vol.* 5, No. 1, 1997
10. A. Seznec, E. Toullec, O. Rochecouste, "Register Write Specialization Register Read Specialization: A Path to Complexity-Effective Wide-Issue Superscalar Processors," *International Symposium on Microarchitecture,* 2002
11. L. Tran, et al., "Dynamically Reducing Pressure on the Physical Register File through Simple Register Sharing," *Intl. Symposium on Performance Analysis of Systems and Software,* 2004
12. M. Pericas, et al., "An Optimized Front-end Physical Register File with Banking and Writeback Filtering," *Workshop on Power-Aware Computer Systems,* 2004
13. L. Wehmeyer, et al., "Analysis of the Influence of Register File size on energy consumption, code-size and execution time," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 20, No. 11, November 2001
14. J. Zalamea, et al., "Hierarchical Clustered Register File Organization for VLIW Processors," *Proc. of the Intl. Parallel and Distributed Processing Symposium, 2003*
15. X. Zhao, Y. Ye, "Structure Configuration of Low-power register file using energy model," *Proc. of the IEEE Asia-Pacific Conference on Application-Specific Integrated Circuits,* 2002
16. V. Zyuban, P. Kogge, "The Energy Complexity of Register Files," *Proc. of ISLPED,* 1998.
17. "OpenRISC Architecture Manual," http://www.opencores.org, 2003
18. N. S. Kim et al., "Leakage current: Moore's law meets static power," IEEE Computer, Vol. 26, Issue 12, 2003
19. R. Goering, "Synopsys launches more powerful power-analysis tool," EE-times, 2000
20. "Embedded Benchmark Consortium", http://www.eembc.org/