Journal of Advances in Science and Technology

Vol. III, No. VI, August-2012, ISSN 2230-9659

FILE DESIGN A LOW POWER CONSUMING REGISTER: DYNAMIC PROFILING OF MEASUREMENTS

www.ignited.in

Journal of Advances in Science and Technology
Vol. III, No. VI, August-2012, ISSN 2230-9659

File Design a Low Power Consuming Register: Dynamic Profiling Of Measurements

Rashi Malhotra

Research Scholar, Singhania University, Rajasthan

Abstract: ***Register file access in RISC processors is highly asymmetric in nature; a relatively small number of registers account for a majority of the register file accesses during program execution. This is mostly because the lifetime of typical program variables is very small and a small number of registers are heavily reused. Also, because of calling conventions followed in most RISC processors; these are rules enforced by the software during procedure call on we propose a strategy to determine an application-specific register file banking structure and an appropriate register mapping strategy that minimizes power dissipation in the register file. A straight forward method to identify the best bank structure is to exhaustively simulate all possible configurations of banking and choose the configuration yielding maximum power reduction. But this approach is infeasible if the number of registers is large; moreover, there are an exponentially large number of different mappings of logical to physical registers (register allocation decisions).***

----------♦----------

MOTHODOLOGY

We propose a profile-based method to estimate the optimal bank configuration for a given application by analyzing access frequency of access to each register. A register renaming technique is proposed to generate efficient code for the best bank configuration, which maximizes power savings on the selected register file bank configuration. To obtain measurement traces, a cycle-accurate simulator is used to simulate the micro architecture-level behavior of a MIPS RISC scalar microprocessor with a five-stage pipeline as shown in Figure 1-1. The MIPS processor executes the MIPS-II ISA. This simulator only traces user level instructions and records register data 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 measurements and Power stone measurements [14] are used as a workload. Table 1.1 lists the measurements and input data sets used in this research, and shows total instruction and cycle counts. Each measurement was run to completion. This study only covers predominantly integer measurements and Table 1.2 shows the distribution of instruction types in each measurement. Each measurement is compiled with gcc version 2.7.0 with -O3 optimization and linked with the new lib standard C library.

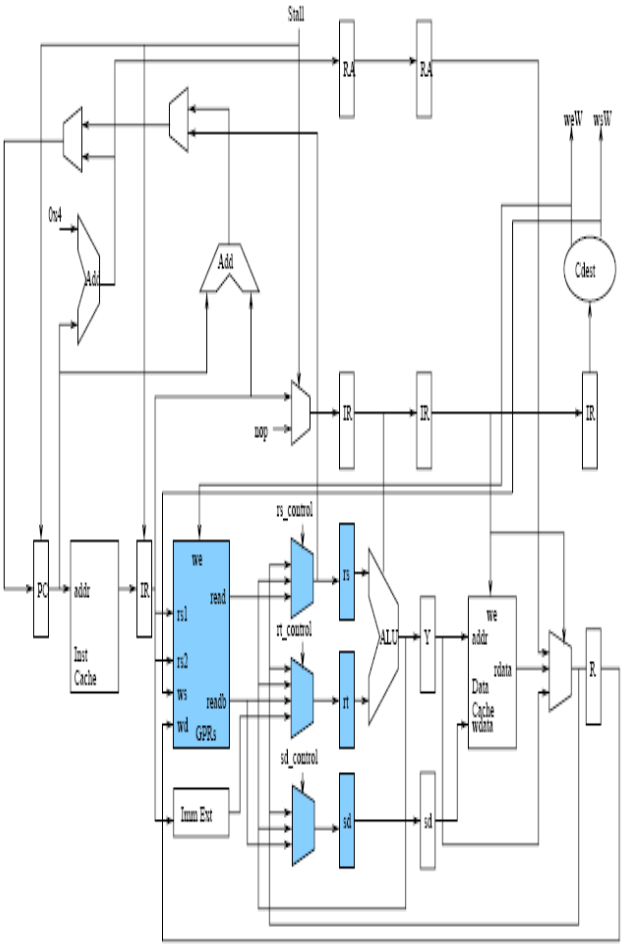


Figure 2-1: Five-stage pipeline data path.

1.2 POWER ESTIMATION MODEL

Initially the demand for reducing power in digital systems was limited to products which are required to operate under conditions where battery life is an issue, while power was not a primary concern in the high performance processor market. In recent years, however, reducing power dissipation has also become a critical design goal for high-performance microprocessors, because the limited power-dissipation capabilities of packages using inexpensive air-cooling techniques have begun to constrain further performance growth.

To date most of the power reduction in high-performance processors was achieved through supply voltage reduction and process shrinks. Unfortunately, there is a limit to how far supply voltages may be reduced without increasing circuit delays, and the power dissipated on chip is increasing each year even as process technology improves.

Meanwhile, the power-dissipation capabilities of inexpensive air-cooled packages are limited. Hence other solutions to the power growth problem must be found. It is the goal of this thesis to explore architectural level solutions to the power problems in high-performance superscalar microprocessors. We attempt to bring the power issue to the earliest phases of high-performance microprocessor development, where there is still a significant potential for power reduction. Register file access in RISC processors is highly asymmetric in nature; a relatively small number of registers account for a majority of the register file accesses during program execution. This is mostly because the lifetime of typical program variables is very small and a small number of registers are heavily reused. Also, because of calling conventions followed in most RISC processors; these are rules enforced by the software during procedure call on we propose a strategy to determine an application-specific register file banking structure and an appropriate register mapping strategy that minimizes power dissipation in the register file. A straight forward method to identify the best bank structure is to exhaustively simulate all possible configurations of banking and choose the configuration yielding maximum power reduction. But this approach is infeasible if the number of registers is large; moreover, there are an exponentially large number of different mappings of logical to physical registers (register allocation decisions).

We propose a profile-based method to estimate the optimal bank configuration for a given application by analyzing access frequency of access to each register. A register renaming technique is proposed to generate efficient code for the best bank configuration, which maximizes power savings on the selected register file bank configuration.

Measurement {Data Set}	Instruction Count (Millions)	Cycle Count (Millions)	Description
M88ksim {test}	519	567	A chip simulator for Motorola 88100 microprocessor
1i (test)	997	1,129	Xlisp interpreter
Go {train}	579	631	An internationally ranked go playing program
Gcc {fref:2c-decl-s}	1,396	1,524	Based on theGNU C compiler version 2.5.3
vortex ftestg	10,054	11,123	An object oriented database
Jpeg {test:specmum.ppm}g	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 1.1: Measurement 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 1.2: Measurement 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}h is the switching capacitance and V_{dd} d is the supply voltage. Therefore, the average power 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 r is the average data transition frequency of the node r within the functional block as determined by the dynamic measurement profiling. The C_{switch_r} is the switching capacitance related to node r.

The parameters used in this power estimation model are based on a 0.6 n well CMOS process technology with 3.3 V power supply and two layers of metal. The design of register data 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 net list 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 power estimation model. The register data and the bypass network designs used in this power model are described in the following subsections. A base-case scenario power estimation analysis is illustrated in the summary subsection. CPU design strategy based on the insight that simplified (as opposed to complex) instructions can provide higher performance if this simplicity enables much faster execution of each instruction. A computer based on this strategy is a reduced instruction set computer (also RISC). There are many proposals for precise definitions,[1] but the term is slowly being replaced by the more descriptive load-store architecture. Well known RISC families include DEC Alpha, AMD 29k, ARC, ARM, Atmel AVR, Blackfin, MIPS, PA-RISC, Power (including PowerPC), SuperH, and SPARC.

Some aspects attributed to the first RISC-labeled designs around 1975 include the observations that the memory-restricted compilers of the time were often unable to take advantage of features intended to facilitate manual assembly coding, and that complex addressing modes take many cycles to perform due to the required additional memory accesses. It was argued that such functions would be better performed by sequences of simpler instructions if this could yield implementations small enough to leave room for many registers,[2] reducing the number of slow memory accesses. In these simple designs, most instructions are of uniform length and similar structure, arithmetic operations are restricted to CPU registers and only separate load and store instructions access memory. These properties enable a better balancing of pipeline stages than before, making RISC pipelines significantly more efficient and allowing higher clock frequencies

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 bit lines are pre-charged high and the write bit lines are driven. Registers are written during the first half of the cycle while the read data is 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 1-1.

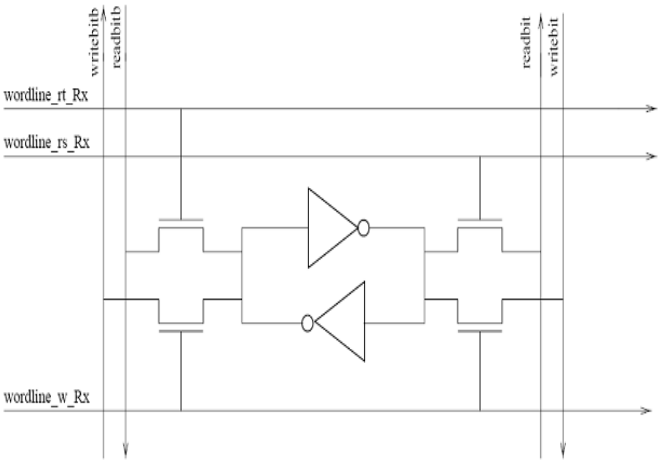


Figure 1-2: Register data storage cell.

It is observed that the power dissipation of the read and write bit lines dominates the regfile power consumption. Therefore, the power estimation model for the regfile is based on the transition activity of read bitlines and write bit lines. The address decoding of regfile is not included in this power estimation model. The regfile consists of a 32x32 matrix of storage cells, Figure 1-2, for the 32 32-bit-wide registers with a column circuitry module, Figure 1-3, at the end of each bit line. 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 bit line pre-charges, write drive, and data buffering. The switching capacitance of the read bit lines, write bit lines, and pre-charging transistors of the regfile for one of the 32 bit slices is shown in Table 2.3. In the early days of the computer industry, programming was done in assembly language or machine code, which encouraged powerful and easy-to-use instructions. CPU designers therefore tried to make instructions that would do as much work as feasible. With the advent of higher level languages, computer architects also started to create dedicated instructions to directly implement certain central

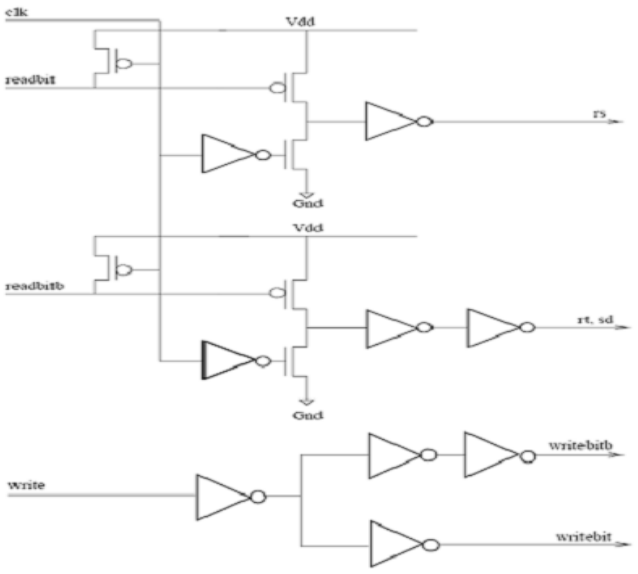
mechanisms of such languages. Another general goal was to provide every possible addressing mode for every instruction, known as orthogonality, to ease compiler implementation. Arithmetic operations could therefore often have results as well as operands directly in memory (in addition to register or immediate).

The attitude at the time was that hardware design was more mature than compiler design so this was in itself also a reason to implement parts of the functionality in hardware or microcode rather than in a memory constrained compiler (or its generated code) alone. This design philosophy became retroactively termed complex instruction set computing (CISC) after the RISC philosophy came onto the scene.

CPUs also had relatively few registers, for several reasons:

- More registers also implies more time-consuming saving and restoring of register contents on the machine stack.
- A large number of registers requires a large number of instruction bits as register specifiers, meaning less dense code (see below).
- CPU registers are more expensive than external memory locations; large register sets were cumbersome with limited circuit boards or chip integration.

An important force encouraging complexity was very limited main memories (on the order of kilobytes). It was therefore advantageous for the density of information held in computer programs to be high, leading to features such as highly encoded, variable length instructions, doing data loading as well as calculation (as mentioned above). These issues were of higher priority than the ease of decoding such instructions.



An equally important reason was that main memories were quite slow (a common type was ferrite core memory); by using dense information packing, one could reduce the frequency with which the CPU had to access this slow resource. Modern computers face similar limiting factors: main memories are slow compared to the CPU and the fast cache memories employed to overcome this are limited in size. This may partly explain why highly encoded instruction sets have proven to be as useful as RISC designs in modern computers. 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 1-4 shows the design for a three-input mux. The latches in this bypass network are similar to the IBM PowerPC603MS latch designs [15], Figure 1-5. The bit-slice switching capacitance of mux input, mux output, mux control line, latch data value, latch clock input is listed in Table 1.4.

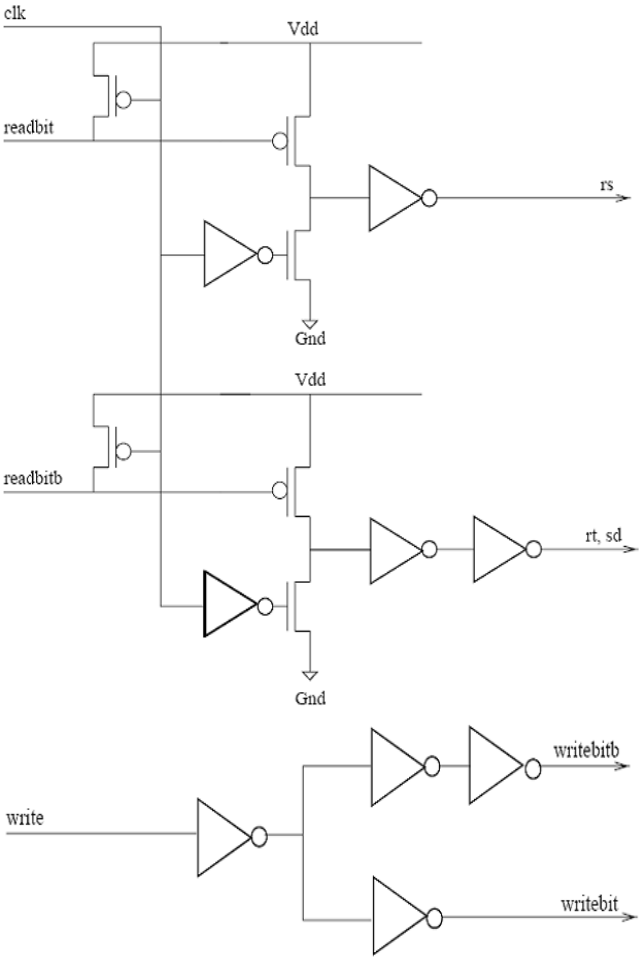


Figure 1-3: Column circuitry for one bit slice.

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

Table 1.3: Register data switching capacitance.

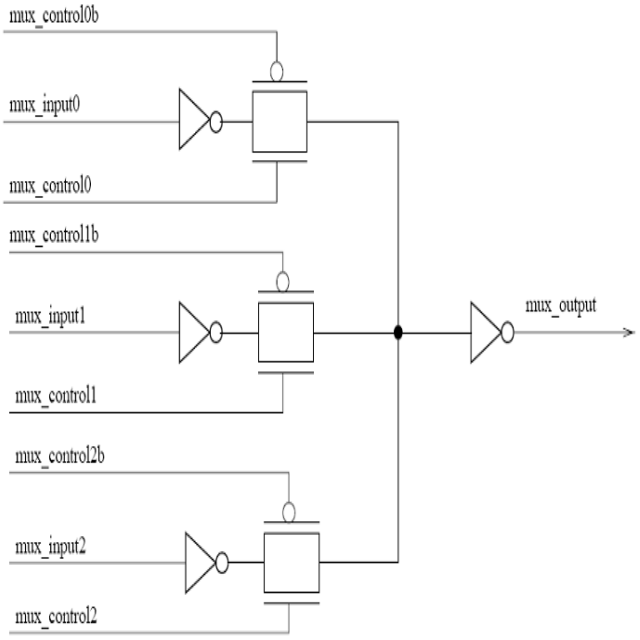
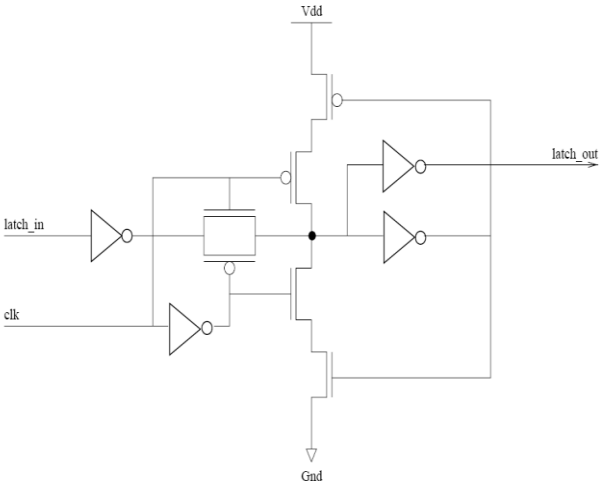


Figure 1-4: Three-input transmission gate mux.

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 1.4: Bypass network switching capacitance.



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 1.5: The average bit slice power consumption analysis for g721 base case.

The heterogeneous banking of register file with n registers used in our study. It consists of two banks, one bank with a small number of registers (0 to k) and a second bank with a larger number of registers (k+1 to n-1), with k much smaller than n/2. This non-hierarchical register file banking is different from conventional register file banking where register file is divided into a number of banks of equal size. From the compiler's perspective, the structure remains a single register file. If we can ensure that the placement of registers is such that most of the accesses occur to the smaller bank, there will a significant reduction in the overall power dissipation

as the smaller bank has a relatively small bit-line switching capacitance. Heterogeneous register file banks differ in number of registers in each bank, hence making the address decoder and output selection logic is more complex, which introduces overheads in area, delay, and power. For most processor/cache configurations we studied, the memory access pipeline stage effectively determined the overall clock cycle time and there was a reasonable amount of slack available for utilization in register file access, so the overall performance was unaffected. The additional circuit does lead to power overheads, so the structure is worth it if this is offset by the savings due to the optimization. Our experiments indicate that this is indeed the case. The area overhead is real however, and needs to be traded off against the power savings. To calculate the total power per cycle, the power estimation model sums up all the power dissipation per cycle due to data value transitions. Table 1.5 is an example to show how the power model calculates the total power per cycle for the g721 measurement 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 power consumption per cycle is 5.9 pJ and the total power consumption per cycle for the 32-bit wide data path is 190 pJ.

REFERENCES:-

1. K. Asanovi´c. 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 Micro architecture 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 data 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 data architectures for inherently low power microprocessors. In Power Driven Micro architecture Workshop at ISCA98, Barcelona, Spain, June 1998.