

# CPSC614: Computer Architecture

E.J. Kim	<b>Homework #3</b>	Spring 2008
TA: Yuho Jin	Due (source code): April 15 2008 (Tue), 11:59 PM	
TA: Baiksong An	Due (report hard-copy): April 16 2008 (Wed), 2:50 PM	

## 0 Introduction

These problems are meant to help you better understand cache design and its simulation.

- **Homework #1 Solution**

Source codes (modified only) for Part 1 are in `~yuho/cpsc614/sp08/hw1/sol_part1` directory.

- **Note for Homework #3**

We keep `-fastfwd` parameter unit in SimpleScalar as 1000 instructions used in Homework #1. If `-fastfwd` is 2000000, `sim-outorder` will fastforward 2 billion instructions, not 2 million. SPEC2000 and `ref` input data set are used for CPU workload. Default machine is specified as `hw3.cfg` file.

- **Cacti**

We will use Cacti 4.1 for cache performance model. Executable for Linux is `~yuho/cpsc614/sp08/hw3/cacti`. The following shows Cacti command line:

```
$ cacti C B A TECH NSubbanks
```

C, B, A, TECH, and NSubbanks represent cache size in bytes, block size in bytes, associativity, feature size of technology, and number of sub-banks, respectively. We recommend reading technical report for Cacti 2.0 at [http://www.hpl.hp.com/personal/Norman\\_Jouppi/cacti4.html](http://www.hpl.hp.com/personal/Norman_Jouppi/cacti4.html), although each generation has a separate technical report. You can also download source codes.

- **Necessary Files**

Machine configuration	<code>linux.cs.tamu.edu ~yuho/cpsc614/sp08/hw3/hw3.cfg</code>
Cacti 4.1	<code>linux.cs.tamu.edu ~yuho/cpsc614/sp08/hw3/cacti</code>
SPEC2000 commands	<code>http://kbarr.net/specint2000-commandlines</code> <code>http://kbarr.net/specfp2000-commandlines</code>
SPEC2000 Alpha binary	<code>linux.cs.tamu.edu ~yuho/cpsc614/sp08/spec2000binary.tgz</code>

## 1 Cache Designs

Modern processors have multi-level caches in memory hierarchy for both performance and cost. The cache design space is a quite large such as block placement, block identification, block replacement, and write strategy. Generally, a larger cache can achieve better performance by reducing cache misses. However, doubling cache capacity does not always result in halving cache misses because of spatial and temporal locality. Also the larger cache causes longer access time, higher power dissipation, and larger area. Similarly, set-associative caches that have multiple blocks for one set can significantly reduce conflict misses at the cost of the tag array.

In this homework, we model L2 cache characteristics using Cacti and apply them for performance simulation using SimpleScalar. We fix the L2 cache capacity as 4MB (not including tags) and change its structure for five machines (**M1** – **M5**).

**M1**: 64B block direct-mapped 4MB

**M2**: 64B block 4-way set-associative 4MB

**M3**: 64B block 16-way set-associative 4MB

**M4**: 128B block direct-mapped 4MB

**M5**: 128B block 2-way set-associative 4MB

Machines	<b>M1</b>	<b>M2</b>	<b>M3</b>	<b>M4</b>	<b>M5</b>	<b>M3-LFU</b>
Access latency (ns)						
Access latency (cycles)						
Read Energy (nJ)						
Write Energy (nJ)						
Area ( $mm^2$ )						
IPC						
Hit rate (L2 cache)						
Total Dynamic Energy (L2 cache)						

Table 1: Cache Organization and Performance

## 1.1 Cache Modeling from Cacti

Run Cacti to obtain the access latency, energy consumption, and area for above five machines. Set the number of sub-banks as 1. We assume 45nm technology and 4GHz clock frequency. Cacti will use 0.7V supply voltage for 45nm. Fill the first five rows in the Table 1. For “Read (Write) Energy”, use values for “Total dynamic Read (Write) Energy all Banks”.

Compare latency, energy consumption, and area of the cache for each machine. Explain sub-banking and discuss pros and cons of sub-banking.

## 1.2 Cache Design Effect on Performance

We will use `sim-outorder` for experiments. Change parameters related to unified L2 cache in `hw3.cfg` for each machine according to Part 1.1 results. Except L2 cache, all machines have the same hardware organization. Make sure that the L2 caches you specify are all 4MB in total size. All set-associative caches will use LRU replacement policy. Run `sim-outorder` with `lucas` or `swim` benchmarks by fastforwarding 2G instructions and executing 100M instructions. Fill the last three rows in Table 1.

Use the following simple formula to get Total Dynamic Energy for L2 cache.

$$[Total\ Dynamic\ Energy] = [Read\ Energy] \times [Total\ Accesses] + [Write\ Energy] \times [Total\ Replacements]$$

For better energy modeling, read this paper, *D. Brooks, V. Tiwari, and M. Martonosi, Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, 27th International Symposium on Computer Architecture (ISCA), 2000.*

Give your analysis in terms of performance and energy consumption. Which machine gives the best hit rate? Which machine gives the best IPC? Which machine gives the lowest energy consumption? What conclusions do you draw from experiment? (*Optional*: If you understand power/energy concept well, you are welcome to include leakage energy consumption in your analysis and can use power or energy-efficiency metrics instead of energy metric.)

### 1.3 Replacement Policy

Implement LFU (Least Frequently Used) replacement policy in SimpleScalar. Each cache block will have a frequency counter. We model it as a 4-bit saturation counter, which limits the largest value as  $2^4 - 1$ . If a counter of one block reaches a saturation point, all the counters of other blocks that belong to the same set (for set-associative caches) decrease the counter value by one-bit shift operation. In one set that has multiple blocks for the same block address, the most frequently used block must be positioned as the first way and the least frequently used one must be the last way. The modified simulator must take care of LFU as `x` like existing replacement policies (`l` for LRU, `r` for Random, and `f` for FIFO) and must report hit percentage of each way in total hits.

Run `sim-outorder` with the same benchmark, configuration (`hw3.cfg`), instruction fastforwarding, and instruction execution as given in Part 1.2. Fill the last column of Table 1 and compare LFU results (including hit distribution on different ways) with LRU results for only machine **M3** (64B block 16-way set-associative 4MB L2 cache). Discuss the benefits of each replacement policy.

## 2 Turnin Process

Turn in a short written report (hard-copy only) in which, you show Table 1, answer questions, and summarize your findings (Due: April 16, 2:50 PM, class room). Arrange your results with graphs for ease of data interpretation.

Turn in only modified source files as one gzip compressed file for Part 1.3 in csnet (Due: April 15, 11:59 PM). Your submitted files will be tested in Linux machine.