Energy-Efficient Register Access

Jessica H. Tseng and Krste Asanović

MIT Laboratory for Computer Science, Cambridge, MA 02139, USA

SBCCI2000

1

Introduction

- **Motivation**
  - Hand-held portable multimedia wireless device. (ex: Palm-pilots and wireless-phone)
  - Register files represent a substantial portion of energy budget in modern microprocessor. (ex: Motorola's M.CORE--16% of total processor power, 42% of datapath power)
- **Objective**
  - Software invisible techniques to reduce switching activity frequency and switching capacitance of a register file.

2

Overview

- Methodology / Background.
- 7 Energy Saving Techniques.
- Combining Techniques.
- Conclusion.

3

Methodology

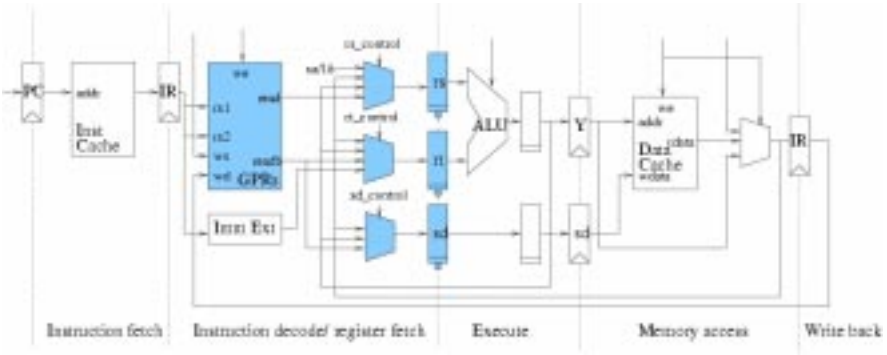
- Use dynamic benchmark traces of SPECint95 and g721 (total of 14 billion instructions) to evaluate modification to a conventional register file for energy efficiency.
- Custom layout the register file and bypass network in Magic layout program—0.25µm TSMC CMOS process.
- Use SPACE 2D extractor to extract layout parasitic for circuit simulation.
- Use HSpice to simulate the extracted netlist and to determine the effective switching capacitance.
- Use Matlab to build energy evaluation model.

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

4

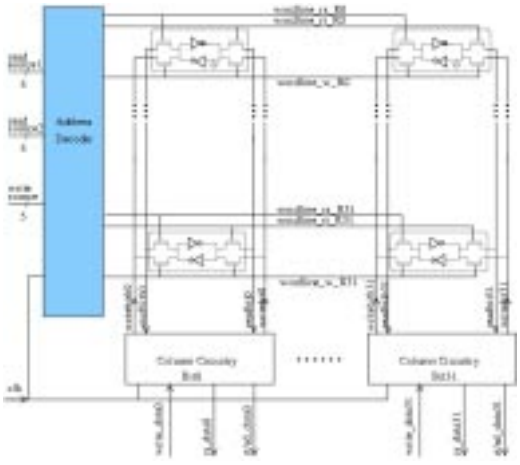
Single Issue MIPS-II Compatible RISC Microprocessor

- For low-power embedded application



5

Register File



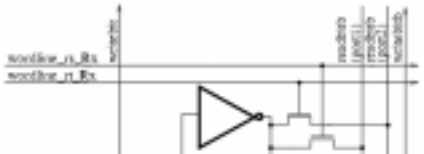
- High performance dynamic design. (target for nominal processor clock rate of 400 MHz).

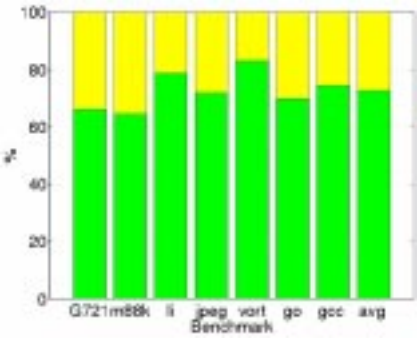
- Static address decoder.
- Three ports integer register file. (two singled-ended read ports and one differential write port)

6

Modified Storage Cell

- On average, 82% of the bits fetched from the register file are zeros.
- 17% storage cell area increase (9% overall regfile area increase) and no read delay penalty.
- 27% energy saving.

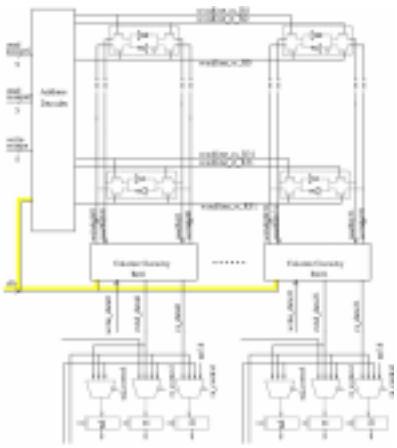




7

Precise Read Control

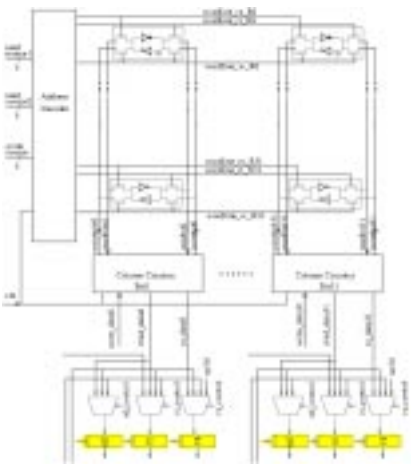
- On average, each instruction only requires 1.3 operands.
- Gating the read word line enable pulse prevents discharge of bitlines.
- No area overhead and no access time penalty, assuming the decoding of required operands completes in the first half of the cycle.



- Saving ranges from 15% to 27% with an average of 21%.

8

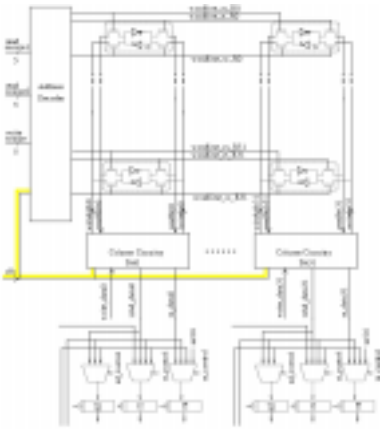
Latch Clock Gating



- Only 81% of instructions use values held in the ***rs*** or ***rt*** latches. 10% of instructions use values held in ***sd*** latches.
- Not clocking latches whose values are not needed.
- No area overhead and no read delay penalty.
- 8% energy saving consistently.

9

Bypass Skip

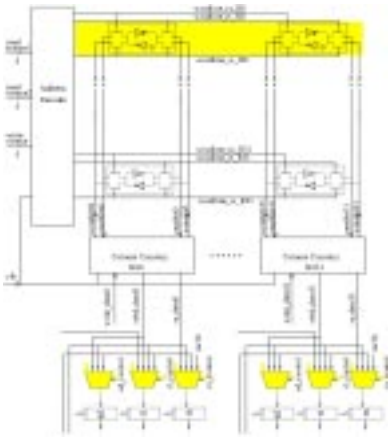


- 36% of all necessary operands are supplied by the bypass network.
- Similar implementation to precise-read control.
- No area overhead and no increase in latency if we can finish determining the bypass control in the first half of the cycle.
- Saving ranges from 11% to 23% with an average of 16%.

10

Bypass Register 0

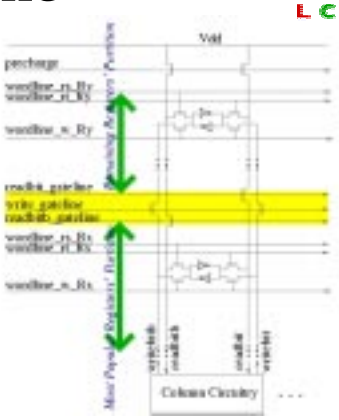
- 40% of regfile write accesses and 25% of regfile read accesses are Register 0.
- Remove the zero cells from the register file and provide zero input to the bypass mux.
- Avoid bitline swings on reading R0 and writing R0.
- 3% total area increase and no access time penalty.
- Saving ranges from 7% to 17% with an average of 14%.



11

Split Bitline

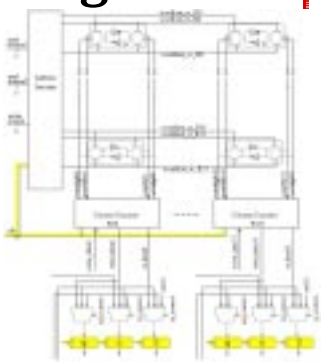
- The 8 most popular register account for 83% of all regfile accesses. Particular registers are always accessed more frequently than others. (MIPS assembler conventions)
- Use n-type transistor to separate the two partitions.



- 2% total area increase and 3% delay penalty to access the least-frequently-used registers.
- Saving ranges from 11% to 13% with an average of 12%.

12

Read Caching

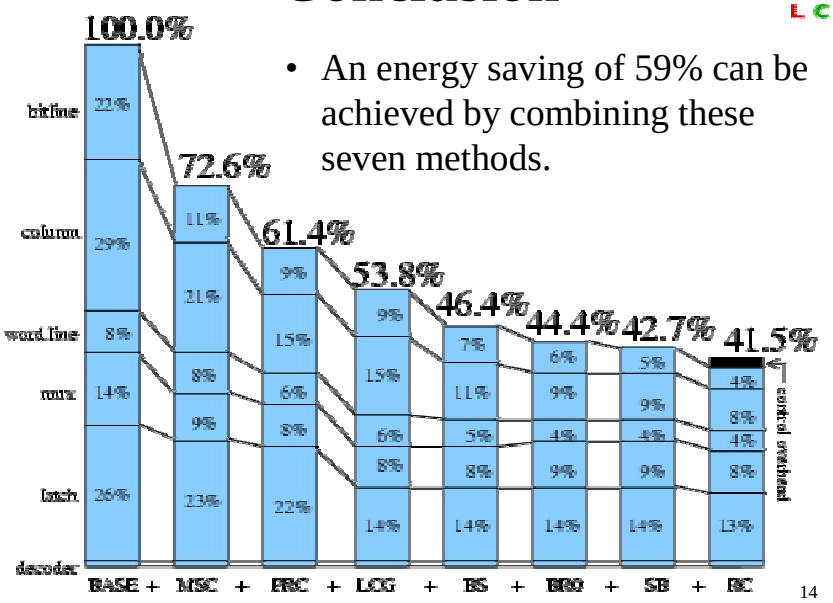


- In some cases, two successive instructions read the same register from the register file.
  - *add r4, r1, r6*
  - *xor r9, r1, r2*
- Not clocking the latch and not reading the register file for the second instruction.
- 9% of accesses to the *rs* latch can be supplied via read cashing and only 2% for *rt* and *sd* latches.
- 1% total area increase with no access time penalty.
- Only 1% energy saving due to the high control logic overhead.

13

Conclusion

- An energy saving of 59% can be achieved by combining these seven methods.



14