Energy-Efficient Register File Design

by

Jessica Hui-Chun Tseng

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrial Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 1999

Author . . . .
Department of Electrical Engineering and Computer Science
December 22, 1999

Certified by . . . .
Krste Asanović
Assistant Professor
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Energy-Efficient Register File Design

by

Jessica Hui-Chun Tseng

Submitted to the Department of Electrical Engineering and Computer Science
on December 22, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrial Engineering and Computer Science

Abstract

In this thesis, we evaluate five techniques for energy efficient register file design by studying dynamic traces of SPECInt95 and Powerstone benchmarks. A single-issue MIPS RISC microprocessor with a five-stage pipeline is used for this study. The five proposed techniques are precise read control, bypass skip, separate R0, modified storage cell, and split bitline. Their potential energy savings are examined through an energy dissipation model. On average, each technique shows 23%, 20%, 22%, 38%, and 20% respectively. An energy saving of 63% can be achieved by combining these five low power register file design methods without changing existing software.

Thesis Supervisor: Krste Asanović
Title: Assistant Professor

Acknowledgments

First, I thank my thesis supervisor, Krste Asanović, for supporting me throughout this work and for his advice and encouragement. Also, I like to thank him for giving me such a great opportunity to work with him on development of energy-efficient microprocessor architectures.

I thank Mukaya Panich, Seongmoo Heo, Ronny Krashinsky, Albert Ma, and Jonathan Babb for technical discussions and help with my thesis. Thanks to my officemate, Jason Miller for help using the LaTeX text formatter and many other tools. Thanks to all the people who have let me run benchmark simulations on their computers.

Special thanks to all my friends who have made my life in MIT so enjoyable and intellectual. Thank you my best friend, Danny, for reducing my stress level in every possible ways. Thanks to my brother, David, for balancing my graduate life at MIT with his funny jokes and stories.

I would like to thank MIT Laboratory for Computer Science (LCS) for providing the environment for this work. Particular thanks to NSF Graduate Research Fellowship for funding part of this work.

Finally, thank you my beloved parents for all the support and for making all this possible for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As we move into the next millennium, notebook computers and hand-held portable multimedia wireless devices such as Palm-Pilots and wireless-phones are becoming a major part of our daily life for personal management and communication functions. The hardware size and cost of such human-machine interface applications closely depends on the energy consumption of the designs. Therefore, designing an energy-efficient microprocessor for these systems is one of the major challenges for future designers.

Many studies have shown that register files represent a substantial portion of the energy budget in modern microprocessors [2, 4, 7, 19]. The circuitry and technology used to design register files influence the energy dissipation; however, neither technology scaling nor circuitry techniques prevents register files from being the dominant power component in modern microprocessors [19]. 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 [4].

The energy dissipation of a CMOS integrated circuit is proportional to the switching activity frequency and switching capacitance of the transistors and the bus lines [13]. In this research, we evaluate techniques to reduce register file energy by lowering the switching activity and the bitline capacitance in the register file. Dynamic profiles of different benchmarks are analyzed and studied to suggest modifications to the conventional register file to reduce the register file switching activity and bitline capacitance while maintaining the same level of performance.

The microprocessor studied here is a single-issue RISC processor for high performance embedded applications. In these microprocessors, register files perform two register reads and one register write per cycle. Every register read and register write contributes switching activity to the register file. However, 58% of fetched operands are discarded while 39% of register file updates are redundant. Therefore, this thesis evaluates three techniques to eliminate unnecessary register accesses, one technique to lower bitline switching activity, and one technique to decrease bitline switching capacitance. These five techniques are listed below:

1. **Precise Read Control** Different instruction types require different numbers of source operands. We can reduce register file read accesses by only fetching the necessary operands.

2. **Bypass Skip** In the case of an operand read-after-write (RAW) pipeline hazard [5], an operand value is supplied by the bypass network instead of read from the register file. We can further reduce register file read accesses by not fetching stale operands from the register file.

3. **Separate Register 0** RISC machines usually have a single register R0 whose value is always zero [8]. We can provide the R0 value in the bypass network instead of fetching it from the register file to reduce read and write accesses to the register file.

4. **Modified Storage Cell** We can take advantage of the asymmetry in one and zero distribution of register values to modify the register storage cell to minimize the number of high-to-low and low-to-high bitline transitions.

5. **Split Bitline** Due to the compiler's register allocation policy, some registers are used much more frequently than others. We can split the register file into the frequently used registers and the remaining registers to reduce average bitline capacitance.

Each technique's potential energy saving is validated through an energy estimation model for single-ended read register files. The effects of combining differential sense amplifiers with these techniques for double-ended read register files are discussed in the last chapter.

This thesis is organized as follows: Chapter 2 describes and reviews the methodology used. Chapters 3-7 discuss the five low energy register file design proposals and their potential savings and possible disadvantages. Chapter 8 concludes, summarizes, and discusses the contribution of this thesis and suggests future work.

Chapter 2

Methodology

The method used to develop a low power register file in this research was to use dynamic benchmark traces to evaluate modifications to a conventional register file for energy efficiency. A similar methodology was proposed and used in the development of Motorola's low-power Micro-RISC architecture for the wireless market [14]. A MIPS RISC microprocessor [2] was used for this study. From the study of the dynamic traces, different low power register file designs were proposed and their energy savings were validated and compared through an energy dissipation model.

2.1 Dynamic Profiling of Benchmarks

To obtain benchmark traces, a cycle-accurate simulator is used to simulate the microarchitecture-level behavior of a MIPS RISC scalar microprocessor with a five-stage pipeline as shown in Figure 2-1. The MIPS processor executes the MIPS-II ISA. This simulator only traces user level instructions and records register file access information, instruction operands' bypass frequency, and critical data value switching activity. The five-stage pipeline has a single cycle memory system, zero cache misses, one 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.

A combination of SPECInt95 benchmarks and Powerstone benchmarks [14] are used as a workload. Table 2.1 lists the benchmarks and input data sets used in this research, and

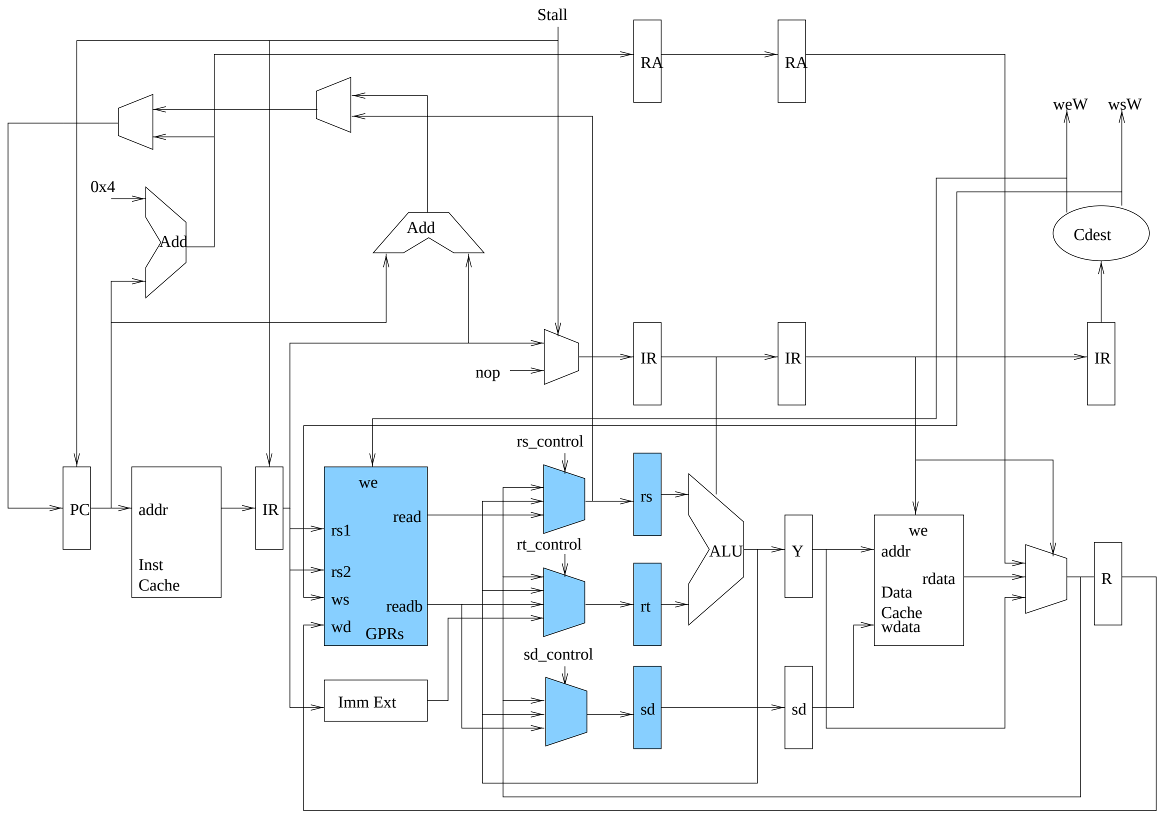

Figure 2-1: Five-stage pipeline datapath.

shows total instruction and cycle counts. Each benchmark was run to completion. This study only covers predominantly integer benchmarks and Table 2.2 shows the distribution of instruction types in each benchmark. Each benchmark is compiled with gcc version 2.7.0 with -O3 optimization and linked with the newlib standard C library.

2.2 Energy Estimation Model

The energy estimation model evaluates only the energy consumption of the register file and the bypass network in the instruction decode stage, shown as the shaded region in Figure 2-1. This energy estimation model is an activity sensitive model [10, 3] because the energy dissipation is proportional to the frequency of switching activity of the transistors and bus lines. The energy dissipation caused by each low-to-high and high-to-low transition activity

Benchmark {Data Set}	Instruction Count (Millions)	Cycle Count (Millions)	Description
m88ksim {test}	519	567	A chip simulator for Motorola 88100 microprocessor
li (test)	997	1,129	xlisp interpreter
go {train}	579	631	An internationally ranked go-playing program
gcc {ref:2c-decl-s}	1,396	1,524	Based on the GNU C compiler version 2.5.3.
vortex {test}	10,054	11,123	An object oriented database
jpeg {test:specmum.ppm}	567	710	JPEG 24-bit image compression /decompression standard
g721 {clinton.g721}	528	625	Adaptive differential PCM for voice compression
Average	2,093	2,330	

Table 2.1: Benchmark descriptions, instruction counts, cycle counts, and input types.

Benchmark	Load Store	ALU Imm	ALU R-type	Multi. Divide	Jump Branch	Shift	Nop	Floating Point
m88ksim	24.68	29.88	18.47	0.04	21.43	2.17	3.33	0.00
li	43.69	14.90	9.79	0.00	21.79	0.80	8.97	0.06
go	27.71	24.01	21.70	0.06	13.46	10.94	2.11	0.00
gcc	36.85	21.21	14.60	0.18	18.24	4.40	4.48	0.04
vortex	47.66	14.75	15.39	0.13	16.03	2.02	4.01	0.00
jpeg	25.08	20.39	28.01	2.51	9.26	13.61	1.12	0.01
g721	16.55	26.29	15.58	1.12	20.46	13.59	6.30	0.08
Average	31.75	21.63	17.65	0.58	17.24	6.79	4.34	0.03

Table 2.2: Benchmark instruction type % distribution.

in a full rail-to-rail swing CMOS circuit is equal to

$$\frac{1}{2} \cdot C_{switch} \cdot V_{dd}^2$$

Where C_{switch} is the switching capacitance and V_{dd} is the supply voltage. Therefore, the average energy consumption of each functional block per CPU cycle is computed as follows:

$$E = \sum_r (\frac{1}{2} \cdot f_r \cdot C_{switch\_r} \cdot V_{dd}^2)$$

Where f_r is the average data transition frequency of the node r within the functional block as determined by the dynamic benchmark profiling. The C_{switch_r} is the switching capacitance related to node r.

The parameters used in this energy estimation model are based on a 0.6-μ n-well CMOS process technology with 3.3V power supply and two layers of metal. The design of register file and bypassing network is based on the T0 design [1] and is laid out using Magic [12]. The layout-to-circuit extraction tool, Space [17], is used to extract a circuit netlist for further circuit simulation. Space extracts capacitance to the substrate, fringe capacitance, crossover coupling capacitance, and capacitance between parallel wires. Hspice [11], a circuit simulator, is used to simulate the circuit netlist generated from Space and to determine the effective switching capacitance, C_{switch}, for the energy estimation model. The register file and the bypass network designs used in this energy model are described in the following subsections. A base-case scenario energy estimation analysis is illustrated in the summary subsection.

2.2.1 Register File

The regfile used in this research is a high performance dynamic regfile with two read ports and one write port. This design provides both read and write access in the same cycle. During the first half of the cycle, the read bitlines are precharged high and the write bitlines are driven. Registers are written during the first half of the cycle while the read data is

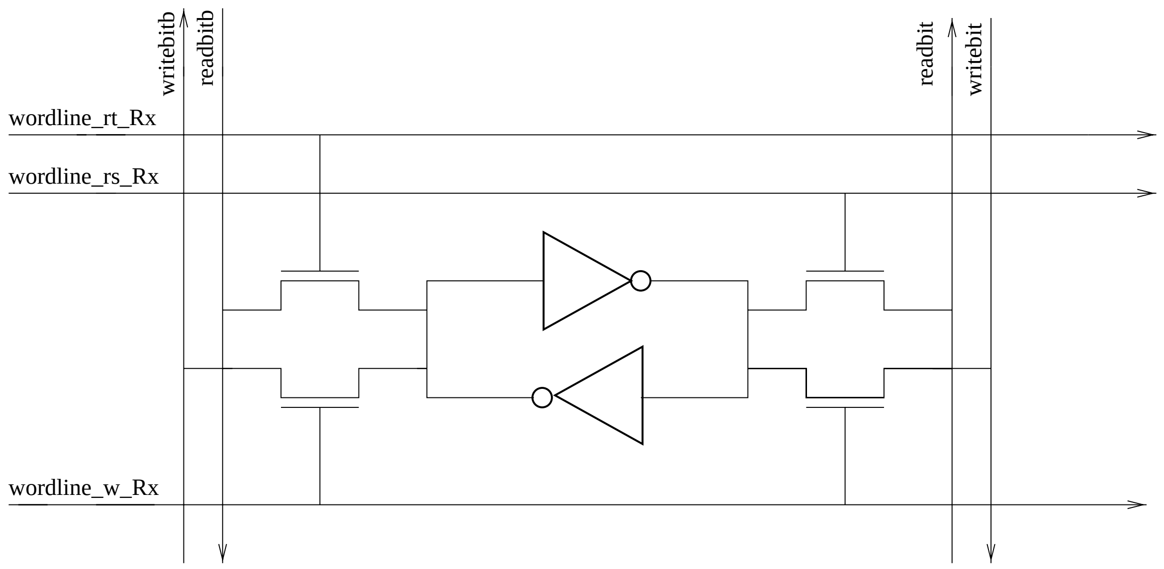

Figure 2-2: Register file storage cell.

sensed during the second half of the cycle. This avoids a bypass path from the write-back stage of the five-stage pipeline microprocessor as shown in Figure 2-1.

It is observed that the energy dissipation of the read and write bitlines dominates the regfile energy consumption. Therefore, the energy estimation model for the regfile is based on the transition activity of read bitlines and write bitlines. The address decoding of regfile is not included in this energy estimation model. The regfile consists of a 32x32 matrix of storage cells, Figure 2-2, for the 32 32-bit-wide registers with a column circuitry module, Figure 2-3, at the end of each bitline. The storage cell is a conventional static RAM cell [18]. The column circuitry consists of a clocked inverter sense amplifier to provide faster read port output sensing and it also controls the read bitline precharges, write drive, and data buffering. The switching capacitance of the read bitlines, write bitlines, and precharging transistors of the regfile for one of the 32 bit slices is shown in Table 2.3.

2.2.2 Bypass Network

The bypass network consists of two three-input muxes, one four-input mux, and three latches. Transmission gate muxes are used in this design. Figure 2-4 shows the design for a

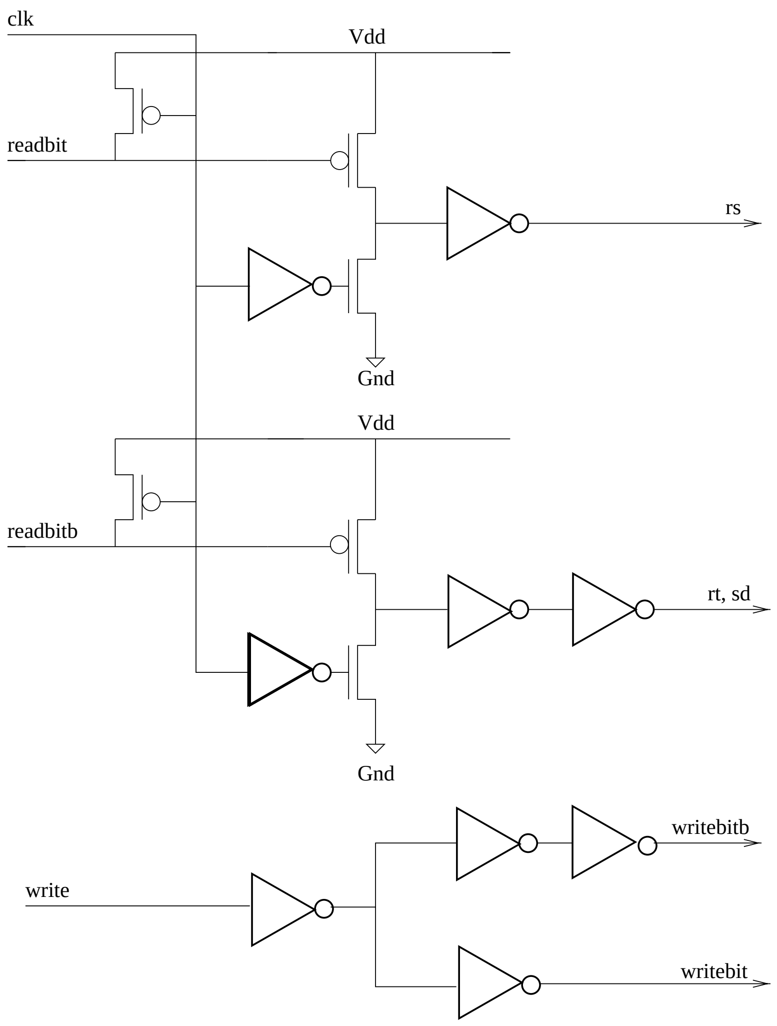

Figure 2-3: Column circuitry for one bit slice.

Switch Capacitor	Capacitance (Unit: fF)
$C_{switch\_readbit}$	304
$C_{switch\_readbitb}$	331
$C_{switch\_wbit,wbitb}$	679
$C_{switch\_precharge}$	50

Table 2.3: Register file switching capacitance.

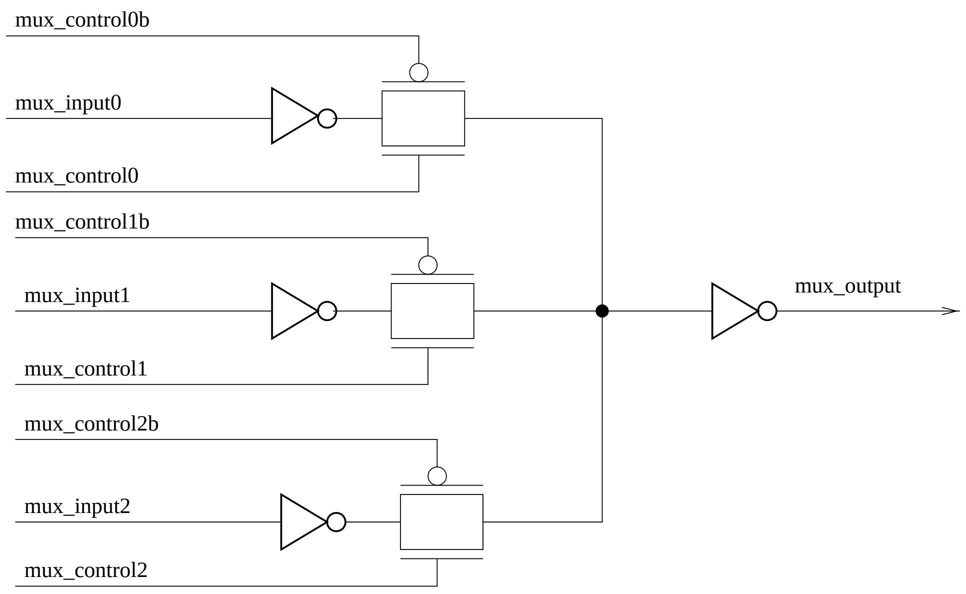

Figure 2-4: Three-input transmission gate mux.

three-input mux. The latches in this bypass network are similar to the IBM PowerPC603MS latch designs [15], Figure 2-5. The bit-slice switching capacitance of mux input, mux output, mux control line, latch data value, latch clock input is listed in Table 2.4.

Switch Capacitor	Capacitance (Unit: fF)
$C_{switch\_mux3input}$	7.3
$C_{switch\_mux3output}$	22.3
$C_{switch\_mux3control}$	2.0
$C_{switch\_mux4input}$	7.4
$C_{switch\_mux4output}$	24.6
$C_{switch\_mux4control}$	2.0
$C_{switch\_latchdata}$	62.0
$C_{switch\_latchclk}$	19.3

Table 2.4: Bypass network switching capacitance.

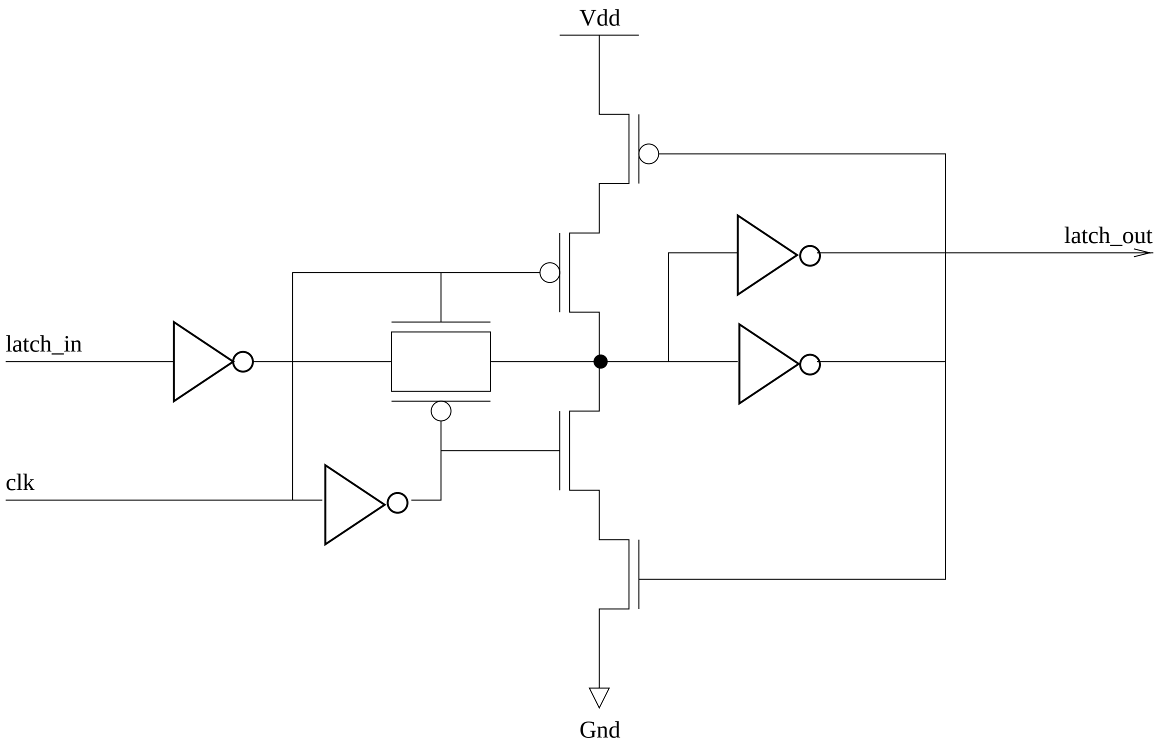

Figure 2-5: Latch, similar to IBM PowerPC603MS latch.

Data Connection Node r	Switch Capacitance (fF)	Transition Frequency	Energy Dissipation (fJ)
Read-bit	304	1.6753	2776
Read-bitb	331	0.2265	408
Write-bit,bitb	679	0.1710	632
Rs-mux-input	7	2.0270	81
Rs-mux-output	22	1.2893	157
Rs-mux-control	2	0.7499	8
Rs-latch-data	62	1.2893	435
Rt-mux-input	7	0.7662	31
Rt-mux-output	25	0.2093	28
Rt-mux-control	2	0.7898	9
Rt-latch-data	62	0.2093	71
Sd-mux-input	7	0.5781	23
Sd-mux-output	22	0.1953	24
Sd-mux-control	2	0.6946	8
Sd-latch-data	62	0.1953	66
Precharge	50	2.0000	540
Clk	58	2.0000	631
Total	1705		5927

Table 2.5: The average bit slice energy consumption analysis for g721 base case.

2.2.3 Summary

To calculate the total energy per cycle, the energy estimation model sums up all the energy dissipation per cycle due to data value transitions. Table 2.5 is an example to show how the energy model calculates the total energy per cycle for the g721 benchmark in the base case scenario. The base case scenario is a common simple regfile, which always performs one write and two reads per cycle regardless of the instruction opcode and pipeline state. The average bit-slice energy consumption per cycle is 5.9 pJ and the total energy consumption per cycle for the 32-bit wide datapath is 190 pJ.

Chapter 3

Precise Read Control

3.1 Motivation

Only instructions such as register-register arithmetic, store, conditional-branch, and shift instructions require fetching both source operands. Therefore, a large fraction of source operand data is discarded because of over fetching of operands from the regfile. Over fetching operands creates extra unnecessary regfile switching activity contributing to the energy consumption. The benchmark profiling shown in Figure 3-1 shows on average, each instruction requires 1.3 source operands; 70% of dynamic instructions require only one source operand. So, a precise-read-control regfile has an potential of decreasing the regfile read activity by 35%.

3.2 Implementation

One of the most straightforward implementations of precise read control is by adding an opcode pre-decoder prior to the wordline drivers in the regfile as shown in Figure 3-2. When the wordline is not enabled, the read bitline value retains its precharged value and no switching occurs. We also keep the precharge transistors turned on to avoid switching their gate capacitance. The precise-read-control regfile has only an AND-gate area overhead because the opcode pre-decoders are part of the original bypassing interlock circuit. There is no latency overhead if the opcode pre-decoder utilizes the first half of the cycle to finish

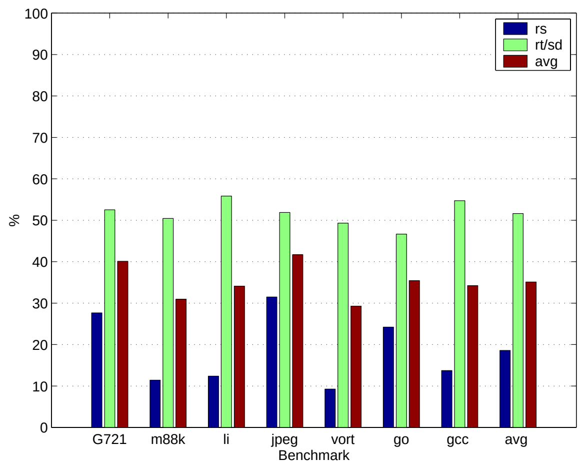

Figure 3-1: Percentage of discarded operands due to over fetching.

performing all its necessary decoding and is able to provide the issue signal in time for the read bitline enables in the second half of the cycle. However, if the opcode decoding cannot finish in the first half of the cycle, precise read control is going to add latency to the regfile.

The precise-read-control regfile handles NOP instructions differently from shift left logical (SLL) instructions even though they have the same opcode. Since NOP instructions do not require any operands, the opcode decoders disables both read operand fetches.

3.3 Results

Figure 3-3 shows the energy savings of precise-read-control in comparison with the base case scenario. The energy saving ranges from 16% to 31% across benchmarks with an average of 23%.

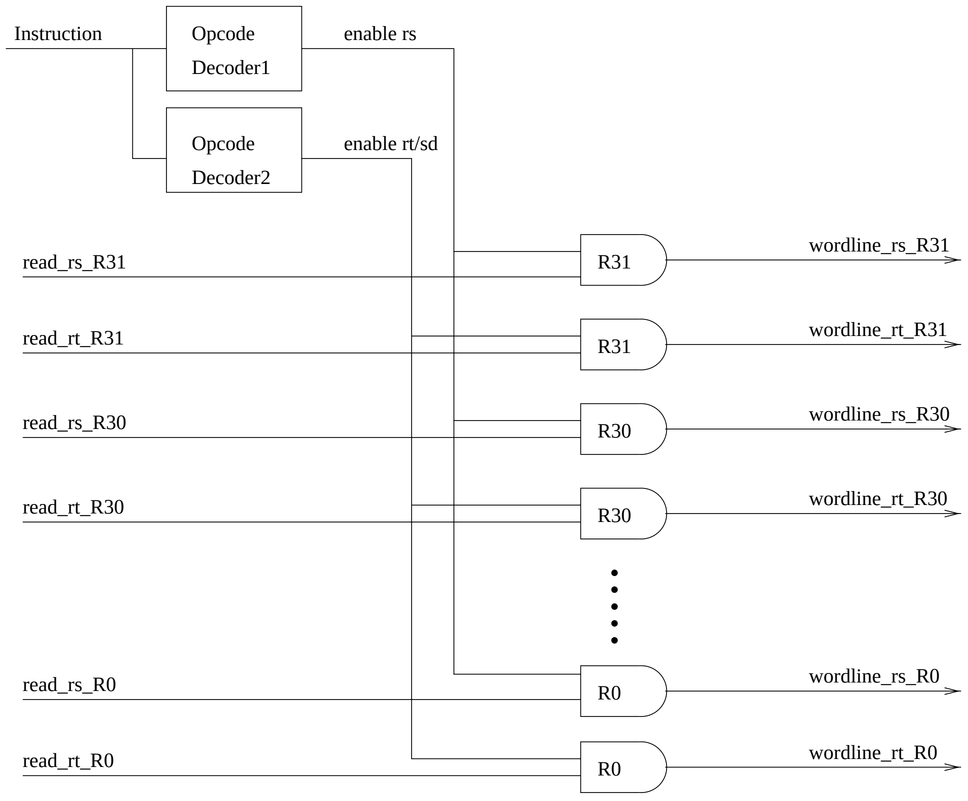

Figure 3-2: Implementation of precise read control.

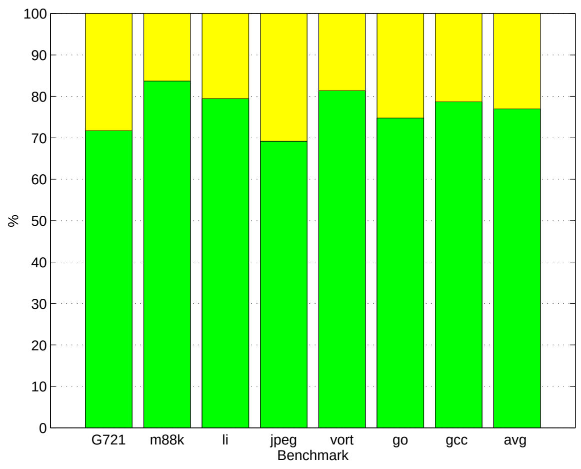

Figure 3-3: Comparative energy consumption for the base case regfile and the precise-read-control regfile.

Chapter 4

Bypass Skip

4.1 Motivation

Fetching stale values from the regfile creates unnecessary switching activity in the regfile if the fetched values are discarded because of bypassing. Simulation data, Figure 4-1, shows an average of 26% of the operands (base case scenario without precise read control) are bypassed from other stages of pipeline instead of the regfile. Therefore, a bypass-skip regfile is expected to decrease the regfile read bitline activity by 26%.

4.2 Implementation

The bypass-skip regfile determines the RAW pipeline hazard prior to fetching values from the regfile. It detects the RAW pipeline hazard and disables the wordline drivers of the regfile as shown in Figure 4-2. The bypass skip technique only contributes an AND-gate area overhead per wordline to the regfile for a fully-bypassed datapath. A fully-bypassed datapath already contains a RAW hazard detector which directs the most recent operand values from the bypass network to the source registers to avoid a pipeline stall. The only change in the original design is moving the existing RAW hazard detector forward in the pipeline prior to the regfile read bitline enablers. Therefore, a latency penalty can occur if the RAW hazard detector takes longer than the first half of the cycle to finish hazard detection. If the latency is too long, we can even consider adding an extra pipeline stage

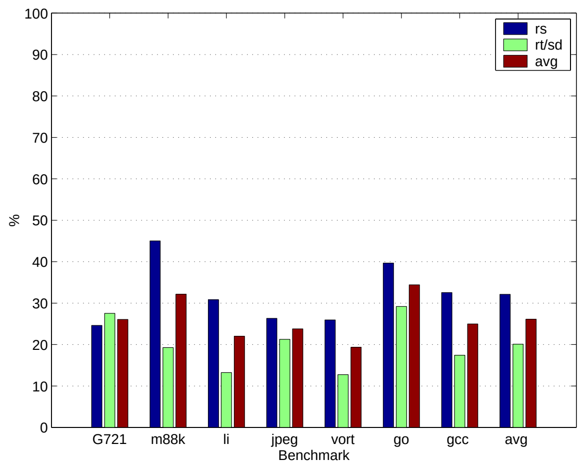

Figure 4-1: Percentage of operands supplied by the bypass network.

for hazard detection to have the same throughput [5]. When the wordline is disabled, the read bitline remains in its precharged state with no switching activity.

4.3 Results

Figure 4-3 shows the energy consumption of the bypass-skip regfile and the bypass network in comparison with the base case scenario. The energy saving ranges from 14% to 29% across different benchmarks with an average of 20%. The energy saving is proportional to the number of source operands being supplied by the bypass network.

To determine the benefits from bypass skip after applying precise read control, we find out that 36% of the necessary operands are supplied by the bypass network, Figure 4-4. Therefore, the bypass-skip method has a potential of decreasing regfile read accesses by a further 36% after applying the precise read control method. Note, this is greater than for the base case because we remove many R0 fetches which are never bypassed. When combining these two methods, the wordline still has only a single AND gate area overhead because the two enabling signals can be combined.

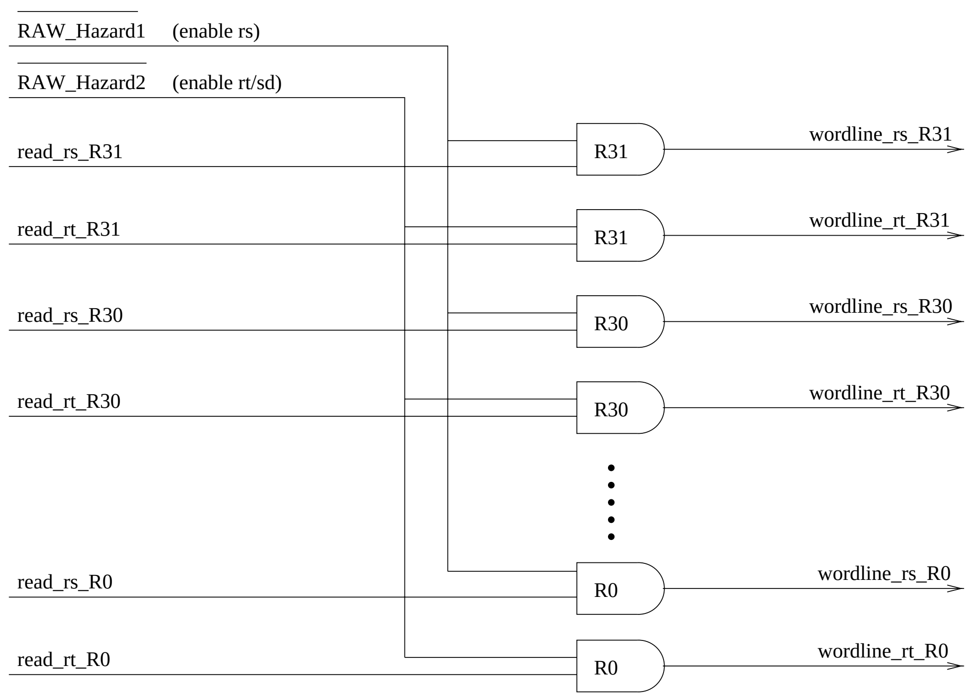

Figure 4-2: Bypass-skip regfile implementation.

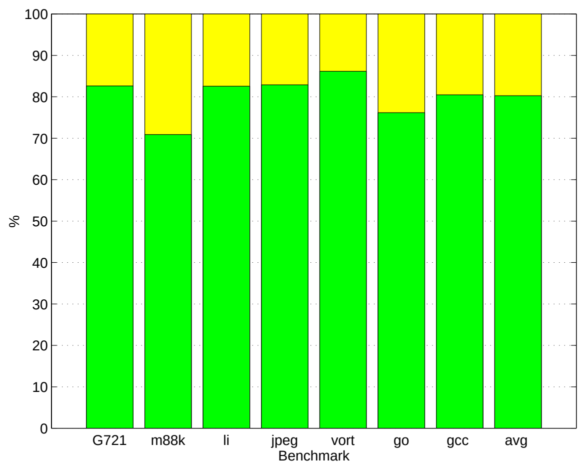

Figure 4-3: Comparative energy consumption for the base case regfile and the bypass-skip regfile.

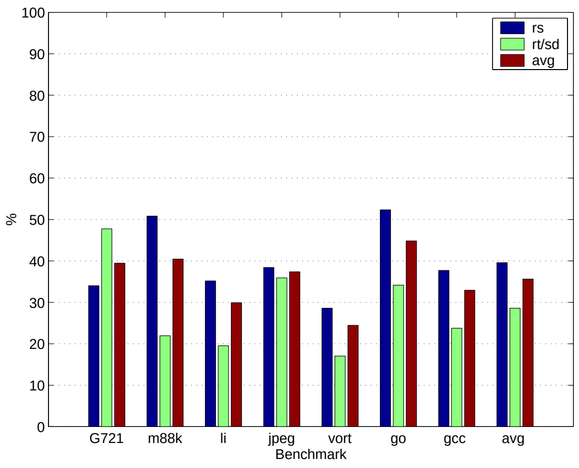

Figure 4-4: Percentage of necessary operands that are supplied by the bypass network.

Chapter 5

Separate Register 0

5.1 Motivation

Register 0 (R0) always retains its fixed zero value and can be determined without accessing it from the regfile. Therefore, we can provide R0 value in the bypass network instead of fetching it from the regfile to reduce read and write accesses of the register file. The switching activity of regfile read bitlines and write bitlines are decreased by reducing the number of read and write accesses. The separate R0 technique also eliminates R0 from the regfile to reduce its size from 32 registers to 31 registers, thereby decreasing its bitline capacitance slightly.

R0 is the most frequently referenced register throughout all the benchmarks. Figure 5-1 shows that 40% of regfile write accesses and 25% of regfile read accesses are R0. Moving R0 out of the regfile can greatly reduce its read and write accesses.

The inverted read bitlines discharge their pre-charged value only when the stored value is high while the non-inverted read bitlines discharge only when the stored value is low. Therefore, separate-R0 regfiles can only improve the non-inverted read bitlines switching activity. However, all of the regfile bitline capacitance is reduced slightly regardless.

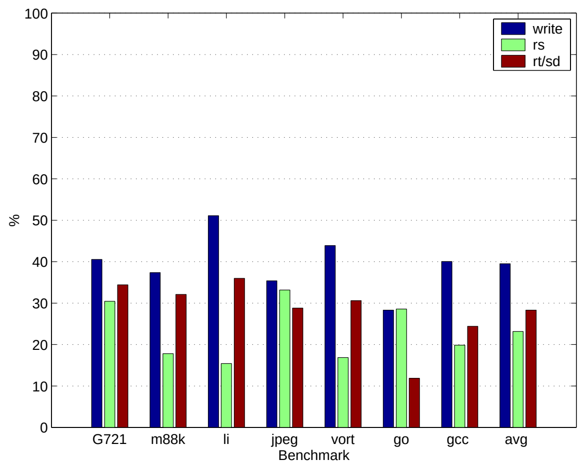

Figure 5-1: Percentage of R0 usage.

Switch Capacitor	Capacitance (Unit: fF)
$C_{switch\_readbit}$	299
$C_{switch\_readbitb}$	325
$C_{switch\_wbit,wbitb}$	666

Table 5.1: Regfile bitlines switching capacitance for 31 registers.

5.2 Implementation

To implement the separate-R0 regfile, one has to modify the circuitry of the regfile and the bypass network as shown in Figure 5-2. The size of the regfile decreases from 32 registers to 31 registers and its bitlines become shorter with lower switching capacitance as shown in Table 5.1. R0 value is provided by the grounded input of muxes. The mux for operand rt changes from a four-input mux to a five-input mux while the muxes for operand rs and sd change from three-input muxes to four-input muxes. The bit-slice switching capacitance of the five-input mux used in the energy estimation model is listed in Table 5.2. Also, we now disable the wordlines when the destination register is R0 to eliminate unnecessary write bitline switching activity.

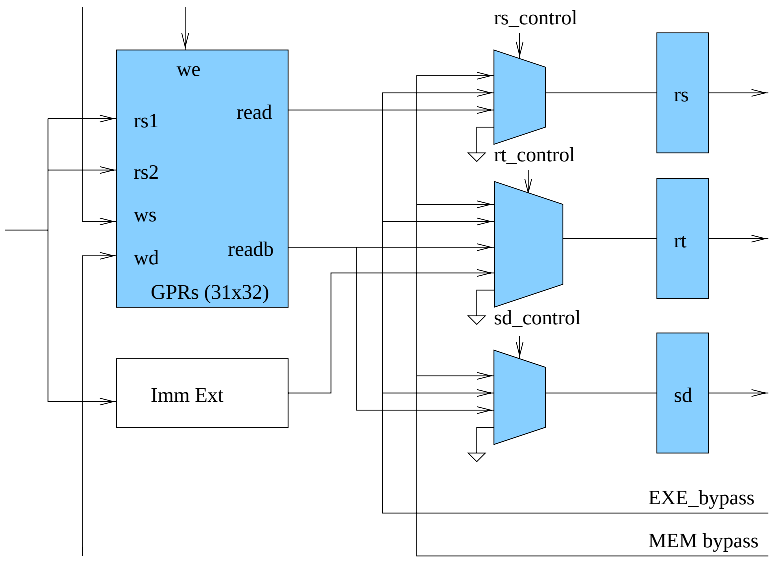

Figure 5-2: Separate-R0 regfile implementation.

Switch Capacitor	Capacitance (Unit: fF)
$C_{switch\_mux5input}$	7.5
$C_{switch\_mux5output}$	30.7
$C_{switch\_mux5control}$	2.0

Table 5.2: Five-input mux switching capacitance.

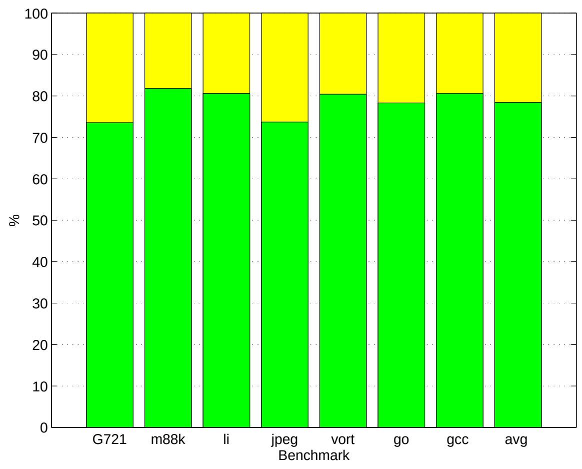

Figure 5-3: Comparative energy consumption for the base case regfile and the separate-R0 regfile.

5.3 Results

Figure 5-3 shows the energy saving of the separate-R0 regfile in comparison with the base case scenario across benchmarks. The energy saving ranges from 18% to 26% with an average of 22%. The energy saving is proportional to the read usage and the write usage of R0. However, write bitlines only contribute 3% energy saving due to the small reduction in their already low switching activity. Figure 5-4 shows that the separate-R0 regfile only reduces the write bitlines switching activity by 20% from its original 18% switching rate. The small write bitlines switching activity reduction is because only 12% of write bit values are ones.

The energy savings are not 100% additive when we combine the separate-R0 technique with the precise-read-control technique. The reason is that 78% of R0 references are caused by regfile over fetching. So, the energy saving of applying the bypass R0 method over the precise-read-control regfile is scaled down according to the percentage of R0 reference caused by over-fetching, Figure 5-5. On the other hand, the energy savings are 100% additive when we combine the separate-R0 method and the bypass-skip method because fetching R0 does not cause RAW pipeline hazards.

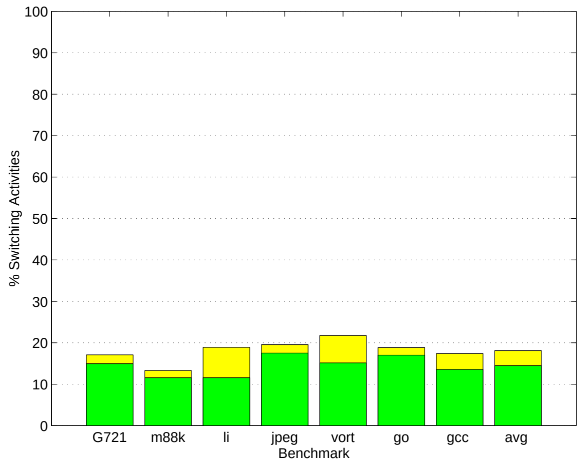

Figure 5-4: Write bitlines switching activity for regfiles with R0 and regfiles without R0 (dark).

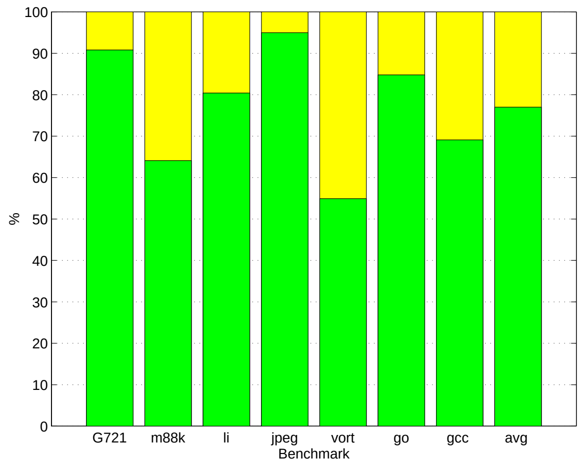

Figure 5-5: % of R0 reference caused by regfile over-fetching.

Chapter 6

Modified Storage Cell

6.1 Motivation

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 only when the storage cells cause them to discharge their precharged value. For example, the inverted bitlines are energy efficient when the majority of fetched bit values are low while the non-inverted bitlines are energy efficient for high bit values.

The benchmark traces show that 82% of the bits fetched from the regfile are zeros as shown in Figure 6-1. Therefore, the inverted read bitlines are four times more energy efficient than the non-inverted ones. Regfiles with only inverted read bitlines can reduce read bitline switching activity as much as 60% over the base case scenario. If both read ports use inverted bitlines, we need not include R0 explicitly. If no wordline is enabled, the regfile will return the required zero value.

6.2 Implementation

Figure 6-2 shows the most straightforward implementation of a regfile with only inverted read bitlines. The modified circuitry has no latency overhead but the asymmetry of the modified cell might contribute slight area overhead. The transistors have to be sized

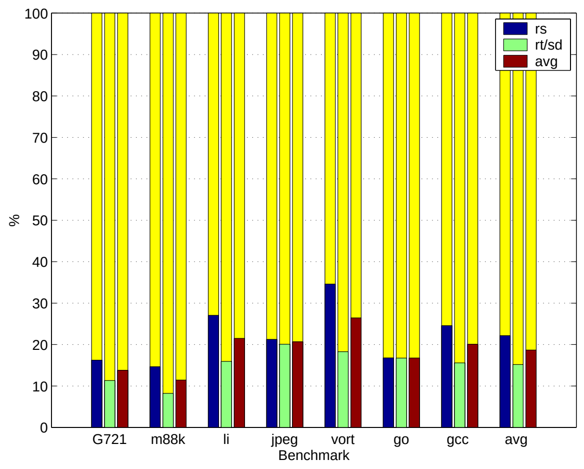

Figure 6-1: Percentage of 0 (light) and 1 (dark) bit value ratio.

carefully to avoid upsetting the stored value when both read ports access the same cell. To avoid the above complication, an inverter buffer can be added to the non-inverted end of the storage cells as in Figure 6-3.

6.3 Results

Figure 6-4 shows the energy consumption of the modified regfile and the bypass network in comparison with the base case scenario across benchmarks. The energy saving ranges from 23% to 50% with an average of 38%. The energy saving is proportional to the percentage of fetched zero bit values.

The energy saving from this modified-storage-cell regfile changes when we apply the precise-read-control method and the separate-R0 method. Figure 6-5 shows that the precise-read-control method decreases the low fetched bit value percentage from 82% to 78%, partly because many over fetches are from R0. Figure 6-6 shows that the separate-R0 method decrease the percentage of zero bit values from 82% to 75%. Therefore, the modified storage cell regfile energy saving over these two methods decreases slightly according to the new ratio of high and low fetched bit values.

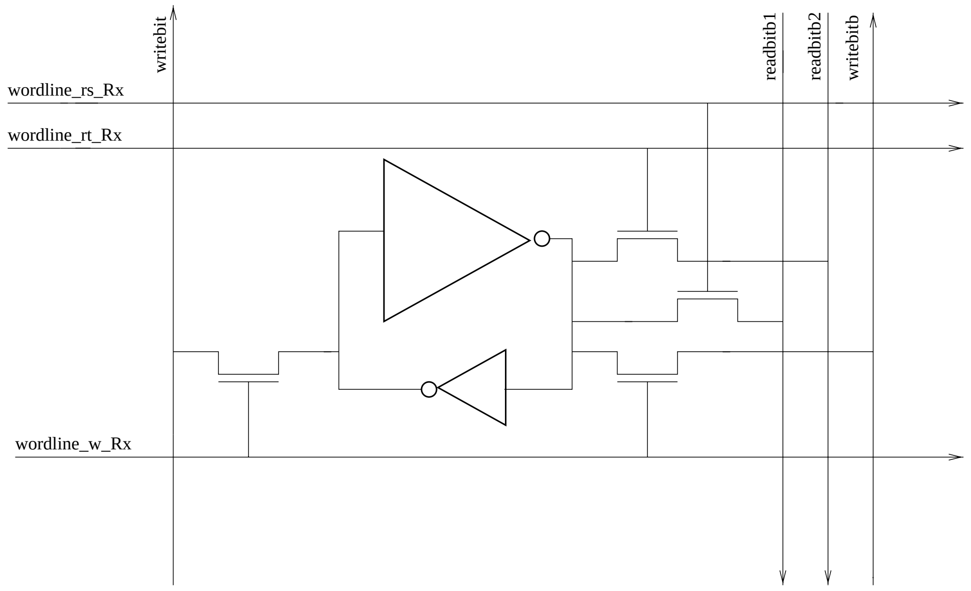

Figure 6-2: Modified storage cell implementation 1.

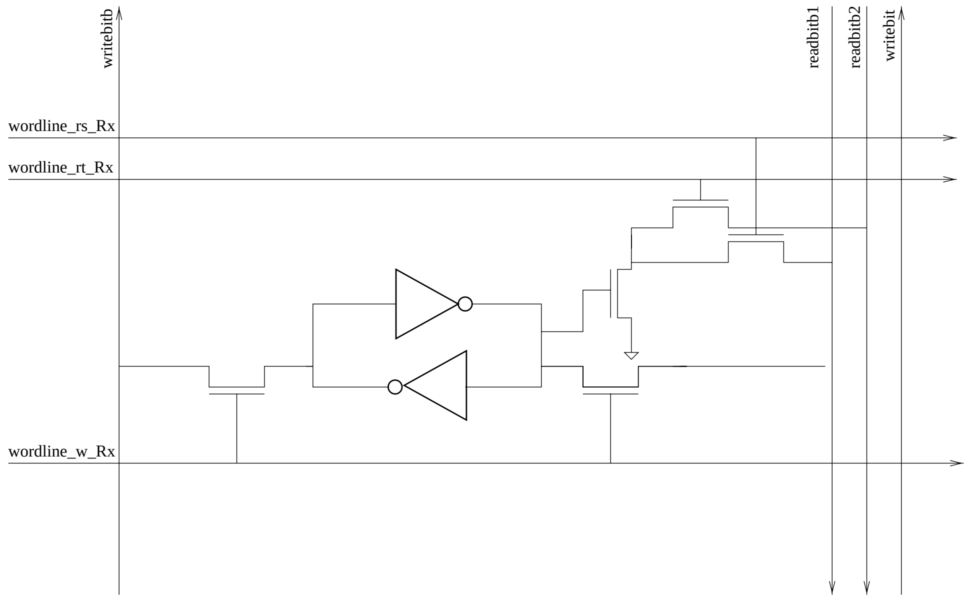

Figure 6-3: Modified storage cell implementation 2.

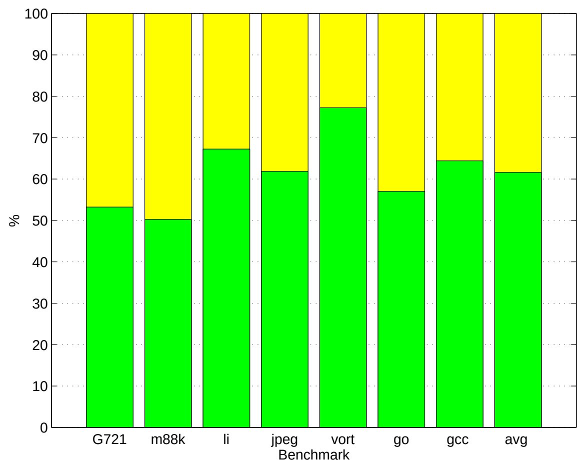

Figure 6-4: Comparative energy consumption for the base case regfile and the modified-storage-cell regfile.

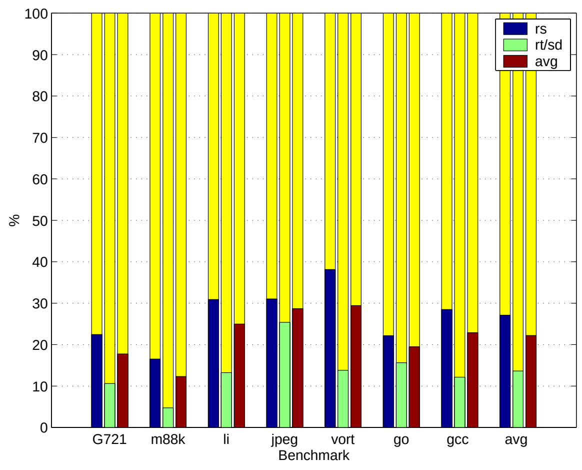

Figure 6-5: Percentage of 0 (light) and 1 (dark) bit value ratio with precise read control.

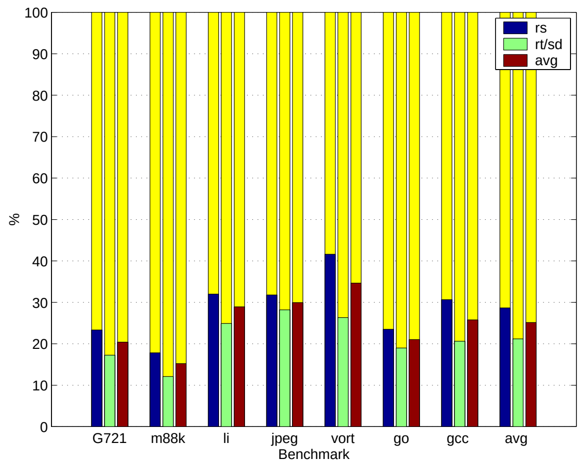

Figure 6-6: Percentage of 0 (light) and 1 (dark) bit value ratio with separate-R0.

Chapter 7

Split Bitline

7.1 Motivation

We can decrease the average bitline switching capacitance of a regfile by splitting the bitlines into segments and only accessing the segment which connects to the required register. In this thesis, we only investigate the use of a single bitline split. The split bitline method splits the regfile into two partitions. A transmission gate separates the two partitions. One partition holds the popular registers while the other holds the remaining registers. When the operand's reference is within the most popular registers, the transmission gate turns off the access to the remaining registers' partition. The gate is opened only when the operand's reference is not among the most popular registers. The split bitline regfile reduces the bitline energy by using an adaptive bitline length and is expected to decrease the regfile bitline energy by at least half.

Figure 7-1 shows that the 8 most popular registers account for 75% to 92% (average of 83%) of all regfile accesses while the most popular 10 registers account for 81% to 95% with an average of 88% of all regfile accesses. Moreover, the benchmark traces indicate that particular registers such as R0, R2, R3, R4, R5, R6, R16, and R29 always get accessed more frequently than others, Figure 7-2. According to the MIPSpro Assembly Language Programmer's Guide [6], 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. R29 contains the stack pointer.

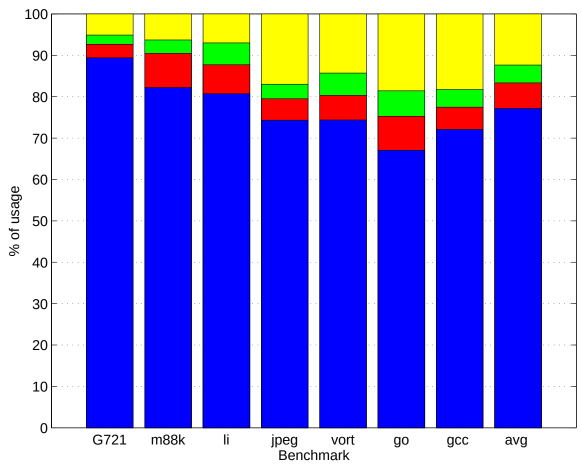

Figure 7-1: Most popular 6, 8, and 10 registers % usage.

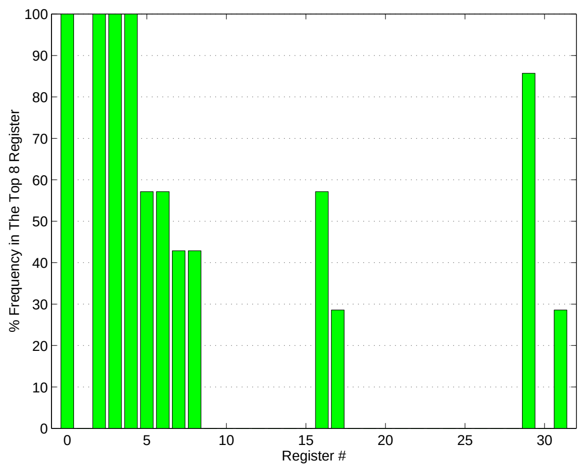

Figure 7-2: Frequently referenced registers.

7.2 Implementation

The main modification to the regfile is adding a transmission gate to split it into two partitions as shown in Figure 7-3. The addition of transmission gates is estimated to contribute only slightly to both access time and area. Read bitlines and write bitlines can have different ratios for their bitline partitions; however, the order in which registers line-up on the bitlines must be the same because we share storage cells between the bitlines.

The energy saving of the split bitline method comes from the regfile bitline energy reduction and is estimated by the following equation:

$$p_n \cdot E_{bn}$$

Where p_n is the percentage of regfile accesses to the small partition, n registers, and E_{bn} is the bitline energy difference between accessing from the small partition and the remaining partition. Therefore, the optimal size of the small partition is the size, n, which give the maximum energy reduction. Figure 7-4 shows the total energy saving percentage of the split bitline technique with different number of registers in the small partition. The benchmark traces indicate that the optimal size of the small partition for both read bitlines is 7 registers while the optimal size of the small partition for write bitlines is 5 registers.

We want to avoid having additional logic circuitry to control the bitline split gate. The regfile address decoder, Figure 7-5, can be used to control the bitline split gate. We can gate an 8 register partition by connecting one of the outputs of the 2-to-4 predecoders to the transmission gate. Figure 7-4 shows that using an 8/24 bitline partition for both reads and writes instead of the optimal sizes only decreases the energy saving by less than 1%. Therefore, a ratio of 8 to 24 registers bitline split is used for the implementation here due to the simplicity of the transmission gate's control circuitry and its high energy saving potential. Six (R0, R2, R3, R4, R5, and R6) of the eight most frequently accessed registers' wordlines are connected with the control line, X0. So, we can choose control line X0 to control the split bitline transmission gate. The other two most frequently accessed registers, R16 and R29, can be renamed with R1 and R7 before instruction fetch to keep these eight most popular registers together in one section which is gated by the control line,

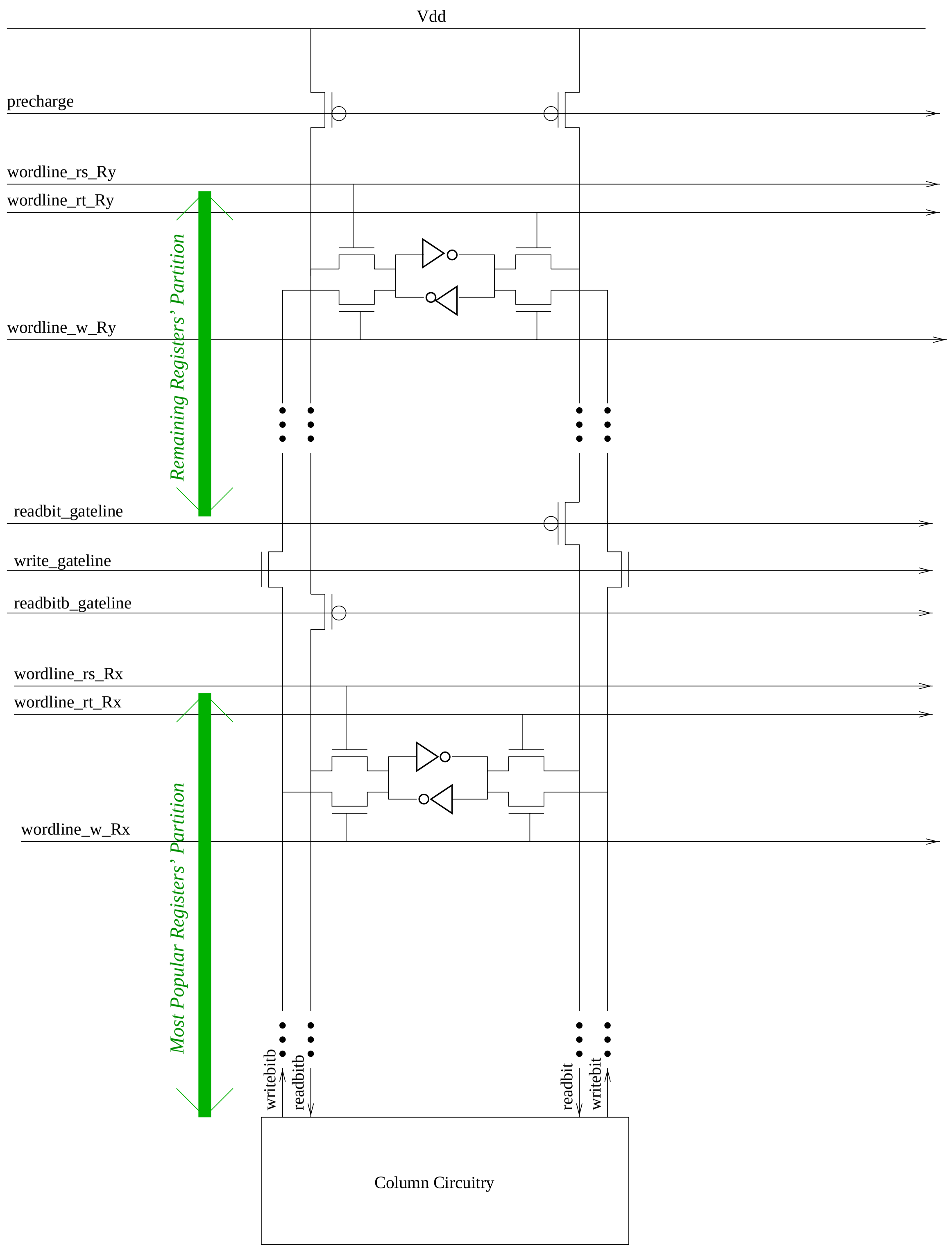

Figure 7-3: Split-bitline regfile implementation.

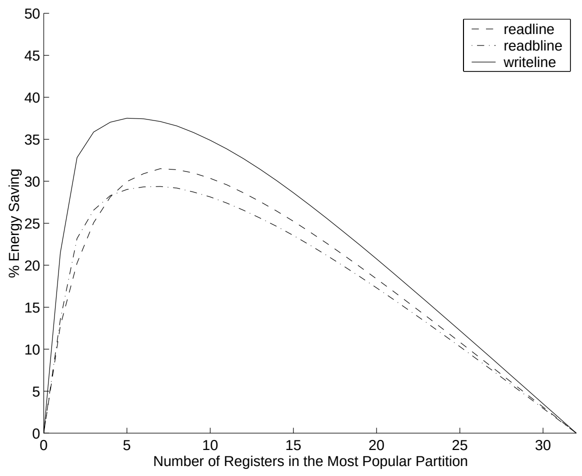

Figure 7-4: Optimal number of registers in most popular partition.

X0. Alternatively, we could change the software convention for register usage.

To determine the order in which registers line up in the most popular partition, the read bitlines and the write bitlines register usage are analyzed separately across benchmarks. Then, we weight the read bitlines and the write bitlines register usage percentage by their bitline capacitance and their bitline switching activity factor. The average register usage percentage can be calculated by adding the weighted register usage percentage of the read bitlines and the write bitlines. Lastly, the register usage order can be determined by ranking its average usage percentage. The efficiency of this ranking technique depends on the homogeneous and consistency of register usage percentage between benchmarks and between the read bitlines and the write bitlines. This ranking technique is efficient only when the read bitlines and write bitlines of all the benchmarks display a similar register usage density, as shown in Figure 7-6 for benchmark set.

Since the switching capacitance of a bitline is linearly proportional to its length, the switching capacitance of bitlines can be calculated using the equation listed in Table 7.1.

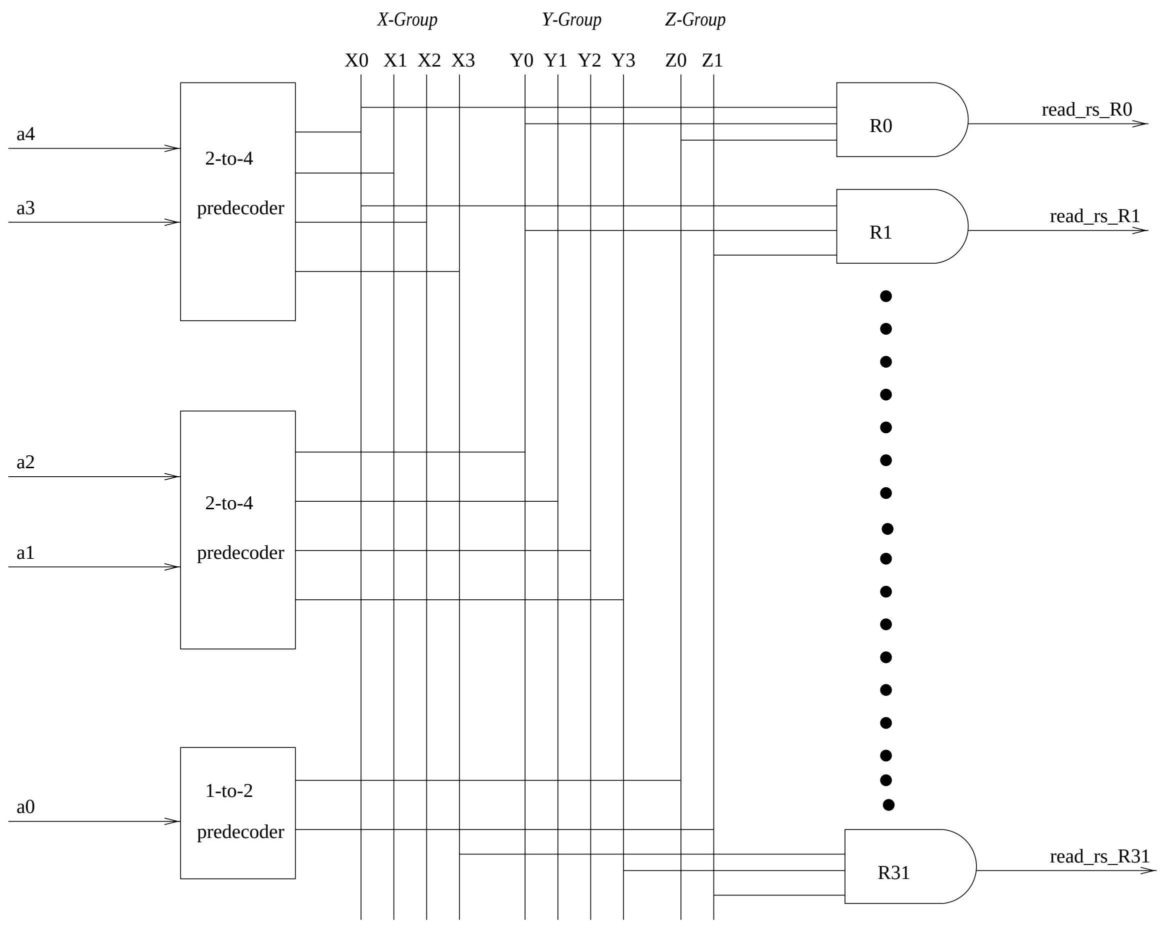

Figure 7-5: Regfile address decoder.

Switch Capacitor	Capacitance (Unit: fF)
$C_{\text{switch\_readbit}}$	$153 + 4.73 * n$
$C_{\text{switch\_readbitb}}$	$174 + 4.90 * n$
$C_{\text{switch\_wbit,wbitb}}$	$298 + 11.9 * n$

Table 7.1: Regfile bitline switching capacitance, where n is the number of registers on the bitline.

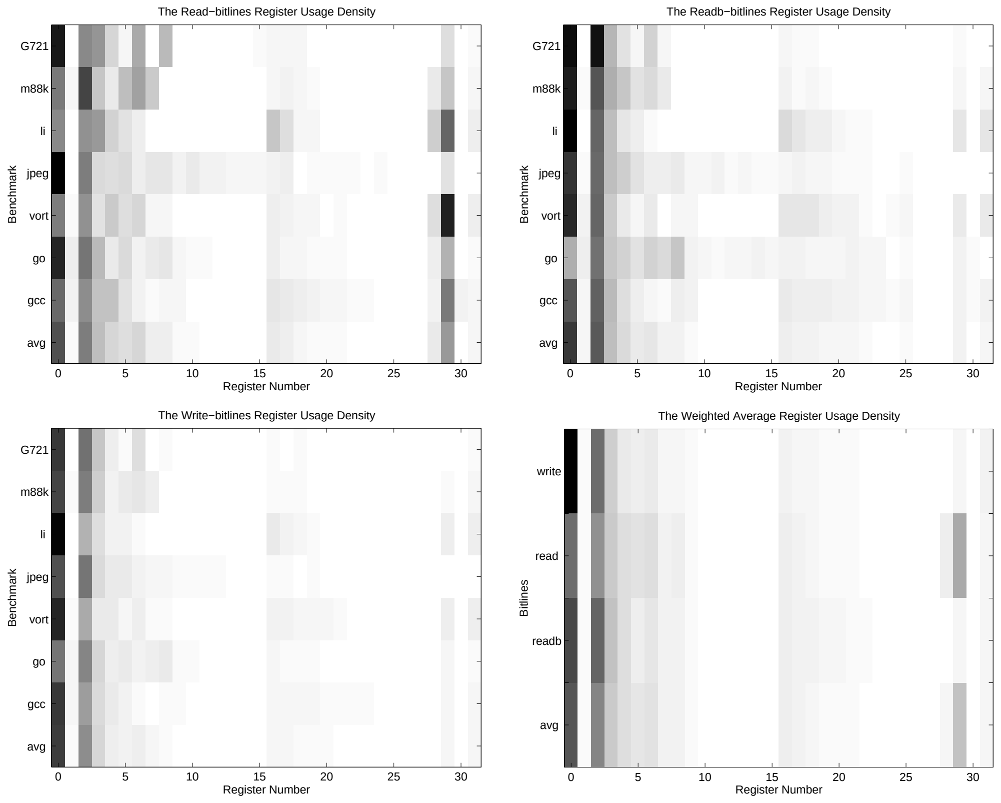

Figure 7-6: Register usage density distribution for the base case regfile, the usage density is proportional with the darkness of the color.

7.3 Results

Figure 7-7 shows the savings from split bitline in comparison with the base case. The total energy saving ranges from 19% to 21% with an average of 20%. The energy consumption of the column circuitry and the bypass network bounds the maximum energy saving to 33%. The energy saving for just the bitlines ranges from 57% to 65% with an average of 61%. The energy saving variation depends on the efficiency of register ordering technique and it is proportional to the percentage of regfile accesses to the smallest partition.

Applying low power regfile design techniques such as precise read control, bypass skip, separate-R0, and modified storage cell changes the register usage distribution from the base case scenario as shown in Figure 7-8. However, the simulations show the optimal sizes of the small regfile partition are still around 8 registers. The 8 most frequently accessed registers remain as R0, R2, R3, R4, R5, R6, R16, and R29. In the cases of separate-R0 and modified-storage-cell regfiles, because R0 is removed from the regfile, R17 becomes one of the 8 most frequently referenced registers. R17 is another callee-saved register like R16 [6]. Figure 7-9, Figure 7-10, Figure 7-11, and Figure 7-12 show the energy dissipation of the base case regfile and the energy-efficient regfiles with and without the split bitline. The split bitline technique adds an additional 12% total energy saving and 50% bitline energy reduction to the precise-read-control regfile, 11% total energy saving and 47% bitline energy reduction to the bypass-skip regfile, 5% total energy saving and 21% bitline energy reduction to the separate-R0 regfile, and 4% total energy saving and 24% bitline energy reduction to the modified-storage-cell regfile. The removal of R0 from the regfiles in the separate-R0 and the modified storage cell techniques causes a more even distribution of register usage as shown in Figure 7-8. Therefore, the split bitline energy saving contributions are smaller when combining it with these two methods than with others.

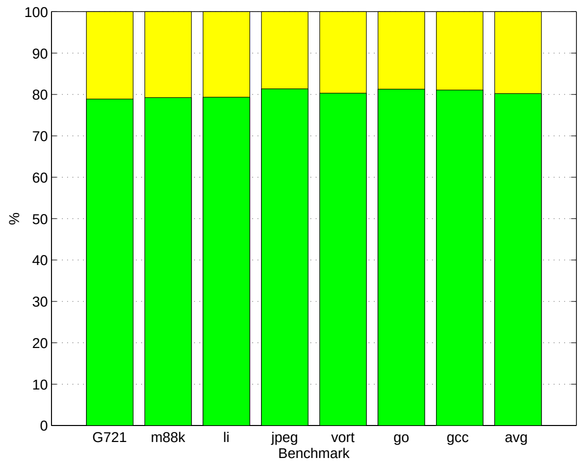

Figure 7-7: Comparative energy consumption for the base case regfile and the split-bitlines regfile.

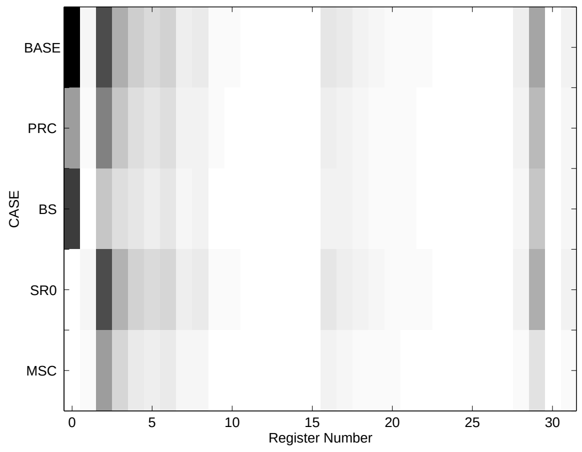

Figure 7-8: Comparative weighted average register usage density for the base case regfile (BASE), the precise-read-control regfile (PRC), the bypass-skip regfile (BS), the separate-R0 regfile (SR0), and the modified-storage-cell regfile (MSC).

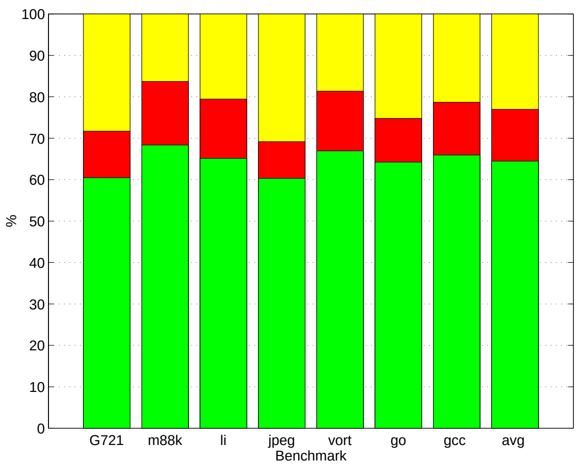

Figure 7-9: Comparative energy consumption for the base case regfile and the precise-read-control regfile with and without split bitline.

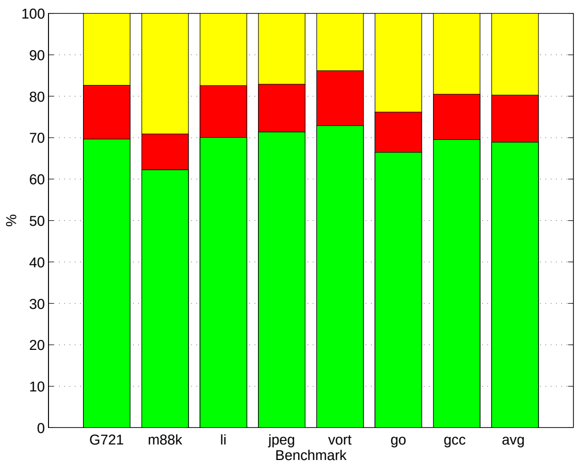

Figure 7-10: Comparative energy consumption for the base case regfile and the bypass-skip regfile with and without split bitline.

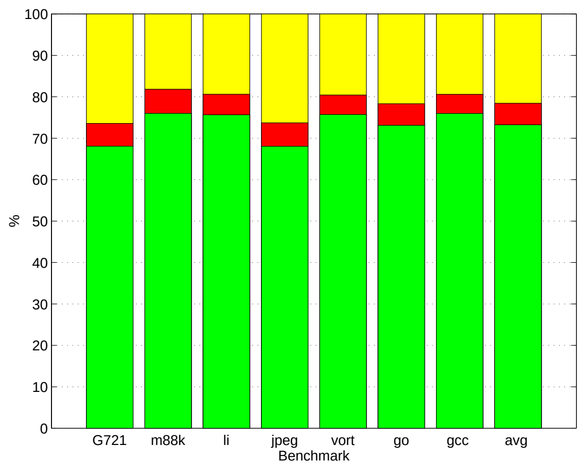

Figure 7-11: Comparative energy consumption for the base case regfile and the separate-R0 regfile with and without split bitline.

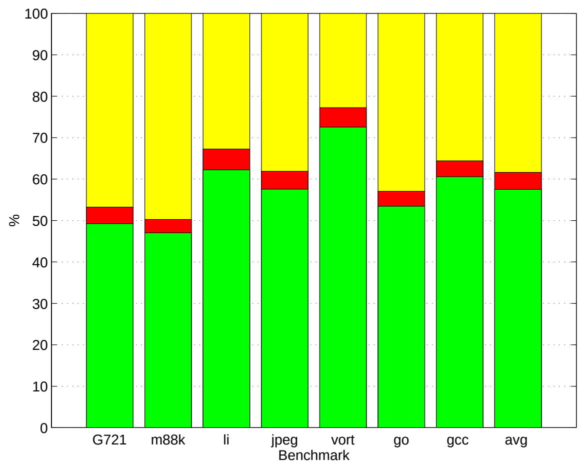

Figure 7-12: Comparative energy consumption for the base case regfile and the modified-storage-cell regfile with and without split bitline.

Chapter 8

Conclusions and Future Work

In this thesis, five different methods were proposed and evaluated in an attempt to reduce regfile energy dissipation. Since energy dissipation of CMOS circuits is proportional to the switching activity and switching capacitance, the approach here is to reduce regfile energy consumption by minimize switching activity and switching capacitance. The precise read control, the bypass skip, and the separate R0 methods successfully reduce regfiles' switching activity by minimizing the number of accesses and each has an energy saving potential of 23%, 20%, and 22%. Also, a 38% energy saving is accomplished by a slight modification of the regfile storage cells alone. The normalized weighted switching activity and energy saving of these four methods are cross-compared in Table 8.1 for each critical node in the regfile and the bypass-network. The switching activities are weighted by their switching capacitance and normalized by the base case switching activity factor. Switching index is the summation of all the normalized weighted switching activity. Energy consumption is linearly proportional to the switching index.

On the other hand, split bitline reduces regfile's switching capacitance instead of the switching activity. Split-bitline regfile reduces regfiles' bitline capacitance by having a two-size adaptive bitline length. It has an average of 61% bitline energy saving. Split-bitline regfile's total energy saving is bounded by the column circuitry and the bypass networks' energy consumption. We can obtain a greater percentage of split-bitline regfile total energy saving by having more energy-efficient column circuitry and bypass network.

These five methods have different area and latency overheads as listed in Table 8.2.

Node/Case	BASE	PRC	BS	SR0	MSC
Read-bit	43.98	33.48	29.33	30.42	12.33
Read-bitb	9.31	4.06	7.11	9.17	9.17
Write-bit,bitb	11.42	11.42	11.42	8.96	11.22
Rs-mux-input	1.30	1.05	0.95	1.00	0.55
Rs-mux-output	2.28	1.59	2.28	1.60	0.78
Rs-mux-control	0.16	0.16	0.16	0.22	0.16
Rs-latch-data	6.32	4.42	6.32	4.04	2.15
Rt-mux-input	0.61	0.50	0.56	0.62	0.61
Rt-mux-output	0.53	0.50	0.53	0.66	0.53
Rt-mux-control	0.18	0.18	0.18	0.20	0.18
Rt-latch-data	1.33	1.27	1.33	1.33	1.33
Sd-mux-input	0.45	0.34	0.40	0.46	0.45
Sd-mux-output	0.55	0.26	0.55	0.60	0.55
Sd-mux-control	0.10	0.10	0.10	0.19	0.10
Sd-latch-data	1.51	0.73	1.51	1.51	1.51
Precharge	9.12	5.97	6.80	6.60	9.21
Clk	10.76	10.76	10.76	10.76	10.76
Switching index	100.00	76.80	80.29	78.34	61.58
Energy saving	0.00	23.20	19.71	21.66	38.42

Table 8.1: Regfile switching activity percentage and energy saving evaluation for the base case regfile (BASE), the precise-read-control regfile (PRC), the bypass-skip regfile (BS), the separate-R0 regfile (SR0), and the modified-storage-cell regfile (MSC).

Method	Area Overhead	Latency Overhead
Precise read control	AND-gate	Decoder latency - half of the cycle time
Bypass skip	AND-gate	RAW hazard-detector latency - half of cycle time
Separate R0	Muxes	
Modified storage cell	Undetermined	
Split bitline	Transmission gate	Transmission gate

Table 8.2: Overhead analysis for each low energy regfile technique.

Method	Dependency
Precise read control	Number of over-fetched operands
Bypass skip	Number of bypassed operands
Separate R0	Number of R0 references
Modified storage cell	Percentage of zero bits fetched
Split bitline	Percentage of reference are most popular registers

Table 8.3: Dependency analysis for each low energy regfile technique.

Moreover, the energy saving percentage of these methods also depends on different benchmarks behaviors as shown in Table 8.3. One can achieve higher energy saving percentage by combining multiple methods together. However, the energy savings are not 100% additive among them as discussed in earlier chapters. Different combinations are tested and the results are shown in Figure 8-1. The combination of modified storage cell, precise read control, bypass skip, separate R0, and split bitline methods has the largest energy saving of 63%.

8.1 Summary of Contributions

In conclusion, this research shows that 58% of fetched operands are discarded, 39% of regfile writes are redundant, 30% of regfile references are R0, 82% of fetched bit values are zero, and 83% of regfiles references are accounted for by the 8 most popular registers. Therefore, this thesis investigates five regfile energy saving methods–precise read control,

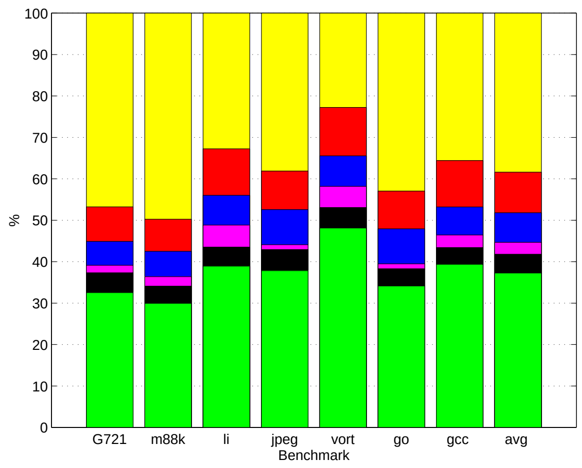

Figure 8-1: Comparative energy consumption for the base case regfile with each additional low energy regfile technique in the following order: modified storage cell, precise read control, bypass skip, separate R0, and split bitline.

bypass skip, separate R0, modified storage cell, and split bitline. Each has an energy saving of 23%, 20%, 22%, 38%, and 20% respectively. A total energy saving of 63% can be achieved by combining all of these five methods without any software changes.

8.2 Discussion

In this thesis, we reduced regfile energy consumption by decreasing the switching activity and the switching capacitance. Another approach to reduce regfile energy consumption is by using differential sense amplifiers in the column circuitry to avoid rail-to-rail read bitline swings [13]. The energy dissipation for each limited bitline swing can be calculated as

$$\frac{1}{2} \cdot C_{switch} \cdot (s \cdot V_{dd}) \cdot V_{dd}$$

Where C_{switch} is the switching capacitance, s is the voltage scaling coefficient, and V_{dd} is the supply voltage. The bitline swings $s \cdot V_{dd}$ for each low-to-high and high-to-low transition. Table 8.4 shows the required voltage scaling coefficient and the maximum bitline swing voltage for double-ended read regfiles to obtain the same read bitline energy consumption as each single-ended read regfile case discussed in this thesis. For example, s

Single-Ended Read w/ ERT	Equivalent Double-Ended Read w/o ERT	
Regfile Case ($s = 1$, $sV_{dd} = 3$V)	s	sV_{dd} (V)
Base	0.47	1.53
Precise read control	0.33	1.09
Bypass skip	0.32	1.05
Separate R0	0.34	1.13
Modified storage cell	0.18	0.61
Split bitline	0.33	1.08

Table 8.4: The required voltage scaling coefficient and the maximum bitline swing voltage for obtaining the same read bitline energy consumption. ERT stands for energy reduction technique.

has to to less than 0.47 to get energy saving over the single-ended read base case.

We can use differential sense amplifiers with four of our five regfile energy reduction techniques to further decrease the energy dissipation. Modified storage cell is the only technique which doesn't affect regfile energy dissipation when differential sense amplifiers are used. Because the differential-sense-amplifier regfile has double-ended sensing with a pair of bitlines for each read port, one of the paired bitlines always swings regardless of the value read.

Figure 8-2 shows the energy saving from using differential sense amplifiers with and without the four regfile energy reduction techniques–precise read control, bypass skip, separate R0, and split bitline. The energy saving ranges from 2% to 50% as the bitline swing voltage ranges from 1.5V to 100mV. The four methods further reduce the remaining regfile energy dissipation by almost one half; the maximum energy saving of 71% can be achieved with the combination of the energy reduction techniques and a 100mV bitline swing. Differential sense amplifiers reduce both the power dissipation and read sensing delay but increase the area overhead because of the extra read bitlines required for double-ended sensing.

Figure 8-2 also shows that for differential-sense-amplifier regfiles to have energy saving over the optimal single-ended read regfiles, the read bitline voltage swing has to be less than 0.9V. The optimal single-ended read regfile has a constant 63% of energy saving while the double-ended read regfile has a maximum of 71% energy saving. To achieve the

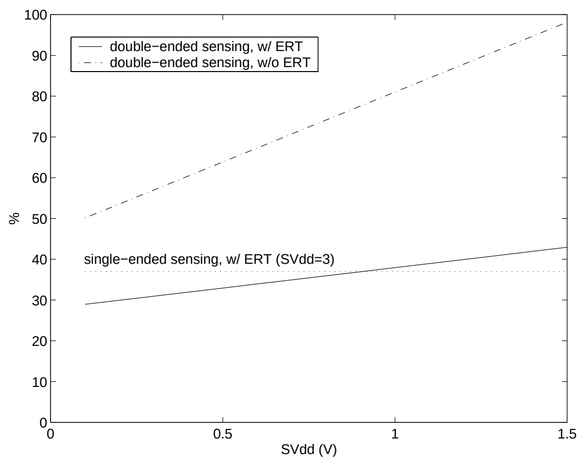

Figure 8-2: Comparative energy consumption for the rail-to-rail bitline swing regfile and the limited bitline swing regfile with and without the energy reduction techniques (ERT).

additional 8% energy reduction, the read bitline is only allowed to swing 100mV, which is difficult to accomplish due to CMOS technology limitations. The small additional saving of using differential sense amplifiers is because it must always swing one read bitline and because of the constant bypass network energy. Moreover, the supply voltages of modern microprocessors are dropping faster than the CMOS threshold voltage, which leads to less room for decreasing bitline swings. Therefore, using differential sense amplifiers does not help us much in energy saving after applying our five energy reduction techniques. It might also be possible to limit the bitline swing of our single-ended read design, further reducing the potential advantages of differential reads.

8.3 Future Work

Designing an energy efficient regfile becomes more critical as we move toward high performance multiple-issue microprocessors. Microprocessor designs are moving towards wider instruction issue and increasingly complex out-of-order execution [19]. For example, the Alpha 21264 microprocessor can fetch and execute up to four instructions per cycle [9]. Its out-of-order execution leads to register renaming and increased regfile size, with 80 physical registers. The average energy dissipation per regfile access increases with the size of regfile. Also, multiple-issue high performance microprocessors require regfiles with many read and write ports. The silicon area grows quadratically in the number of ports [16]. Therefore, one can expect that these microprocessors' regfiles consume a higher percentage of total energy than in single-issue microprocessors. The potential energy saving of the five regfile energy reduction techniques proposed for the single-issue microprocessors in this thesis should be evaluated for multiple-issue high performance microprocessors.

Bibliography

[1] K. Asanović. *Vector Microprocessors*. PhD thesis, University of California at Berkeley, May 1998.

[2] T. D. Burd and B. Peters. Power analysis of a microprocessor: A study of an implementation of the MIPS R3000. Technical report, ERL Technical Report, University of California, Berkeley, May 1994.

[3] R. Y. Chen, R. M. Owens, M. J. Irwin, and R. S. Bajwa. Validation of an architectural level power analysis technique. In *DAC '98. Proceedings of the 35th Annual Design Automation Conference*, San Francisco, CA, June 1998.

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

[5] J. L. Hennessy and D. A. Patterson. *Computer Architecture — A Quantitative Approach, Second Edition*. Morgan Kaufmann, 1996.

[6] L. Huffman and D. Graves. MIPSpro Assembly Language Programmer's Guide. Technical Report 007-2418-002, Technical Report, Silicon Graphics, 1996.

[7] 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.

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

[9] R. E. Kessler. The Alpha 21264 microprocessor. *IEEE Micro*, 19(2):24–36, March/April 1999.

[10] P. Landman. High-level power estimation. In *Proceedings ISLPED*, pages 29–35, Monterey, CA, 1996.

[11] L. Nagel. SPICE2. Technical Report ERL-M520, ERL Technical Memo, University of California, Berkeley, 1975.

[12] 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.

[13] J. Rabaey. *Digital Integrated Circuites*. Prentice Hall, 1996.

[14] J. Scott. Designing the low-power M*CORE architecture. In *Power Driven Microarchitecture Workshop at ISCA98*, Barcelona, Spain, June 1998.

[15] V. Stojanovic and V. G. Oklobdzija. 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.

[16] M. Tremblay, B. Joy, and K. Shin. A three dimensional register file for superscalar processors. In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, pages 191–201, January 1995.

[17] 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.

[18] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design, Second Edition*. Addison Wesley, 1993.

[19] 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.