Appears in 13th Symposium on Integrated Circuits and System Design, Manaus, Brazil, September 2000

Energy-Efficient Register Access

Jessica H. Tseng and Krste Asanović
MIT Laboratory for Computer Science, Cambridge, MA 02139
{jhtseng|krste}@lcs.mit.edu

Abstract

We present and evaluate seven techniques to reduce energy dissipation for accesses to a processor register file: **modified storage cell** *avoids bitline discharge for zero bits,* **precise read control** *avoids fetching unused operands,* **latch clock gating** *disables latch clocks when operands are not needed,* **bypass skip** *turns off regfile reads when bypass circuitry will supply the value,* **bypass R0** *treats accesses to R0 separately,* **split bitline** *reduces access energy for frequently-used registers, and* **read caching** *avoids regfile reads when the same register is read twice in succession. For a 0.25 μm CMOS three-port regfile, we find individual energy savings of 27%, 21%, 8%, 16%, 14%, 12%, and 1% respectively, and a combined saving of 59% when all seven techniques are used in combination. The total area overhead is around 17% and the total delay overhead is around 3%.*

1. Introduction

Register files represent a substantial portion of the energy budget in modern microprocessors [2, 3, 9]. For example, in Motorola's M.CORE architecture, the register file consumes 16% of the total processor power and 42% of the data path power [2]. In this paper, we evaluate seven techniques to reduce register file access energy by either lowering the switching activity or the capacitance switched. Several of these techniques have been proposed earlier, but in this paper we present the first detailed evaluation of their energy dissipation and show how all techniques interact for a pipelined RISC processor running large benchmark programs.

The paper is structured as follows. Section 2 describes our experimental methodology. Sections 3–10 describes our base case register file design and the seven energy saving techniques in detail: *modified storage cell* [7] which avoids bitline discharge for zero bits, *precise read control* [1, 7] which avoids fetching unused operands, *latch clock gating* which disables latch clocks when operands are not needed, *bypass skip* [1, 7] which turns off regfile reads when regfile bypass circuitry will supply the value, *bypass R0* [7] which treats accesses to R0 separately, *split bitline* [7] which reduces access energy for frequently-used registers, and *read-caching* which avoids regfile reads when the same register is read twice in succession. In Section 11 we show how the seven techniques can be combined to yield a larger total saving, with a final reduction by a factor of 2.4 in total access energy at a cost of a 17% area increase and a 3% delay increase. We conclude in Section 12.

2. Evaluation Methodology

For this study we focus on the design of the integer register file for a single-issue pipelined MIPS-II compatible RISC microprocessor (similar to the MIPS R3000 [4]). This design point is representative of processors targeted at low-power embedded applications. The regfile contains 31$\times$32-bit writable registers plus a fixed 32-bit zero register (R0), and has two read ports and one write port. A bypass network is used to forward results to subsequent instructions to avoid extra latency from regfile accesses. We evaluate the energy dissipation of our various alternatives by combining the results of bit-accurate and cycle-accurate microarchitectural simulations with energy models extracted from custom layouts of the register file and bypass network.

Our simulator models a five-stage pipeline (Figure 1), which has a single interlocked load delay slot, 17 delay cycles between the issue of an integer multiply and read of result, and 32 delay cycles between the issue of an integer divide and the read of the result. We do not model cache misses as these do not affect regfile energy assuming the processor stalls for cache misses. The simulator traces user-level instructions and records register file access information, instruction operands' bypass frequency, and bit-level data switching activity.

Our benchmark workload is shown in Table 1. Each benchmark was compiled with `gcc` version 2.7.0 with -O3 optimization for the MIPS-II architecture and linked with the Cygnus `newlib` standard C library. Each benchmark was run to completion (a total of over 14 billion cycles of

Benchmark {Data Set}	Instruction Count (Millions)	Cycle Count (Millions)	Description
SPECint95:m88ksim {test}	519	567	Motorola 88100 microprocessor simulator
SPECint95:li {test}	997	1,129	xlisp interpreter
SPECint95:go {train}	579	631	An internationally ranked go-playing program
SPECint95:gcc {ref:2c-decl-s}	1,396	1,524	Based on the GNU C compiler version 2.5.3
SPECint95:vortex {test}	10,054	11,123	An object oriented database
SPECint95:jpeg {test:specmum.ppm}	567	710	JPEG 24-bit image compression standard
Sun:g721 {clinton.g721}	528	625	Adaptive differential PCM voice compression

Table 1: Benchmark and dataset descriptions, instruction counts, and cycle counts.

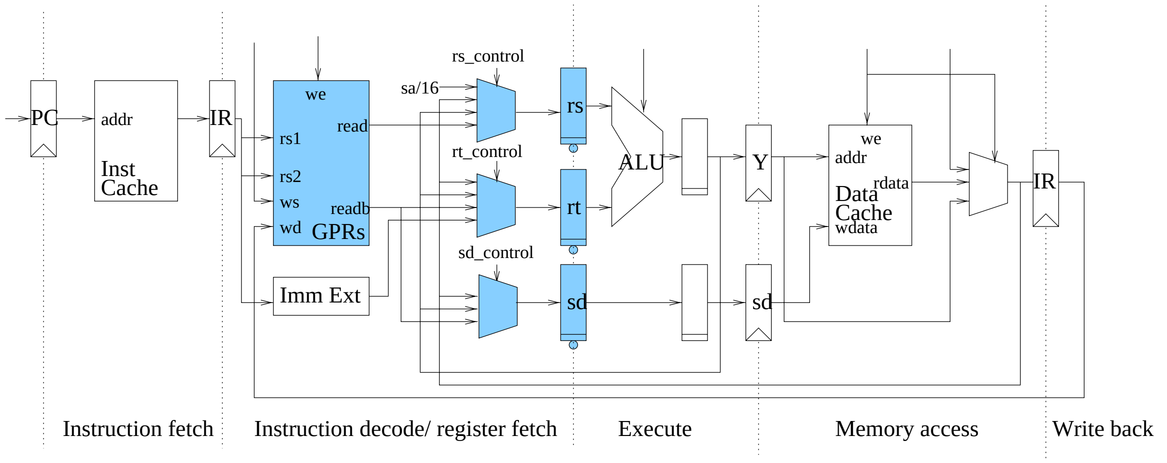

Figure 1: MIPS RISC core pipeline structure.

processor operation) with averages weighting each benchmark equally.

We developed an energy model for the register file and bypass network, shown as the shaded region in Figure 1. We model the average energy consumption as:

$$E = \frac{1}{2}\sum_{r} f_r \cdot C_r \cdot V_r \cdot V_{dd}$$

where f_r is the average transition frequency of node r as determined by the simulator, C_r is the switching capacitance related to node r as extracted from circuit layouts, V_r is the voltage swing on the node, and V_{dd} is the supply potential. We measure energy for the complete register access including bypass muxing and latching.

We designed circuits to run at 2.5 V in a 0.25 μm CMOS technology from TSMC. Magic [5] was used for layout, and the SPACE 2D extractor [8] was used to extract layout parasitics for circuit simulation, including capacitance to the substrate, fringe capacitance, crossover coupling capacitance, and capacitance between parallel wires. HSpice was used to simulate the extracted netlist and to determine the effective switching capacitance, C_r, for the energy estimation model. We measure regfile delay from the start of the second half of the cycle until read data is available at the output of the bypass transparent latch, which represents the critical path in the decode stage. The target read delay is under 1 ns to satisfy the ALU input setup time required to reach our nominal processor clock rate of 400 MHz.

3. Base Case Register File Design

The regfile used in this study is a high performance dynamic design with two single-ended read ports and one differential write port (Figure 2). Registers are written and read bitlines are precharged during the first half of the cycle, while read data is sensed during the second half of the cycle. Static address decoders evaluate a half-cycle ahead of bitline read or write. The base eight-transistor storage cell (Figure 3) occupies 30.5 μm^2, and all regfiles were de-

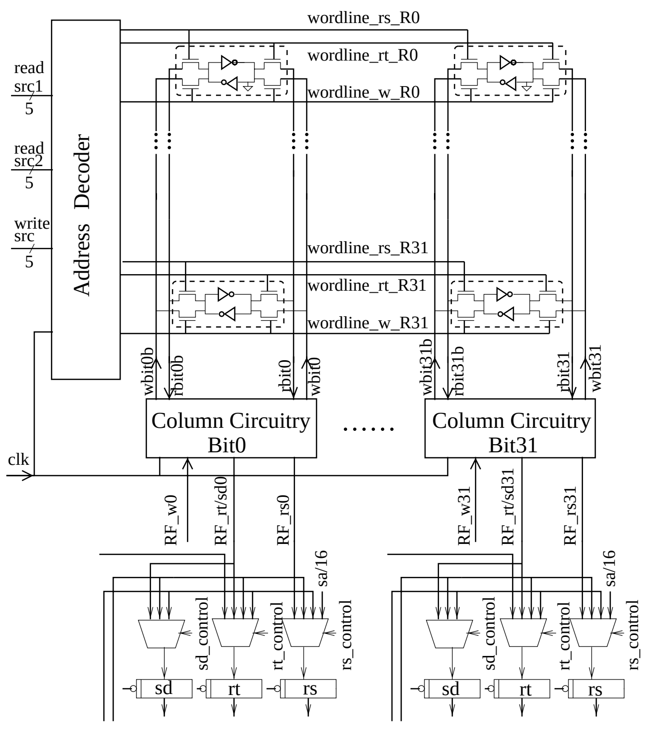

Figure 2: Base register file design.

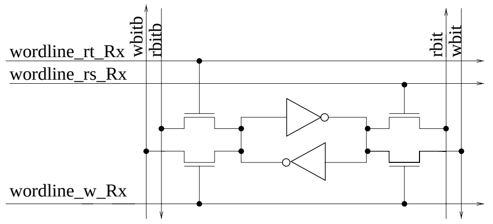

Figure 3: Base register file storage cell.

signed to use only the lowest 3 of the 5 available layers of metal. Figure 4 shows the column circuitry, which includes a clocked inverter sense amplifier to speed bitline sensing. All dynamic nodes have keeper transistors to support fully static operation. The bypass network uses transmission gate muxes and latches, with latches similar to those in the IBM PowerPC603 [6].

4. Modified Storage Cell

Our benchmark simulations show that 82% of the bits fetched from the regfile are zeros. We can reduce the regfile read bitline switching activity by modifying the bitline connections to the storage cells to minimize the number of high-to-low and low-to-high transitions. Since both sets of read bitlines are precharged high, they dissipate energy

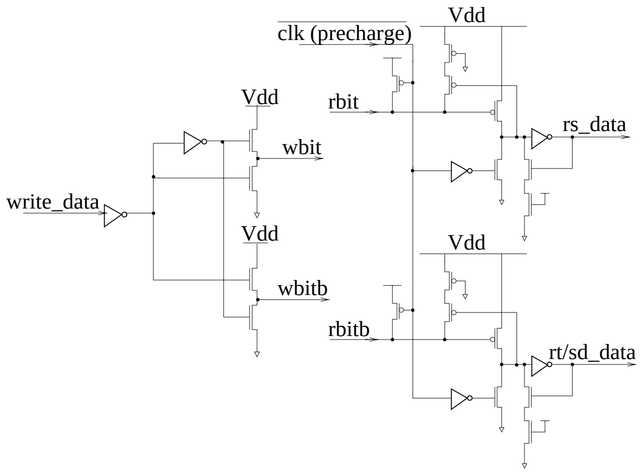

Figure 4: Base column circuitry for one bit slice.

only when the storage cells cause them to discharge their precharged value. We can also remove the R0 row because if no wordline is enabled, the regfile will return the required zero value. The asymmetry of the modified cell (Figure 5) increases cell area by 17% to add a connecting wire, but then also allows larger internal pulldowns which avoid any delay penalty when both read ports are active simultaneously (total regfile area increases by 9%). Energy saving ranges from 17%–36% across benchmarks with an average of 27%.

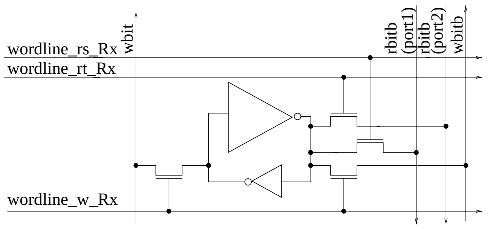

Figure 5: Modified storage cell implementation.

5. Precise Read Control

The base case register file always accesses both operands even if the machine instruction only requires zero or one. Our dynamic benchmark statistics show that on average each instruction only requires 1.3 operands. The decode stage control logic already has to calculate which operands are necessary for bypassing and interlocking. With minimal extra control logic we can also disable word lines, and hence bitline discharge, by gating the word line enable pulse in the second half of the cycle. Although the read address decoders are always active in the first half of the cycle, they represent only a small portion of the total access energy. Compared with the base case, precise read control leads to energy savings from 15%–27% across benchmarks with an average of 21%. We assume that the decoding of required operands completes in the first half of the cycle, and hence that there is no access time penalty to this scheme.

6. Latch Clock Gating

Not all instructions make use of all the values in the bypass latches. Our simulations show that only around 81% of instructions use values held in the *rs* or *rt* latches (which can be either register or immediate values), while the *sd* register is only used by store instructions (around 10% of all instructions). We can reduce energy by not clocking latches whose values are not needed. This results in a 8% energy savings over the base case that always clocks all latches.

7. Bypass Skip

Our simulations of the processor pipeline show that an average of 36% of all necessary operands are bypassed from other stages of the pipeline instead of being read from the regfile. Similar to the precise-read control method, if we can determine that the bypass network will supply the value in the first half of the cycle, we can gate the wordline enable and avoid discharging bitlines in the second half of the cycle. Control logic is already present to drive the bypass network. If determining the bypass control takes longer than the first half of the cycle, this scheme will increase latency otherwise there is no access time penalty. Bypass-skip leads to energy savings between 11%–23% across benchmarks with an average of 16%.

8. Bypass R0

If we provide a separate zero input to the bypass mux we can remove the zero cells from the regfile and avoid discharging bit lines on a read. We can also save energy by never driving write bitlines when writing R0. R0 is accessed frequently in the base case and energy saving ranges from 7% to 17% with an average of 14%.

9. Split Bitline

Our simulations reveal that a few registers account for most of the register file accesses. The 8 most popular registers account for 75%–92% (average 83%) of all regfile accesses. Moreover, the benchmark traces indicate that particular registers such as R0, R2, R3, R4, R5, R6, R16, and R29 are always accessed more frequently than others due to MIPS assembler conventions; R2 and R3 are used for expression evaluation and to hold integer function results; R4, R5, and R6 are used to pass the first three actual integer arguments; R16 is the first callee-saved register; and R29 contains the stack pointer.

We can decrease average bitline switching capacitance by splitting bitlines into two partitions, one with the most popular few registers and the other holding the remainder. This register file hierarchy reduces the energy cost of accessing the most-frequently-used registers, with only a small delay penalty to access the least-frequently-used registers. A tradeoff exists between including more registers in the popular partition and reducing the energy of each access to the popular partition. We determined that there is a broad optimum in the range of 5–9 popular registers and present results for a design with 8. We use a single n-type transistor to separate the two partitions (Figure 6), and this transistor is opened only when accessing the least-frequently-used registers. Also, we only precharge the larger partition to a threshold drop below V_{dd} through an n-type transistor and the address decoder wiring is changed to map the popular register numbers into the short bitline partition. The split-bitline energy saving ranges from 11% to 13% with an average of 12%. The constant energy consumption of the decoders, column circuitry, and bypass network limits the maximum possible energy saving to 22%.

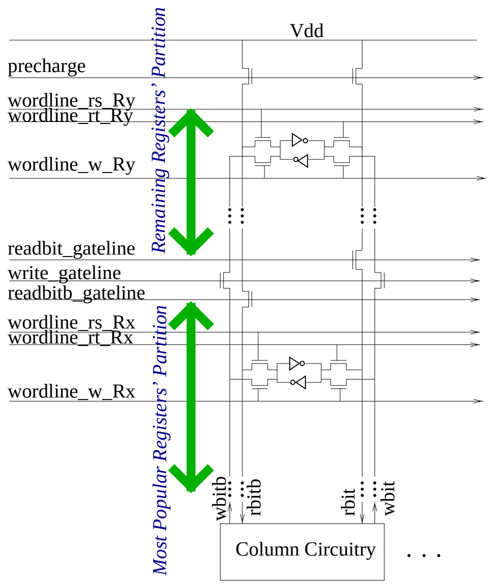

Figure 6: Split-bitline regfile implementation.

10. Read Caching

Our simulations show that in some cases, two successive instructions read the same register from the register file, e.g., in the following sequence,

```
add r4, r1, r6
xor r9, r1, r2
```

the register `r1` is read twice into the same latch by two successive instructions. We can reduce energy in this case by not clocking the *rs* latch and not reading the register file for the second instruction. Our simulations show that around 9% of accesses to the *rs* latch can be supplied via this simple read cache. Most of these cacheable reads are due to repeated use of the stack pointer register during register save/restore code. Because we did not observe much cacheability of the *rt* and *sd* latches, we do not attempt to use read caching for those latches.

There is control logic overhead to managing the register read cache. The previously read register address must be compared with the current register read address. This requires an extra 5 bits of latch to hold the old read register address, a single bit to indicate if the latch state is valid plus another single bit to hold the address comparison result, as well as a 5-bit compare circuit. We include the energy cost of the register address latch and comparison circuit in our numbers. Our simulations show a 1% energy savings from using the read cache over the base case. This low energy saving is due to the overhead of the extra control logic.

11. Combining Techniques

Table 2 summarizes our results showing area, delay, and energy for each of the seven techniques when applied individually to the base case. We can achieve greater savings by combining all seven techniques as shown by the last row in the table. We choose to apply the techniques in the order presented above. The earlier techniques are easiest to add and incur the largest savings. The later techniques have reduced incremental savings because they often have some overlap with earlier techniques in the way they achieve savings.

Figure 7 shows the progressive reduction in regfile energy as we add the techniques, and also illustrates where the energy savings occur. The modified storage cell (MSC) achieves most of its savings in the bitlines but there are also savings in the column circuitry and the muxes and latches due to the reduced number of transitions. MSC is very effective at reducing bitline energy, so when we add precise read control (PRC), we find the biggest saving is now in the column circuitry from reduced activity in the precharge and sense amp circuitry. As expected, latch clock gating (LCG) shows savings only in the latch energy. When bypass skip (BS) is added, again there is a small further reduction in bitline energy, but the largest reduction is in the column circuitry. BS complements PRC; PRC removes reads for operands that are never required while BS removes reads for operands that are required but whose current value is not in the register file. Bypass R0 (BR0) has little effect now (2%) after applying the other techniques given that we have already used MSC to avoid most energy associated with reading zeros. The savings that remain are from avoiding switching on the bitlines for writes to R0, and for not switching the column circuitry on a read of R0. BR0 adds some mux energy to all accesses because the bypass muxes are now larger to support the separate zero input.

Case	Area (ratio)	Read Latency (ns)	Energy/cycle (pJ)	
BASE	1.00	0.94	63.2	(100.0%)
MSC	1.09	0.94	45.9	(72.6%)
PRC	1.00	0.94	50.2	(79.5%)
LCG	1.00	0.94	58.4	(92.4%)
BS	1.00	0.94	53.0	(83.8%)
BR0	1.03	0.94	54.4	(86.1%)
SB	1.02	0.97	55.8	(88.3%)
RC	1.01	0.94	62.4	(98.7%)
COMB	1.17	0.97	26.2	(41.5%)

Table 2: Overall regfile area, performance, and energy evaluation for the base case regfile (BASE), the modified-storage-cell regfile (MSC), the precise-read-control regfile (PRC), the clock-gating regfile (LCG), the bypass-skip regfile (BS), the bypass-R0 regfile (BR0), the split-bitline regfile (SB), the read-cache regfile (RC), and the combination regfile (COMB).

Once we have applied the other techniques, the split bitline technique provides little incremental savings (1%). Many of the popular register accesses are satisfied from the bypass skip, and MSC reduces bitline energy for the remaining accesses. Although the read cache saves energy in a different way than the other techniques, it also has only a small incremental saving (1%) due to its high control overhead. If we only apply the first five techniques, there is no delay penalty and a 54% overall energy saving.

The final breakdown of register file energy shows that we have successfully removed most of the bitline energy. The major contributors to final energy are the column circuitry, bypass muxes, and latches. The breakdown also shows how small the decoder energy is (<2% of final energy), justifying our decision to always activate the decoders regardless of whether a register file port is needed or not.

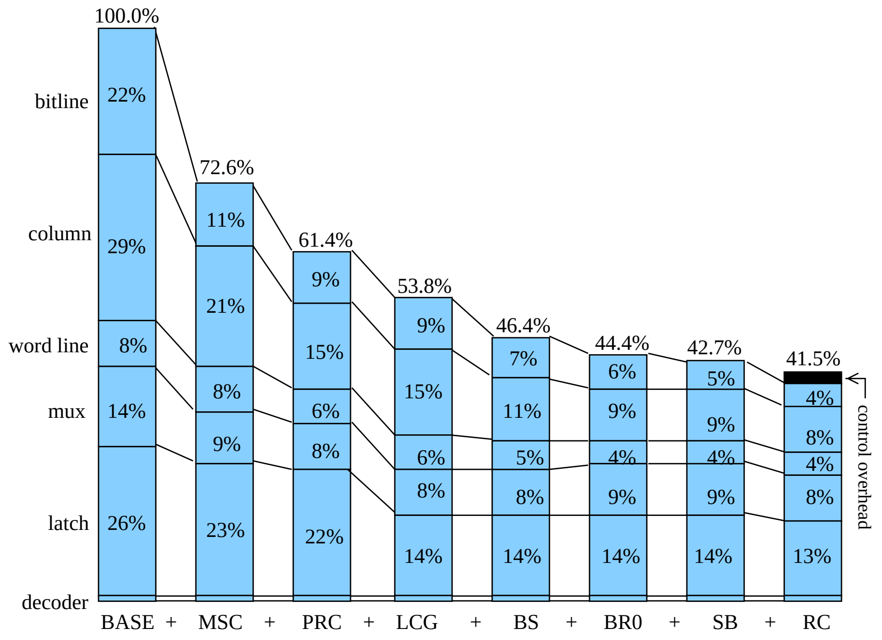

Figure 7: Effect of combining techniques.

The final two techniques (split-bitlines and read caching) are the most complex and give the least benefit when combined with the other methods. These results illustrate the importance of considering the overlap of various energy-reduction techniques when applied to the same problem. The final breakdown also shows that further effort to reduce bitline energy (for example, differential read ports [9]) will yield little overall energy improvement.

12. Conclusions

We have evaluated seven techniques to reduce register file access energy by simulating large benchmark program runs. The overall saving was up to a factor of 2.4 over the base case design. The final energy breakdown in the register file shows less than 10% of the power due to bitline activity, indicating that further work to reduce bitline energy will have limited impact. Further savings might be achieved with new column circuitry. It appears difficult to reduce energy in the muxes and latches at the micro-architectural level because required operands must ultimately pass through this path, although circuit techniques might reduce energy further.

13. Acknowledgments

This work was partially funded by an NSF graduate fellowship.

References

[1] N. Nishi *et al.* A 1GIPS 1W single-chip tightly-coupled four-way multiprocessor with architecture support for multiple control flow execution. In *2000 IEEE International Solid-State Circuits Conference*, February 2000.

[2] D. R. Gonzales. Micro-RISC architecture for the wireless market. *IEEE Micro*, 19(4):30–37, July/August 1999.

[3] A. Kalambur and M. J. Irwin. An extended addressing mode for low power. In *Proceedings of the IEEE Symposium on Low Power Electronics*, pages 208–213, August 1997.

[4] G. Kane and J. Heinrich. *MIPS RISC Architecture (R2000/R3000)*. Prentice Hall, 1992.

[5] J. Ousterhout, G. Hamachi, R. Mayo, W. Scott, and G. Taylor. Magic: A VLSI Layout System. *Proc. 21st Design Automation Conference*, pages 152–159, 1984.

[6] V. Stojanović and V. G. Oklobdžija. Comparative analysis of master-slave latches and flip-flops for high-performance and low-power system. *IEEE Journal of Solid-State Circuits*, 34(4):536–548, April 1999.

[7] J. Tseng. Energy-efficient register file design. Master's thesis, Massachusetts Institute of Technology, December 1999.

[8] N.P. van der Meijs and A.J. van Genderen. SPACE Tutorial. Technical Report ET-NT 92.22, Technical Report, Delft University of Technology, Netherlands, 1992.

[9] V. Zyuban and P. Kogge. Split register file architectures for inherently low power microprocessors. In *Power Driven Microarchitecture Workshop at ISCA98*, Barcelona, Spain, June 1998.