



# The Emerging Power Crisis in Embedded Processors What Can a (Poor) Compiler Do ?

**Weng-Fai Wong**

**National University of Singapore**

# Collaborators

- L.N. Chakrapani
  - College of Computing, Georgia Institute of Technology
- P. Korkmaz, V.J. Mooney III, K.V. Palem, K. Puttaswamy
  - School of Electrical and Computer Engineering, Georgia Institute of Technology
- Funded by DARPA PAC/C Program

# Introduction

- Energy and power consumption is an important barrier towards widespread deployment of embedded systems
  - Computing element accounts for a high percentage of power

- This problem can be tackled at several levels
  - Low power VLSI devices and logic
  - Novel micro architectural features like voltage scaling
  - Operating system innovations like scheduling
  - Compiler optimizations for power

# Problem Statement

- What phenomena in the interactions of the compiler, the application and the processor micro architecture gives rise to energy savings ?
- Classify compiler optimizations into broad categories based on how the achieve power and energy savings
- Serves as a roadmap for compiler designers wishing to tackle the issue of power and energy consumption

# Organization

- Description of the experiment infrastructure
- Experiments that address different aspects of compiler optimizations and micro architectural features that consume power
- Taxonomy of compiler optimizations of power
- Recommendations and insights
- Conclusion and future work

# Experiment Infrastructure

- Previous work in the area
  - Actual measurement of power
  - Mathematical and analytical models for power consumption
  - Architectural simulation
- Optimizing compiler infrastructure
  - Compiles code targeting the StrongARM processor
- Verilog model of a RISC processor
  - Executes the code generated by the compiler
  - Tools to measure various parameters like power consumption
- Skiff board with StrongARM processor
  - Devices to measure system level power

# Trimaran Compiler Infrastructure

- Integrated compilation and performance monitoring infrastructure
- Target is characterized by HPL-PD
  - Parameterized processor architecture
  - Supports predication, control and data speculation, compiler controlled management of memory hierarchy
- Has “Triceps” backend to generate ARM assembly
  - Generated code can run on Verilog model as well as the Skiff board
- Open source, can be easily modified
  - http://www.trimaran.org

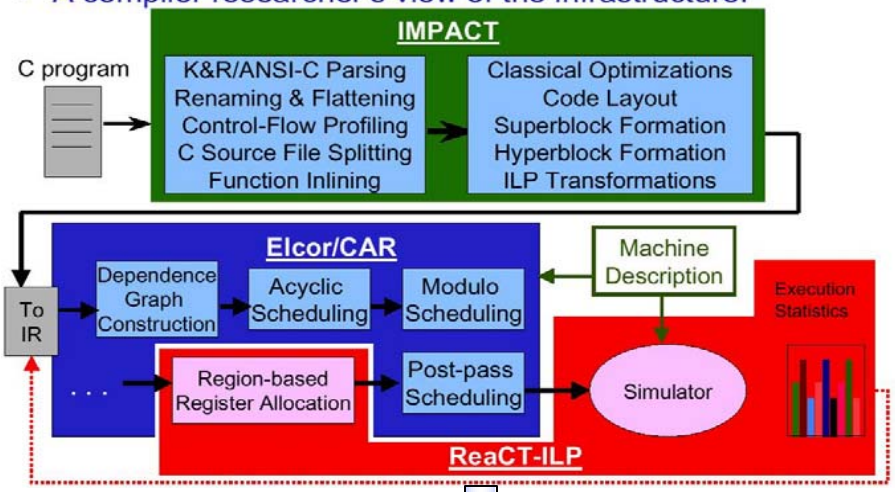
# Verilog Model

- Verilog model of an ARM like RISC processor
  - Developed by the university of Michigan
- Synthesized with the Synopsys design compiler
  - Targets 0.25 micron TSMC library
- Synopsys power compiler used for power estimation
  - Has simulation environment that runs the programs and collects switching activity
  - Has synthesis environment that provides measure of static and dynamic power

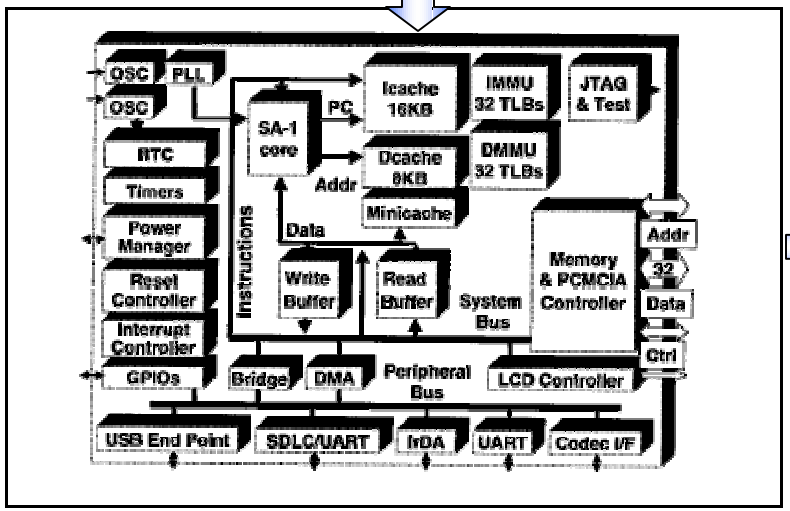
# Experiment Infrastructure

**Trimaran**

- A compiler researcher's view of the infrastructure:

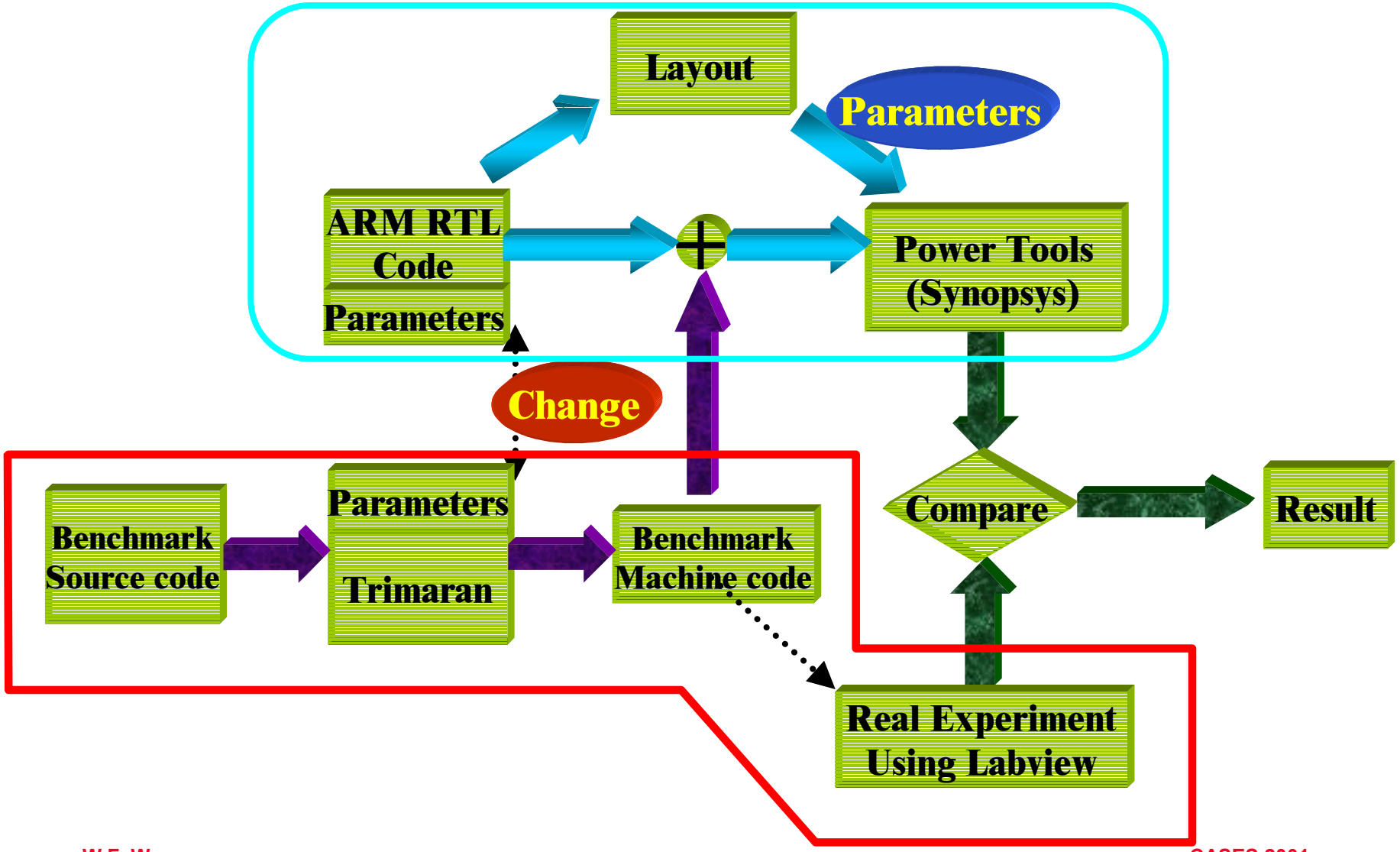



**Verilog ARM Model**

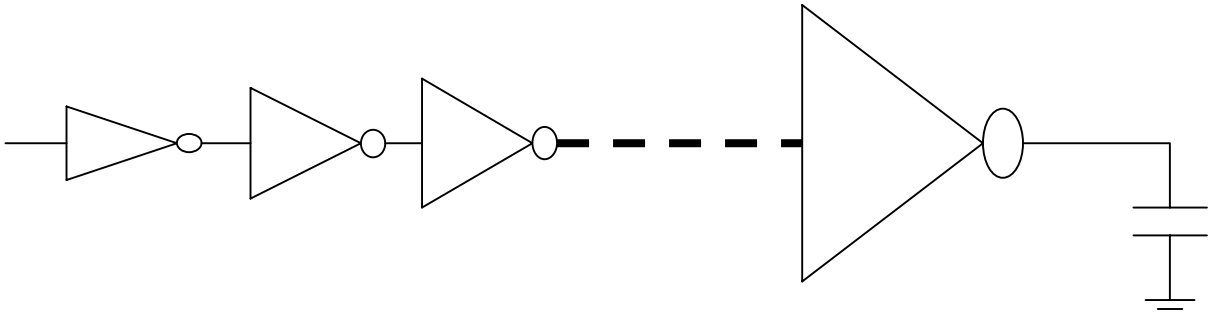



**Power and Energy Consumption**

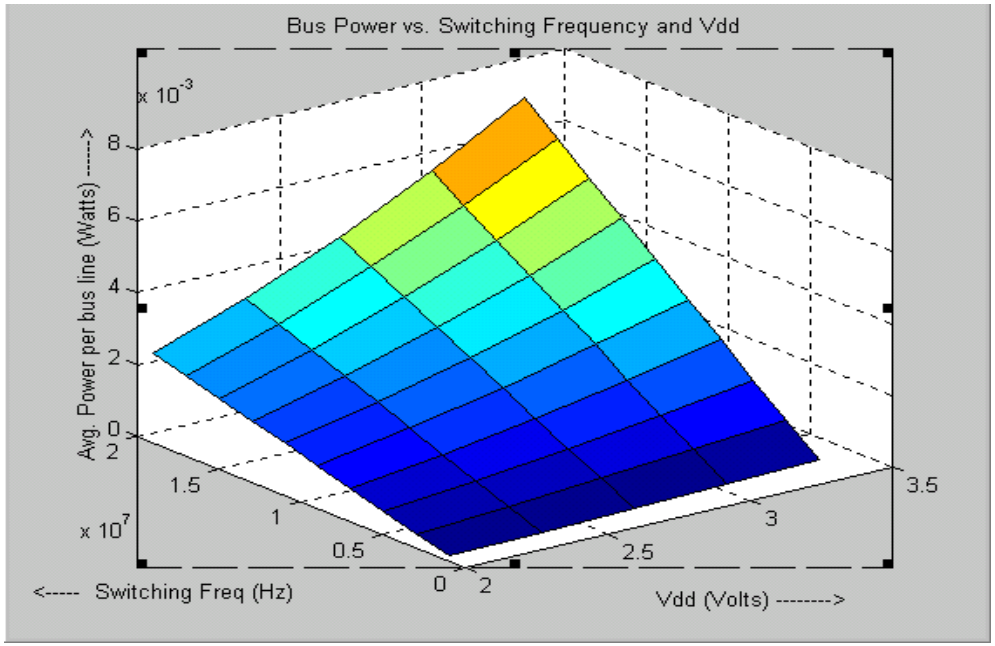
# Power Measurements: Both Simulation and Empirical



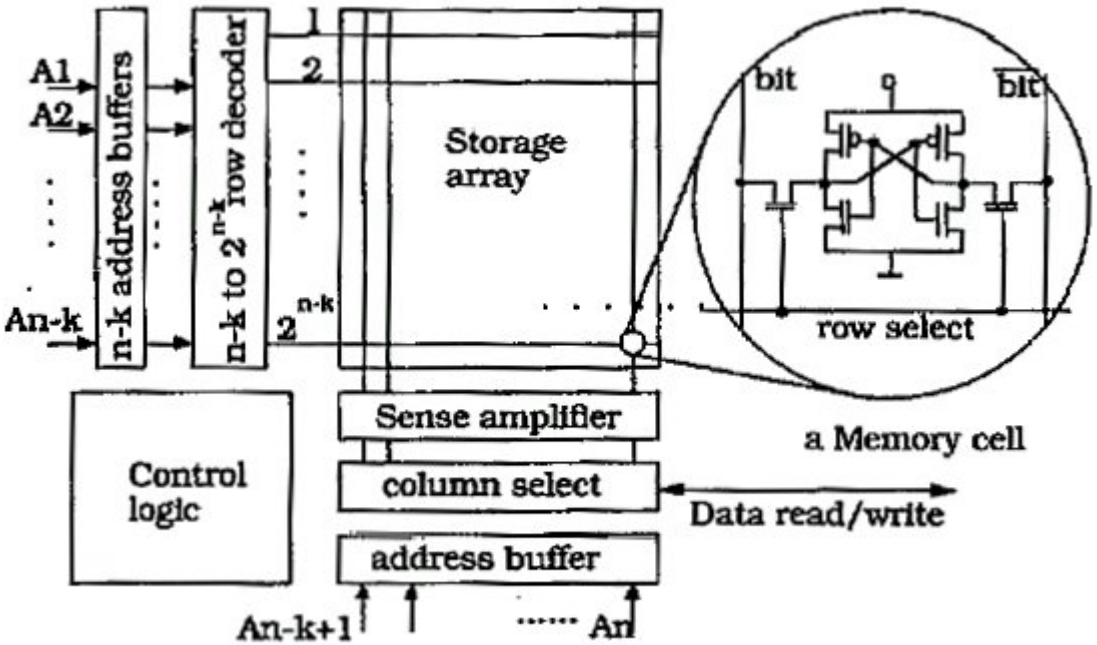
# Bus Model



- Bus Drivers modeled as a series of inverters



# Memory Model




$$P_{\text{memcell}} = \frac{2^k}{2}\left(c_{\text{int}} l_{\text{column}} + 2^{n-k} C_{\text{tr}}\right) V_{\text{dd}} V_{\text{swing}}$$

$$P_{\text{row\_decoding}} = \frac{1+0.3}{2}\frac{2(n-k)}{2} \times \left(\frac{2^{n-k}}{2} C_{\text{tr}} + c_{\text{int}} l_{\text{column}}\right) V_{\text{dd}}^2$$

$$P_{\text{row\_driving}} = \frac{1}{2}\left\{2^k (2C_{\text{tr}}) + 2(n-k)C_{\text{tr}} + c_{\text{int}}[8(n-k)W_{\text{int}} + l_{\text{row}}]\right\} V_{\text{dd}}^2$$

$$P_{\text{column\_select}} = \frac{1.3}{2}\left(\sum_{i=1}^{k-1} 2^{k-i} C_{\text{tr}} + k c_{\text{int}} l_{\text{column}}\right) V_{dd}^2$$

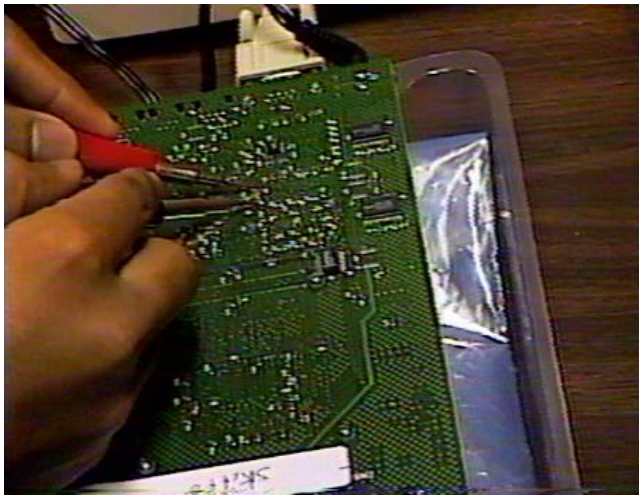
$$P_{\text{sensamp.+load}} = \frac{V_{\text{dd}} I_{\text{sens}}}{f_{\text{mem\_clock}}} + \frac{1.25}{2} c_{\text{int}} \times \left(8kW_{\text{int}} + \frac{l_{\text{row}}}{2}\right) V_{\text{dd}}^2$$

**Total Power = $P_{\text{memcell}} + P_{\text{row\_decoding}} + P_{\text{row\_driving}} + P_{\text{column\_select}} + P_{\text{sensamp.load}}$**

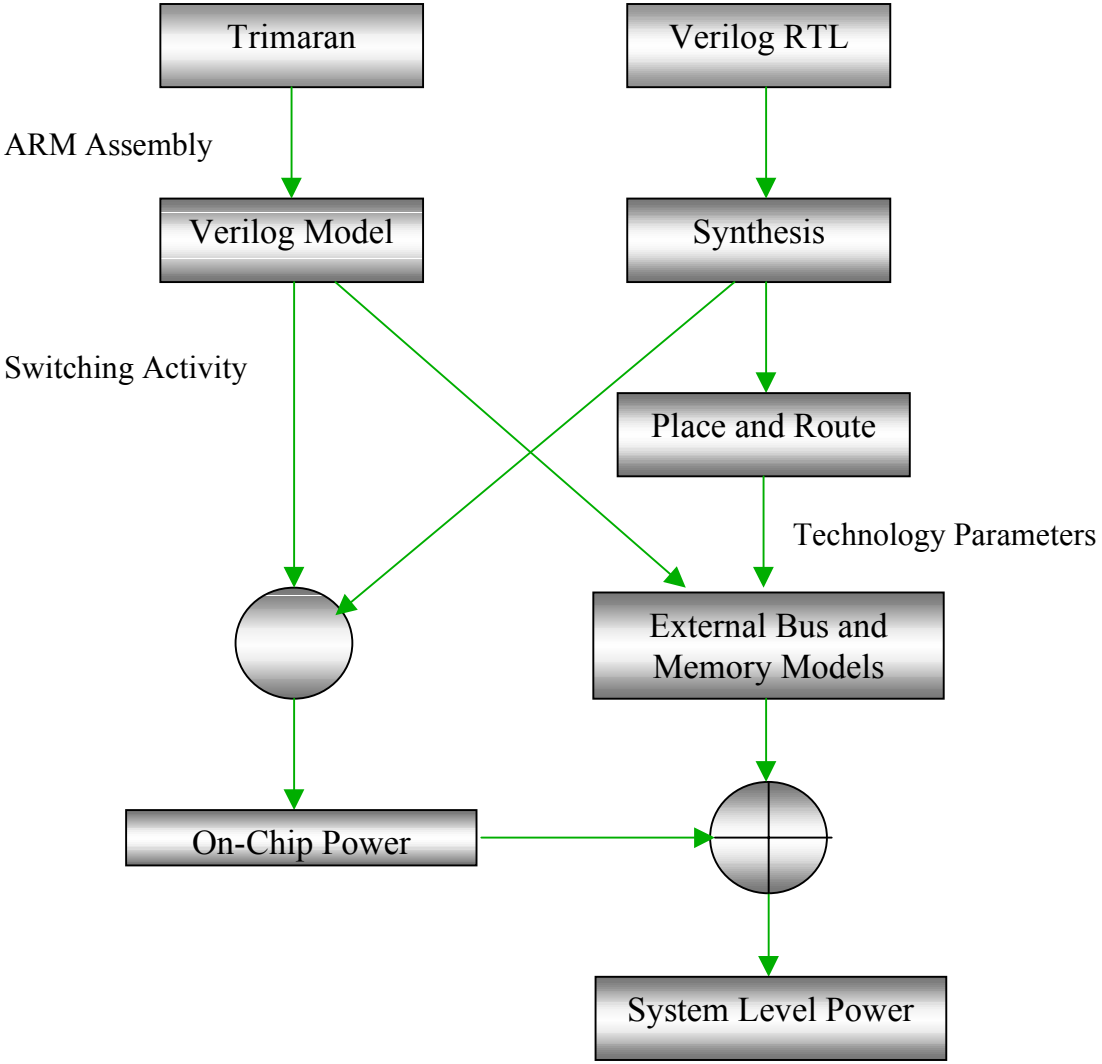
[Ref.] Dake Liu and Christer Svensson, " Power Consumption Estimation in CMOS VLSI Chips", IEEE Journal of Solid-State Circuits, Vol. 29, No.6, June 1994.

# Skiff Power Measurements:

- The current to the core flows through a 20mOhm resistor
- Measurement of the voltage drop on the 20mOhm resistor using Keithley sourcemeter
- 0.012 % basic accuracy with 5.5 digit resolution
- Voltage range of 1uV to 211V



# Experiment Methodology

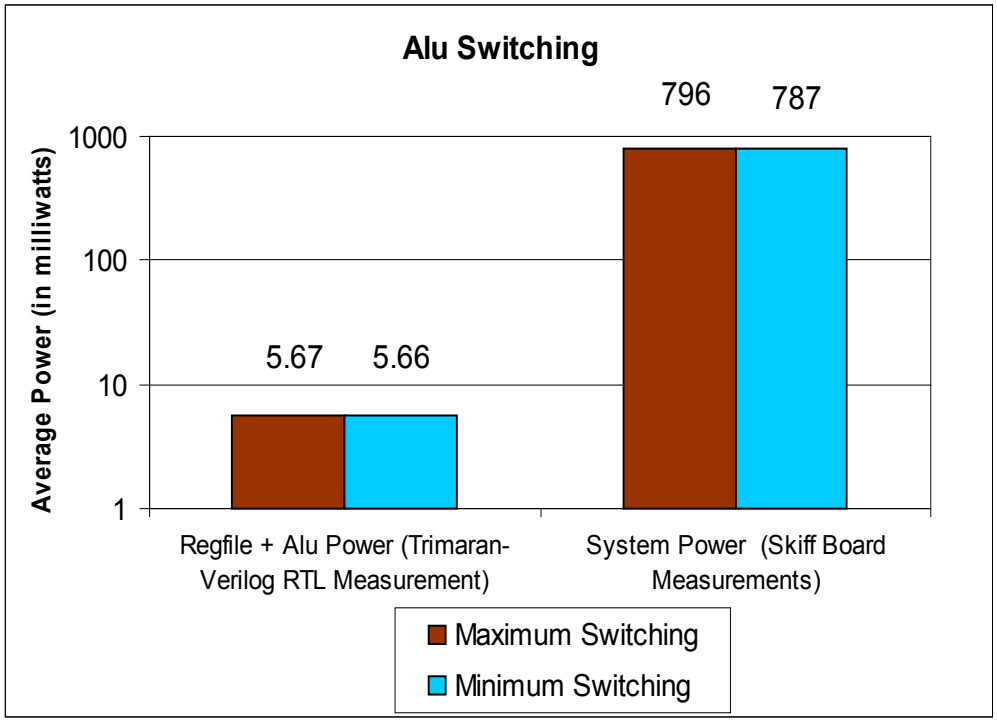


# Experiments

- Experiments to study effect of optimizations on different subsystems of the architecture
  - The ALU subsystem
  - The register file
  - Data and instruction cache
- Optimized and un optimized code run on the Verilog model and StrongARM board
  - Comparative study of the power dissipation

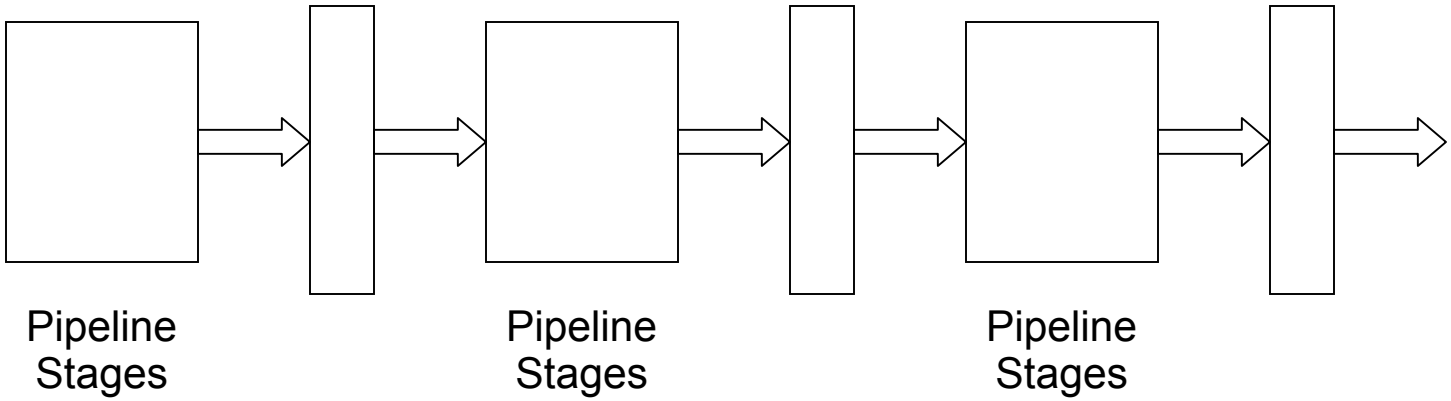
# The ALU Subsystem

- *Does reduction in switching activity reduce power ?*
  - Two sections of code each computing One optimized for minimal switching of inputs, the other for maximum switching
  - Hamming distance used as a measure of switching
  - Applicability of this technique should be explored further



# Intuition

- Minimizing ALU switching does not translate into power savings




- The ALU itself consumes power
- But we are not able to modulate it by controlling the input data
  - A major fraction is spent just pushing the data and control signals through the pipeline

# The ALU Subsystem

- *Do all types of instructions consume the same amount of power ?*
  - Different types of instructions were run in a loop and power numbers collected
- Logical operations, add, sub consume the same amount of power
- Multiply consumes about 30% more power and takes more cycles to execute
  - Strength reduction would be beneficial for power and energy savings
  - Instruction count should not be increased by more than 30%

# The Register File

- *Does the value accessed from the registers affect power ?*
  - Examples where instructions access values from registers that cause maximum, intermediate and minimum switching

- Combined Register File and ALU power varies by 12%
  - Possible optimization by instruction scheduling to reduce switching of value accessed from registers

| | Regfile + ALU Power in mw (Trimaran Verilog) | System Power in mw (Skiff Board) |
|---|---|---|
| Maximum Switching | 5.573 | 769 |
| Intermediate Switching | 5.105 | 736 |
| Minimum Switching | 4.978 | 708 |

# The Register File

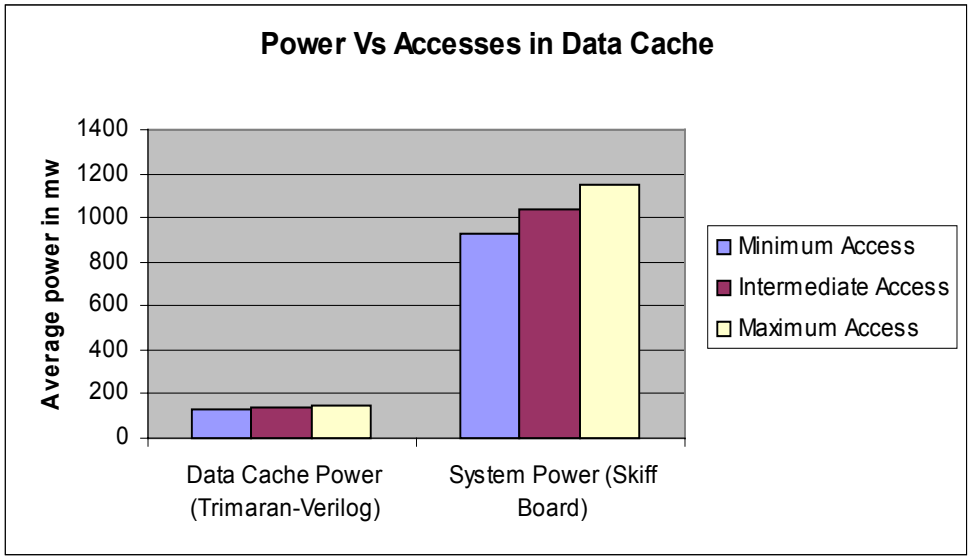
- *Do the number of accesses to the register file play a part in power consumption ?*
  - Two experiments, one that accesses values from registers, the other having immediate operands

| | ALU + Reg File Power in mw (Trimaran-Verilog) | System Power in mw (Skiff Board) |
|---|---|---|
| Register Operands | 4.784 | 776 |
| Immediate Operands | 4.784 | 760 |

- System power shows a difference but not the model
  - Due the architecture of the model
  - Optimizations include aggressive copy propagation and immediate addresses whenever possible

# The Cache Subsystem

- *Does the number of cache access contribute to power consumption ?*
  - Code having instructions that access the data cache 0%, 50% and 100% of the times




- About 24% difference between no access and full access to the cache

# The Taxonomy

- Class A: Energy benefit due to performance improvement
  - Energy = Ave. power dissipated per cycle × No. of cycles
  - Loop unrolling, reduction of load stores, partial redundancy elimination etc
- Class B: Benefit energy, no impact on performance
  - Innovations in instruction scheduling, register pipelining, code selection to replace high power dissipating instructions
- Class C: Negative impact on power dissipation and energy consumption
  - Typically optimizations that have negative impact on performance

# Recommendations

- To the compiler designer
  - Highest impact is by improving performance
  - Instruction scheduling to minimize register file switching
  - Strength reduction and proper code selection to replace power hogging instructions

- To the architect
  - *Novel* compiler optimizations that target power are few
  - More architectural innovations need to be exposed to the compiler
  - Bit width sensitive ALU, compiler controlled voltage and clock scaling etc

# Conclusion

- Compiler optimizations for locality and performance translate into power and energy savings
- Novel optimization opportunities like scheduling to reduce register file switching and use of immediate operands
- To obtain substantial power and energy savings innovating micro architectural features and exposing them to the compiler is necessary