Microarchitecture Support for Interconnect Power-aware Instruction Permutation

Hui Lin, Md. Sajjad Rahaman and
Prof. Masud H Chowdhury
ECE Department, University of Illinois at Chicago

Outline

- Introduction & Background
- Problem Statement
- Detail Design to Reorder Instructions
- Experiments Analysis

Introduction

Power Consumption Interconnect Bus

- VLSI technology steps into the deep sub-micrometer (DSM) level
  - Resulting in reduction on component spacing
  - Resulting in complication of interconnect network
- Self and coupling capacitance badly affects power consumption
  - Effective values are affected by physical features as well as dynamic bit values

Background Work

- Physical approaches to reduce the power consumption on interconnect bus
  - Shielding line
  - Reformat the physical interconnect lines
- Logic approaches
  - Encoding/decoding: operation can change the resulting bits values as well as the switching activity
  - Compiler design: change the instructions order or register name

Separating Program Behavior in Processor and at Interconnect

- Processor execute instructions from the binary code stored in memory
  - It only retrieves instructions from the Cache.
- Interconnect bus transmits instructions, data values, as well as commands
- Reorder instructions in the memory which reduces the switching activity during transmission
- Operations is performed during every cache miss

Problem Statement

Interconnect Bus Model

- RC interconnect model is used describe interconnect bus connecting processor and memory
- Dynamic power is represented as
  - $P=\alpha(C_c+C_g)V_{dd}^2 f$ or
  - $P=(1+N_c\lambda)\alpha C_g V_{dd}^2 f$
  - N_c is used to describe dynamic factor as bit values change

Design Objective

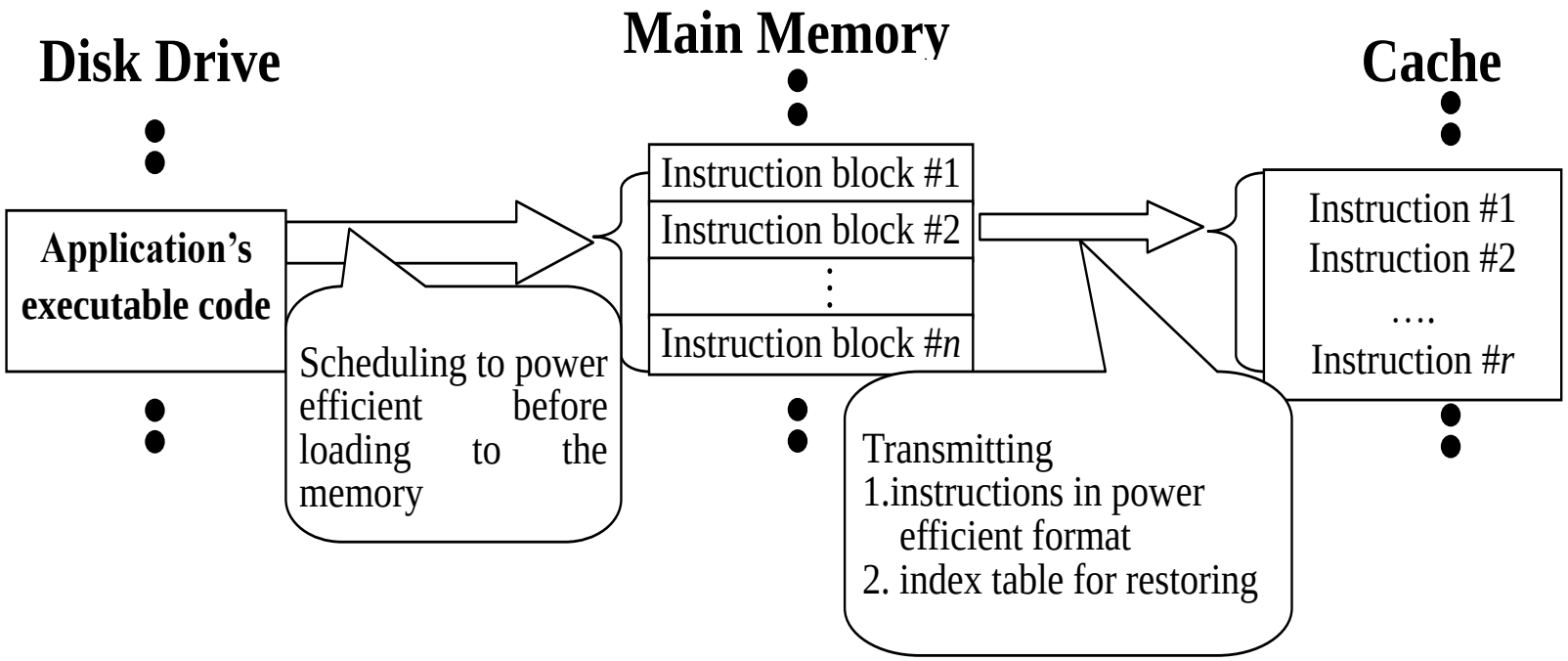
- Reorders program's binary code stored in memory in the size of cache line into the power efficient format
- Reduce the overhead of index table which is used to reinstate the instruction in the cache

Detail Design to Reorder Instructions

Overview of Permutation Algorithm

- When instructions are stored in the memory, reorder instructions according to cache line size
  - No consideration on data or control dependency
  - Could be done offline
  - Construct 1D index table which is used to recover instructions original order
- How to manage such index table?
  - Its transmission requires extra power consumption. In the same level of transmission of reformatted instruction block

Framework of Power Efficient Instruction Rescheduling



Instruction Permutation

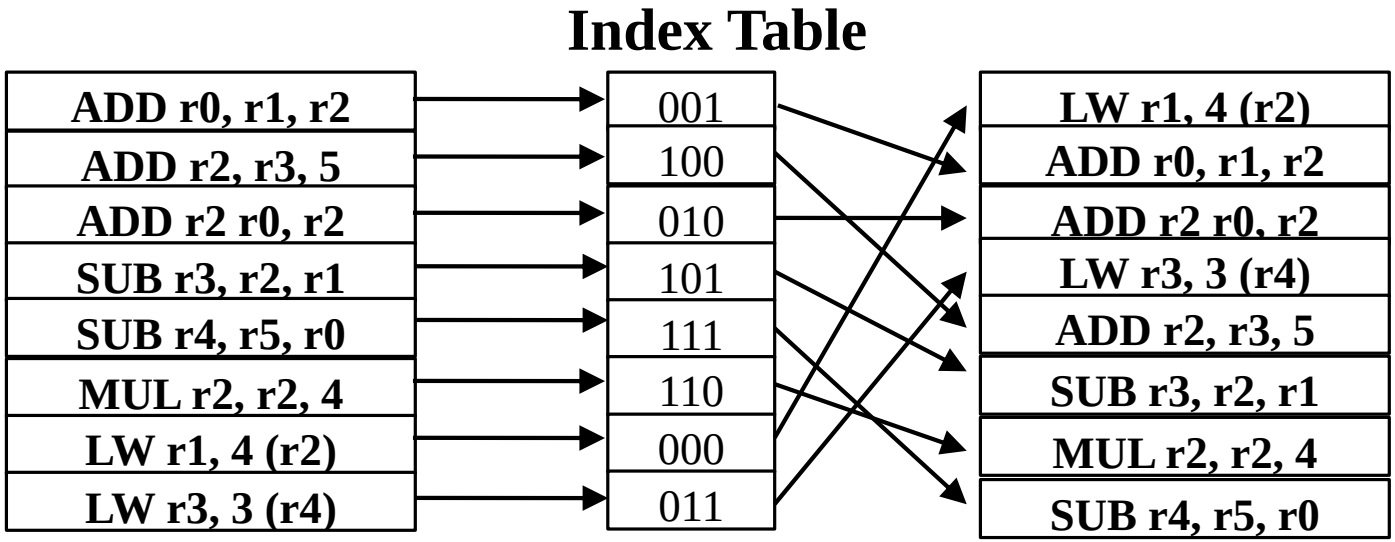
- Bit values within instructions are correlated to each other
  - Divided into several field: opcode, source register, destination register
  - RISC instruction set is adopted
  - Reorder instruction from field to field

Instruction Permutation (cont'd)

- Grouping instructions with same opcode
  - Number of opcode used is very limited
  - Efficient algorithm existed
  - Further algorithm design does not jeopardize the power reduction done before
- Reordering within each group
  - With small size, inefficient algorithm can be used

Index Table Construction

- The index table contains relationship between instructions in its original format and ones in its power efficient format
- Relative location of instructions are recorded



Index Table Overhead Analysis

- Power consumption
  - Index table is sent before instructions
  - Transmission of Index table itself will cause same level of power consumption as the reordered instructions do
  - Reduce the frequency of transmission: store all of them into processor
  - Storing index table also introduces extra power consumption
    - Not dominating as technology goes to small level

Index Table Overhead Analysis (Cont'd)

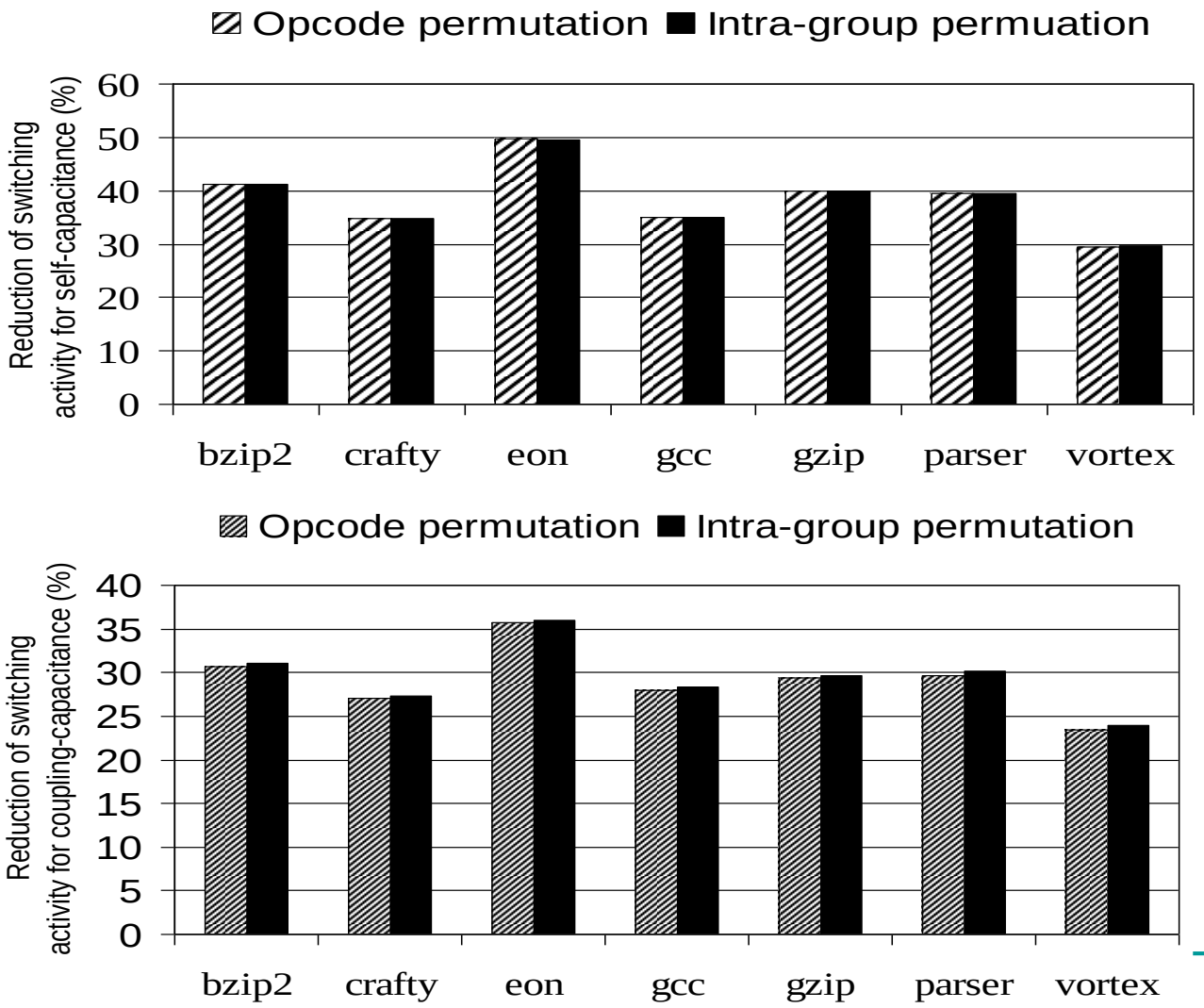
- Area overhead
  - Reorder operations is much easier than the encoding/decoding circuit
  - Storing space is small comparing to the cache size
- Performance overhead
  - The performance comes from accessing index table
  - Increase the cache miss penalty

Experiments Analysis

Power Reduction Achieved

- Switching activity is counted for both original instructions format and the power efficient format
- Self and coupling capacitance are calculated separately
- The effect of grouping and intra-group permutation are compared
- Overall complexity can be regarded as $O(N\log_2 N)$

Power Reduction Result



Index Table Overhead

- In order to reduce the frequency to transmit index table during the runtime, storing all of them in the processor
- Implemented as a level-2 cache: 20k byte size can meet most application programs
- Area, power consumption and access latency are get from the simulation under CACTI 5.0

Area, Power & Access Time Overhead

Technology node	Overhead criterion	L2 Cache (256k)	Index Table (20k)	Overhead
90nm	Area (mm^2)	7.6317	1.0682	14.0%
	Power (W)	0.62060	0.16960	27.3%
	Access time (ns)	1.6857	0.86934	51.6%
65nm	Area (mm^2)	3.9857	0.55717	14.0%
	Power (W)	0.47897	0.13750	28.7%
	Access time	1.1932	0.58601	49.1%
32nm	Area (mm^2)	0.96800	0.13506	14.0%
	Power (W)	0.43274	0.08099	18.7%
	Access time (ns)	0.60690	0.25402	41.9%

Any questions?

Thanks