

Cache Design

Computer Architecture CE-6304

3/14/2014

University of Texas at Dallas

Deepak Mugthihalli Devaprasad Paresh Naik

**Table of Contents**

**Part 1: Project Description3**

**Part 2: Calculation of CPI4**

**Part 3: Optimize CPI for each Benchmark 6**

**Part 4: Cost Function10**

**Part 5: Optimize cache for performance/cost 12**

**INFERENCE15**

**PART 1 : Project Description**

Cache design choices (i.e. # of levels, size, associativity, replacement policy etc.) affect the performance of a microprocessor. In this project, you are asked to fine-tune the cache hierarchy of an Alpha microprocessor for 4 individual benchmarks. The cache design parameters that can be modified are:

* Cache levels: One or two levels, for data and instruction caches.
* Unified caches: Selection of separate vs. unified instruction/data caches. For example, you can have separate L1 caches and a unified L2 cache.
* Size: Cache size, one of the most important choices.
* Associativity: Selection of cache associativity (e.g. direct mapped, 2-way set associative, etc.).
* Block size: Block size of the cache, usually 64 or 32 bytes.
* Block replacement policy: Selection between FIFO, LRU and Random.
* Miss Penalty: For L1 = 6 cycles; For L2 = 50 cycles
* L1 Hit Time: 1 cycle

**Part 2: CPI Calculation**

Calculation of the CPI for the three individual benchmarks for the below Alpha 21264 EV6 configuration:

- **Cache levels**: Two levels.

- **Unified caches**: Separate L1 data and instruction cache, unified L2 cache.

- **Size**: 64K Separate L1 data and instruction caches, 1MB unified L2 cache.

- **Associativity**: Two-way set-associative L1 caches, Direct-mapped L2 cache.

- **Block size**: 64 bytes.

- **Block replacement policy**: FIFO.

**CPI formulae used:**

**L1 & L2 unified**

CPI = 1 + DL1\_AccessPerInst\*DL1\_miss\_rate\* Miss\_Penalty 1 + DL2\_AccessPerInst\*DL2\_miss\_rate\*Miss\_Penalty2

**L1 separate & L2 unified (used for Part 2 calculation)**

CPI = 1 + 1\*IL1\_miss\_rate\* Miss\_Penalty 1 + DL1\_AccessPerInst\*DL1\_miss\_rate\* Miss\_Penalty 1 + DL2\_AccessPerInst\*DL2\_miss\_rate\* Miss\_Penalty 2

**L1 & L2 separate**

CPI = 1 + 1\*IL1\_miss\_rate\* Miss\_Penalty 1 + DL1\_AccessPerInst\*DL1\_miss\_rate\* Miss\_Penalty 1 + DL2\_AccessPerInst\*DL2\_miss\_rate\* Miss\_Penalty 2 + IL2\_AccessesPerInst\*IL2\_miss\_rate\* Miss\_Penalty 2

**CPI for ANAGRAM** :

Total number of Instructions : 25593183

IL1 accesses : 25593183

IL1 miss-rate : 495/25593183 = 0.0000193

DL1 accesses : 11153900  
DL1 miss-rate : 54246/11153900 = 0.0049

UL2 accesses : 92143

UL2 miss-rate : 29019/92143= 0.3149

**CPI = 1.06961032**

**CPI for GCC** :

Total number of Instructions : 337326966

IL1 accesses : 337326966

IL1 miss-rate : 1587719/337326966 = 0.0047

DL1 accesses : 124102754  
DL1 miss-rate : 1318410/124102754 = 0.0106

UL2 accesses : 1742443

UL2 miss-rate : 159779/1742443= 0.0917

**CPI = 1.075279965**

**CPI for GO** :

Total number of Instructions : 545823529

IL1 accesses : 545823529

IL1 miss-rate : 761652/545823529 = 0.0013954

DL1 accesses : 213791066  
DL1 miss-rate : 137444/213791066 = 0.00064289

UL2 accesses : 3330118 (200037)

UL2 miss-rate : 36560/200037 = 0.1827662

**CPI = 1.01172301**

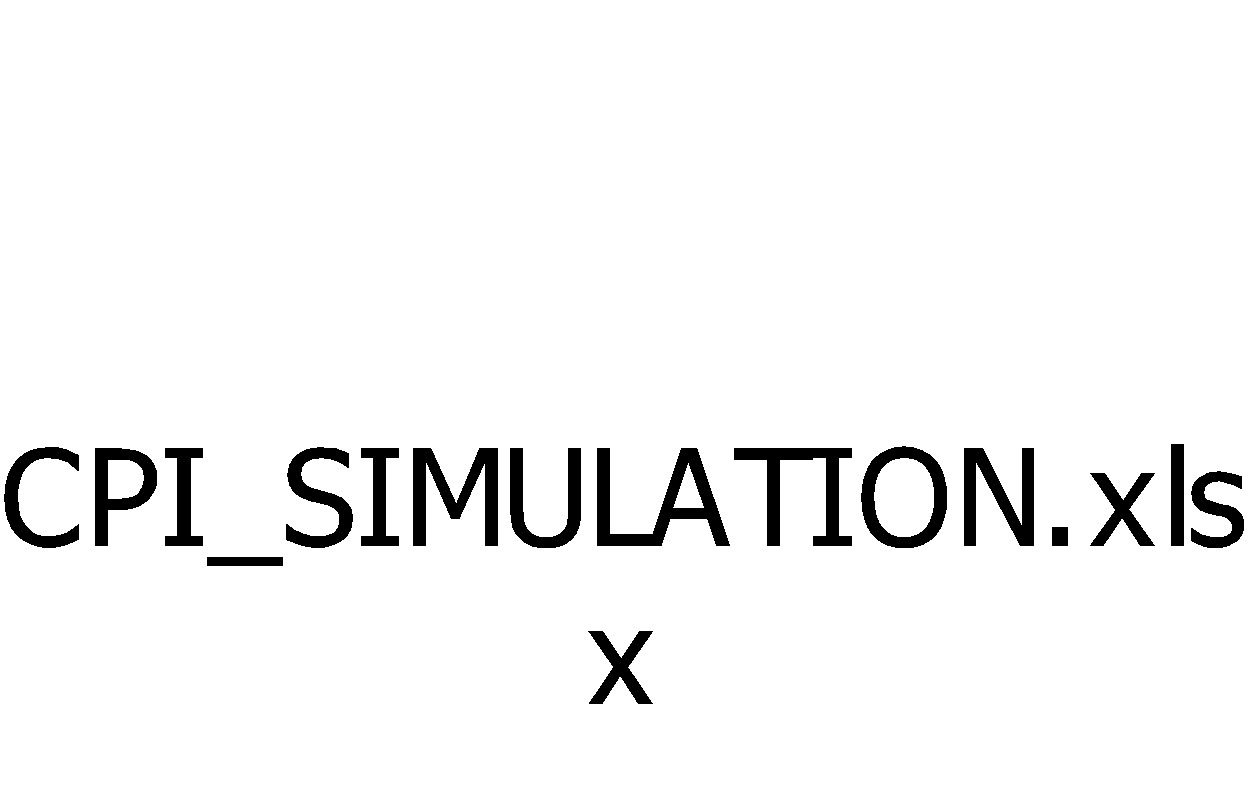
**Part 3 : Optimize CPI for each Benchmark**

In this part, various configurations were developed for the cache design for various cache kinds’ i.e.; L1 unified and L2 unified L1 separate and L2 unified, and L1 separate and L2 separate. The CPI for each configuration was calculated and plotted for each cache configuration. It is given that the amount of L1 cache available is 128KB and that for L2 is 1MB. The following are the values considered for each parameter while determining the optimum cache design (in terms of CPI) for each of the benchmarks:

* L1 Separate data and instruction cache (64KB each), L2 Unified data and instruction cache (1MB)
* L1 Separate data and instruction cache (64KB each), L2 Separate data and instruction cache (512KB each)
* L1 Unified data and instruction cache (128KB), L2 Unified data and instruction cache (1MB)
* Block size : 32 bytes, 64 bytes
* Associativity: 1-way, 2-way, 4-way, 8-way. Design does not consider associativity more than 8 because they have much higher cost for a very little performance improvement in reality.
* Replacement Policy: FIFO (f), Random(r), LRU (l), for different combination of L1 and L2.
* Number of Sets: this parameter is calculated from the above parameters using the formula

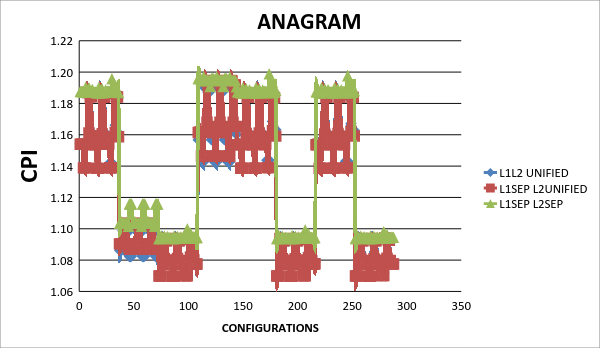
**No. of Sets = (Cache Size)/(Associativity \* Block Size)**

This method resulted in 288 configurations for each benchmark and each of the cases: L1 separate, L2 unified; L1 Separate, L2 separate and L1 Unified and L2 Unified. The list of these configurations with configuration numbers for L1 and L2 separate case is given in the excel sheet attached.

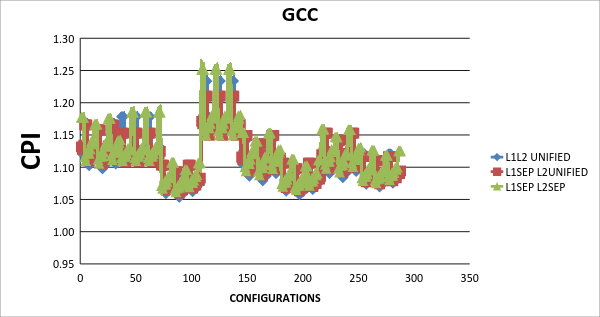


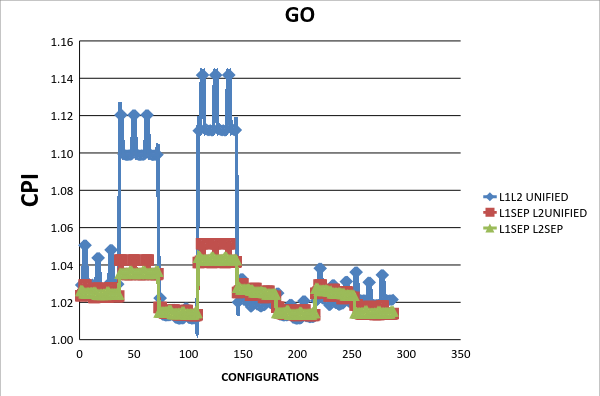
The plot for CPI against different configurations for L1 Separate-L2 separate, L1 Separate L2 unified, and L1,L2 unified for each benchmarks is shown below.

1. **ANAGRAM**



1. **GCC**



**3.GO**

**Conclusion for Part 3:**

From the above graphs, the following configurations are the optimum for each benchmarks.

**ANAGRAM:**

L1 Separate L2 Unified

L1 – 4-way associativity with LRU replacement policy

L2 - 8-way associativity with FIFO replacement policy

Block Size : 64 bytes

Optimum CPI: 1.06972452322260

**GCC:**

L1 Unified L2 Unified

L1 – 8-way associativity with LRU replacement policy

L2 - 4-way associativity with LRU replacement policy

Block Size : 64 bytes

Optimum CPI: 1.052968410415380

**GO:**

L1 Unified L2 Unified

L1 – 8-way associativity with Random replacement policy

L2 - 2-way associativity with LRU replacement policy

Block Size : 64 bytes

Optimum CPI: 1.011049036910210

**Part 4: Cost Function**

Let

    Associativity of L1 be      A1

     Associativity of L2 be      A2

     Replacement policy be    R

     Block Size be                    BS

     Cost of Splitting L1 be     S1

     Cost of splitting L2 be     S2

     Cost Factor for L1          CF1

     Cost Factor for L2          CF2

Assumption:

        Let cost of

     R for Random      = 10

                  R for FIFO           = 30

                  R for LRU            = 80

                  Cost for 1-way        = 10

                  Cost for 2-way        = 20

                  Cost for 4-way        = 40

                  Cost for 8-way        = 80

                  S1                         = 2

                  S2                         = 1

                  CF1 = 5

     CF2          = 1

When the associativity of the cache increases the area of the cache also increases, which results in increase in cost of the cache. So, higher associativity leads to higher cache cost as hardware required is more.

In our design the cost associated with increase in associativity is proportional to n-way associativity (e.g.: 4-way cost is 4).

Since the L1 cache is of smaller size with greater associativity and is placed near the processor than L2 cache, L1 cache is faster than L2 cache. Hence the cost associated with L1 is higher than L2 . This factor is included in our design as Cost Factor(CF) term. The CF associated with L1 is 5 and L2 is 1.

Considering the replacement policy, changing form Random to FIFO increases the cost only by a smaller amount. LRU replacement is much costlier than random and FIFO. Assuming that changing the replacement for Random to FIFO increases the cost only by 3% and changing the replacement for Random to LRU increases the cost by 8% . So, changing the replacement policy form FIFO to LRU increases the cost by 5%. Hence, we got the above values for replacement R.

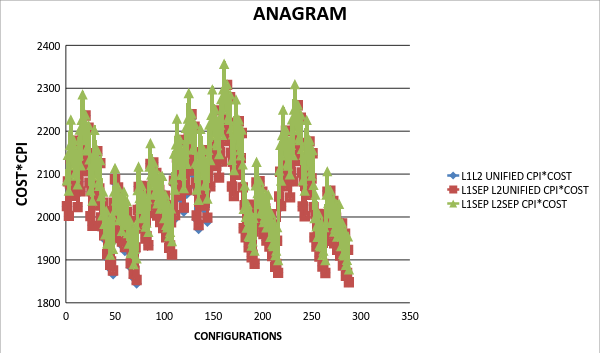
Splitting the cache also results in increase in cache cost. Since they require additional hardware and it also results in increased bandwidth. Hence the cost for splitting the cache is also added to the total cost.

Hence, **Total cost = (Cache sizeL1\*CF1+S1) + (Cache sizeL2\*CF2+S2) +A1+A2+R1+R2**

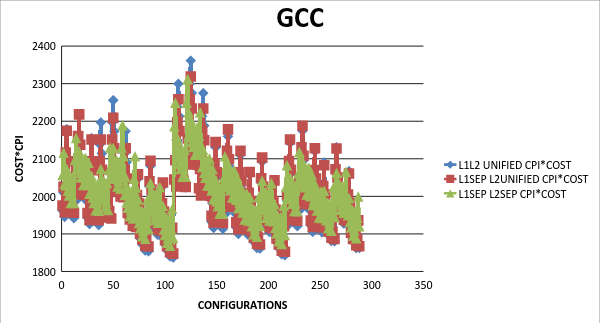
**Part 5: Optimize cache for performance/cost**

For Finding the optimal configuration for cache, various factors like associativity, Replacement policy, cache types is considered along with the calculated cost. The graph of cache configuration for each benchmark is plotted against the product of CPI and cost. The optimal configuration is the one which gives the minimum value of product of cost and CPI among all in the graph.

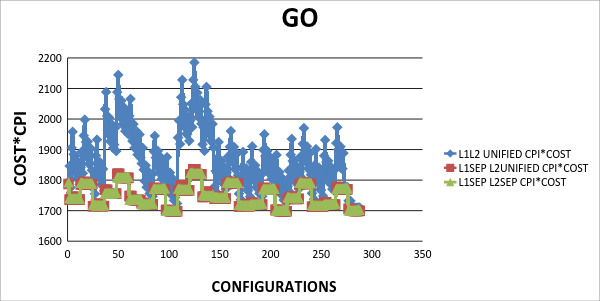
**ANAGRAM:**



**GCC:**



**GO:**



**OPTIMUM CONFIGURATION FOR EACH BENCHMARKS:**

**ANAGRAM:**

L1 Unified L2 Unified

L1 – 1-way associativity with Random replacement policy

L2 - 1-way associativity with Random replacement policy

Block Size : 64 bytes

**Cost \* CPI : 1844.88**

**GCC:**

L1 Unified L2 Unified

L1 – 8-way associativity with Random replacement policy

L2 - 1-way associativity with Random replacement policy

Block Size : 64 bytes

**Cost \* CPI : 1837.17**

**GO:**

L1 Separate L2 Unified

L1 – 8-way associativity with Random replacement policy

L2 - 1-way associativity with FIFO replacement policy

Block Size : 64 bytes

**Cost \* CPI : 1697.67**

**MOST OPTIMUM CONFIGURATION:**

Among all the three benchmarks, the following is the best configuration.

**GO Benchmark:**

L1 Separate L2 Unified

L1 – 8-way associativity with Random replacement policy

L2 - 1-way associativity with FIFO replacement policy

Block Size : 64 bytes

**Cost \* CPI : 1697.67**

**Table for average CPI for each benchmark for different cache level configuration:**

For each benchmark and for each cache level configuration, average CPI is calculated as below.

**CPIaverage = (∑ CPIcachelevel ) / Number of total configurations**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **L1 & L2 UNIFIED** | **L1 SEPARATE L2 UNIFIED** | **L1 SEPARATE L2 SEPARATE** |
| **ANAGRAM** | ***1.1235276556*** | 1.2390230123 | 1.1435955533 |
| **GCC** | ***1.1138027857*** | 1.1155884638 | 1.2168148290 |
| **GO** | 1.0442130204 | 1.0249045216 | ***1.0248269225*** |

Looking at the above table, the optimal configuration, based on average CPI is for **GO** benchmark with **L1 SEPARATE L2 SEPARATE.**

**INFERENCE:**

By varying different cache parameters such as cache type, associativity and replacement policy, we arrived at optimal cache configuration in terms of average CPI, CPI\*Cost and CPI.