21 Oct 2013

**Index**

Cache Design

Computer Architecture : CE6304

Saisagar Setra (2021186254)

Akhileshrao Shivarpatna(2021168042 )

|  |  |  |
| --- | --- | --- |
| Sl No. | Contents | Page Number |
| I | Project Description | 2 |
| II | Approach | 2 |
| III | Part 2 : Find CPI | 4 |
| IV | Part 3 : Optimize CPI for each Benchmark | 5 |
| V | Part 4: Define Cost Function | 21 |
| VI | Part 5 : Optimize caches for performance/cost | 24 |
| VII | Appendix: Scripts | 36 |

1. **Project Description :**

This project aims at designing a Cache with optimum configuration in relation to the cost function for the Alpha 21264 EV6 configuration against the three benchmarks. The cache design parameters are given as follows:

* Cache levels: One or two levels, for data and instruction caches.
* Unified/Separate caches: Selection of separate vs. unified instruction/data caches.
* Size: Cache size
* 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 = 5 cycles; For L2 = 40 cycles
* L1 Hit Time: 1 cycle

1. **Approach :**

The configuration chosen to optimize the cache design with respect to minimizing CPI is Alpha 21264 EV6 configuration. The various cache configurations are tested with 3 benchmarks namely: GCC, GO, Anagram. These benchmarks are run with each cache configuration on a simulation tool Simplescalar V3.0.

There are 2 levels of cache design namely L1 and L2. There are 3 basic configurations:

L1 unified and L2 unified

L1 separate and L2 unified

L1 separate and L2 separate

The 4th configuration L1 unified and L2 separate is not possible to design as it has some inherent violations. For example, there is miss on a data access at L1, the block is to be retrieved from L2 Data cache. During the replacement, there is no information whether the block in L1 is instruction or data block.

The other parameters such as Block Size, Associativity, Replacement Policy are varied in following manner:

**Block Size** : It is maintained constant across the cache levels either at 32 or 64 bytes

**Associativity** : It is varied as follows : Direct-Mapped (1-way set associative), 2-way, 4-way, 8-way and Fully Associative. The associativity is kept constant at a particular cache level in our approach. Often associativity is the one of the parameters that determines the miss-rate. The assumption here is that, the parameter that affects the miss-rate at the top priority is the associativity. Hence this is being varied among the among the cache levels.

**Replacement Policy** : The replacement policy can be implemented in LRU, FIFO or Random policies. In our approach, the policy is maintained constant across the cache levels.

The CPI is calculated as follows :

**CPI = Ideal CPI + Memory Stall cycles per Instruction**

Ideal CPI = 1 = L1 hit time

Memory Stalls per Instruction for each cache level be it unified or separate will be as follows :

Memory Stalls per Instruction = Memory Access per Instruction \* Miss rate \* Miss Penalty

The terms from the Memory Stalls per instruction for each cache are added to get the overall Memory Stall cycles per Instruction for the design.

The CPI for each 3 basic cache configurations are as follows :

**L1 unified and L2 unified**

**CPI = 1 + DL1\_AccessPerInst\*DL1\_miss\_rate\*MPL1 + DL2\_AccessPerInst\*DL2\_miss\_rate\*MPL2**

**L1 separate and L2 unified**

**CPI = 1 + 1\*IL1\_miss\_rate\*MPL1 + DL1\_AccessPerInst\*DL1\_miss\_rate\*MPL1 + DL2\_AccessPerInst\*DL2\_miss\_rate\*MPL2**

**L1 separate and L2 separate**

**CPI = 1 + 1\*IL1\_miss\_rate\*MPL1 + DL1\_AccessPerInst\*DL1\_miss\_rate\*MPL1 + DL2\_AccessPerInst\*DL2\_miss\_rate\*MPL2 + IL2\_AccessesPerInst\*IL2\_miss\_rate\*MPL2**

MP stands for Miss Penalty

1. **Part 2 : Find CPI :**

The configuration specified for the CPI calculation is as follows :

* Cache levels: Two levels.
* Unified/Separate 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 for GCC** :

Total number of Instructions : 337327107

IL1 accesses : 337327107

IL1 miss-rate : 0.0047

DL1 accesses : 124102801  
DL1 miss-rate : 0.0106

UL2 accesses : 3330118

UL2 miss-rate : 0.1311

Using the above formula for CPI, we get the **CPI = 1.094767878**

**CPI for Anagram** :

Total number of Instructions : 25593324

IL1 accesses : 25593324

IL1 miss-rate : 0.0000

DL1 accesses : 11153947  
DL1 miss-rate : 0.0049

UL2 accesses : 92644

UL2 miss-rate : 0.3189

Using the above formula for CPI, we get the **CPI = 1.05685227**

**CPI for GO** :

Total number of Instructions : 545812859

IL1 accesses : 545812859

IL1 miss-rate : 0. 0013

DL1 accesses : 213788561  
DL1 miss-rate : 0. 0010

UL2 accesses : 1019533

UL2 miss-rate : 0. 0904

Using the above formula for CPI, we get the **CPI = 1.015212829**

1. **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 kind. 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 and fully associative.
* Replacement Policy: FIFO(f), Random(r), LRU(l)
* 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 150 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 table below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | DLRU1 | | | ILRU1 | | | ILRU2 | | | DLRU2 | | |
| Configuration Number | Block Size | Associativity | Replacement policy | Block Size | Associativity | Replacement policy | Block Size | Associativity | Replacement policy | Block Size | Associativity | Replacement policy |
| 1 | 32 | 1 | random | 32 | 1 | random | 32 | 1 | random | 32 | 1 | random |
| 2 | 64 | 1 | random | 64 | 1 | random | 64 | 1 | random | 64 | 1 | random |
| 3 | 32 | 1 | random | 32 | 1 | random | 32 | 2 | random | 32 | 2 | random |
| 4 | 64 | 1 | random | 64 | 1 | random | 64 | 2 | random | 64 | 2 | random |
| 5 | 32 | 1 | random | 32 | 1 | random | 32 | 4 | random | 32 | 4 | random |
| 6 | 64 | 1 | random | 64 | 1 | random | 64 | 4 | random | 64 | 4 | random |
| 7 | 32 | 1 | random | 32 | 1 | random | 32 | 8 | random | 32 | 8 | random |
| 8 | 64 | 1 | random | 64 | 1 | random | 64 | 8 | random | 64 | 8 | random |
| 9 | 32 | 1 | random | 32 | 1 | random | 32 | full | random | 32 | full | random |
| 10 | 64 | 1 | random | 64 | 1 | random | 64 | full | random | 64 | full | random |
| 11 | 32 | 2 | random | 32 | 2 | random | 32 | 1 | random | 32 | 1 | random |
| 12 | 64 | 2 | random | 64 | 2 | random | 64 | 1 | random | 64 | 1 | random |
| 13 | 32 | 2 | random | 32 | 2 | random | 32 | 2 | random | 32 | 2 | random |
| 14 | 64 | 2 | random | 64 | 2 | random | 64 | 2 | random | 64 | 2 | random |
| 15 | 32 | 2 | random | 32 | 2 | random | 32 | 4 | random | 32 | 4 | random |
| 16 | 64 | 2 | random | 64 | 2 | random | 64 | 4 | random | 64 | 4 | random |
| 17 | 32 | 2 | random | 32 | 2 | random | 32 | 8 | random | 32 | 8 | random |
| 18 | 64 | 2 | random | 64 | 2 | random | 64 | 8 | random | 64 | 8 | random |
| 19 | 32 | 2 | random | 32 | 2 | random | 32 | full | random | 32 | full | random |
| 20 | 64 | 2 | random | 64 | 2 | random | 64 | full | random | 64 | full | random |
| 21 | 32 | 4 | random | 32 | 4 | random | 32 | 1 | random | 32 | 1 | random |
| 22 | 64 | 4 | random | 64 | 4 | random | 64 | 1 | random | 64 | 1 | random |
| 23 | 32 | 4 | random | 32 | 4 | random | 32 | 2 | random | 32 | 2 | random |
| 24 | 64 | 4 | random | 64 | 4 | random | 64 | 2 | random | 64 | 2 | random |
| 25 | 32 | 4 | random | 32 | 4 | random | 32 | 4 | random | 32 | 4 | random |
| 26 | 64 | 4 | random | 64 | 4 | random | 64 | 4 | random | 64 | 4 | random |
| 27 | 32 | 4 | random | 32 | 4 | random | 32 | 8 | random | 32 | 8 | random |
| 28 | 64 | 4 | random | 64 | 4 | random | 64 | 8 | random | 64 | 8 | random |
| 29 | 32 | 4 | random | 32 | 4 | random | 32 | full | random | 32 | full | random |
| 30 | 64 | 4 | random | 64 | 4 | random | 64 | full | random | 64 | full | random |
| 31 | 32 | 8 | random | 32 | 8 | random | 32 | 1 | random | 32 | 1 | random |
| 32 | 64 | 8 | random | 64 | 8 | random | 64 | 1 | random | 64 | 1 | random |
| 33 | 32 | 8 | random | 32 | 8 | random | 32 | 2 | random | 32 | 2 | random |
| 34 | 64 | 8 | random | 64 | 8 | random | 64 | 2 | random | 64 | 2 | random |
| 35 | 32 | 8 | random | 32 | 8 | random | 32 | 4 | random | 32 | 4 | random |
| 36 | 64 | 8 | random | 64 | 8 | random | 64 | 4 | random | 64 | 4 | random |
| 37 | 32 | 8 | random | 32 | 8 | random | 32 | 8 | random | 32 | 8 | random |
| 38 | 64 | 8 | random | 64 | 8 | random | 64 | 8 | random | 64 | 8 | random |
| 39 | 32 | 8 | random | 32 | 8 | random | 32 | full | random | 32 | full | random |
| 40 | 64 | 8 | random | 64 | 8 | random | 64 | full | random | 64 | full | random |
| 41 | 32 | full | random | 32 | full | random | 32 | 1 | random | 32 | 1 | random |
| 42 | 64 | full | random | 64 | full | random | 64 | 1 | random | 64 | 1 | random |
| 43 | 32 | full | random | 32 | full | random | 32 | 2 | random | 32 | 2 | random |
| 44 | 64 | full | random | 64 | full | random | 64 | 2 | random | 64 | 2 | random |
| 45 | 32 | full | random | 32 | full | random | 32 | 4 | random | 32 | 4 | random |
| 46 | 64 | full | random | 64 | full | random | 64 | 4 | random | 64 | 4 | random |
| 47 | 32 | full | random | 32 | full | random | 32 | 8 | random | 32 | 8 | random |
| 48 | 64 | full | random | 64 | full | random | 64 | 8 | random | 64 | 8 | random |
| 49 | 32 | full | random | 32 | full | random | 32 | full | random | 32 | full | random |
| 50 | 64 | full | random | 64 | full | random | 64 | full | random | 64 | full | random |
| 51 | 32 | 1 | LRU | 32 | 1 | LRU | 32 | 1 | LRU | 32 | 1 | LRU |
| 52 | 64 | 1 | LRU | 64 | 1 | LRU | 64 | 1 | LRU | 64 | 1 | LRU |
| 53 | 32 | 1 | LRU | 32 | 1 | LRU | 32 | 2 | LRU | 32 | 2 | LRU |
| 54 | 64 | 1 | LRU | 64 | 1 | LRU | 64 | 2 | LRU | 64 | 2 | LRU |
| 55 | 32 | 1 | LRU | 32 | 1 | LRU | 32 | 4 | LRU | 32 | 4 | LRU |
| 56 | 64 | 1 | LRU | 64 | 1 | LRU | 64 | 4 | LRU | 64 | 4 | LRU |
| 57 | 32 | 1 | LRU | 32 | 1 | LRU | 32 | 8 | LRU | 32 | 8 | LRU |
| 58 | 64 | 1 | LRU | 64 | 1 | LRU | 64 | 8 | LRU | 64 | 8 | LRU |
| 59 | 32 | 1 | LRU | 32 | 1 | LRU | 32 | full | LRU | 32 | full | LRU |
| 60 | 64 | 1 | LRU | 64 | 1 | LRU | 64 | full | LRU | 64 | full | LRU |
| 61 | 32 | 2 | LRU | 32 | 2 | LRU | 32 | 1 | LRU | 32 | 1 | LRU |
| 62 | 64 | 2 | LRU | 64 | 2 | LRU | 64 | 1 | LRU | 64 | 1 | LRU |
| 63 | 32 | 2 | LRU | 32 | 2 | LRU | 32 | 2 | LRU | 32 | 2 | LRU |
| 64 | 64 | 2 | LRU | 64 | 2 | LRU | 64 | 2 | LRU | 64 | 2 | LRU |
| 65 | 32 | 2 | LRU | 32 | 2 | LRU | 32 | 4 | LRU | 32 | 4 | LRU |
| 66 | 64 | 2 | LRU | 64 | 2 | LRU | 64 | 4 | LRU | 64 | 4 | LRU |
| 67 | 32 | 2 | LRU | 32 | 2 | LRU | 32 | 8 | LRU | 32 | 8 | LRU |
| 68 | 64 | 2 | LRU | 64 | 2 | LRU | 64 | 8 | LRU | 64 | 8 | LRU |
| 69 | 32 | 2 | LRU | 32 | 2 | LRU | 32 | full | LRU | 32 | full | LRU |
| 70 | 64 | 2 | LRU | 64 | 2 | LRU | 64 | full | LRU | 64 | full | LRU |
| 71 | 32 | 4 | LRU | 32 | 4 | LRU | 32 | 1 | LRU | 32 | 1 | LRU |
| 72 | 64 | 4 | LRU | 64 | 4 | LRU | 64 | 1 | LRU | 64 | 1 | LRU |
| 73 | 32 | 4 | LRU | 32 | 4 | LRU | 32 | 2 | LRU | 32 | 2 | LRU |
| 74 | 64 | 4 | LRU | 64 | 4 | LRU | 64 | 2 | LRU | 64 | 2 | LRU |
| 75 | 32 | 4 | LRU | 32 | 4 | LRU | 32 | 4 | LRU | 32 | 4 | LRU |
| 76 | 64 | 4 | LRU | 64 | 4 | LRU | 64 | 4 | LRU | 64 | 4 | LRU |
| 77 | 32 | 4 | LRU | 32 | 4 | LRU | 32 | 8 | LRU | 32 | 8 | LRU |
| 78 | 64 | 4 | LRU | 64 | 4 | LRU | 64 | 8 | LRU | 64 | 8 | LRU |
| 79 | 32 | 4 | LRU | 32 | 4 | LRU | 32 | full | LRU | 32 | full | LRU |
| 80 | 64 | 4 | LRU | 64 | 4 | LRU | 64 | full | LRU | 64 | full | LRU |
| 81 | 32 | 8 | LRU | 32 | 8 | LRU | 32 | 1 | LRU | 32 | 1 | LRU |
| 82 | 64 | 8 | LRU | 64 | 8 | LRU | 64 | 1 | LRU | 64 | 1 | LRU |
| 83 | 32 | 8 | LRU | 32 | 8 | LRU | 32 | 2 | LRU | 32 | 2 | LRU |
| 84 | 64 | 8 | LRU | 64 | 8 | LRU | 64 | 2 | LRU | 64 | 2 | LRU |
| 85 | 32 | 8 | LRU | 32 | 8 | LRU | 32 | 4 | LRU | 32 | 4 | LRU |
| 86 | 64 | 8 | LRU | 64 | 8 | LRU | 64 | 4 | LRU | 64 | 4 | LRU |
| 87 | 32 | 8 | LRU | 32 | 8 | LRU | 32 | 8 | LRU | 32 | 8 | LRU |
| 88 | 64 | 8 | LRU | 64 | 8 | LRU | 64 | 8 | LRU | 64 | 8 | LRU |
| 89 | 32 | 8 | LRU | 32 | 8 | LRU | 32 | full | LRU | 32 | full | LRU |
| 90 | 64 | 8 | LRU | 64 | 8 | LRU | 64 | full | LRU | 64 | full | LRU |
| 91 | 32 | full | LRU | 32 | full | LRU | 32 | 1 | LRU | 32 | 1 | LRU |
| 92 | 64 | full | LRU | 64 | full | LRU | 64 | 1 | LRU | 64 | 1 | LRU |
| 93 | 32 | full | LRU | 32 | full | LRU | 32 | 2 | LRU | 32 | 2 | LRU |
| 94 | 64 | full | LRU | 64 | full | LRU | 64 | 2 | LRU | 64 | 2 | LRU |
| 95 | 32 | full | LRU | 32 | full | LRU | 32 | 4 | LRU | 32 | 4 | LRU |
| 96 | 64 | full | LRU | 64 | full | LRU | 64 | 4 | LRU | 64 | 4 | LRU |
| 97 | 32 | full | LRU | 32 | full | LRU | 32 | 8 | LRU | 32 | 8 | LRU |
| 98 | 64 | full | LRU | 64 | full | LRU | 64 | 8 | LRU | 64 | 8 | LRU |
| 99 | 32 | full | LRU | 32 | full | LRU | 32 | full | LRU | 32 | full | LRU |
| 100 | 64 | full | LRU | 64 | full | LRU | 64 | full | LRU | 64 | full | LRU |
| 101 | 32 | 1 | FIFO | 32 | 1 | FIFO | 32 | 1 | FIFO | 32 | 1 | FIFO |
| 102 | 64 | 1 | FIFO | 64 | 1 | FIFO | 64 | 1 | FIFO | 64 | 1 | FIFO |
| 103 | 32 | 1 | FIFO | 32 | 1 | FIFO | 32 | 2 | FIFO | 32 | 2 | FIFO |
| 104 | 64 | 1 | FIFO | 64 | 1 | FIFO | 64 | 2 | FIFO | 64 | 2 | FIFO |
| 105 | 32 | 1 | FIFO | 32 | 1 | FIFO | 32 | 4 | FIFO | 32 | 4 | FIFO |
| 106 | 64 | 1 | FIFO | 64 | 1 | FIFO | 64 | 4 | FIFO | 64 | 4 | FIFO |
| 107 | 32 | 1 | FIFO | 32 | 1 | FIFO | 32 | 8 | FIFO | 32 | 8 | FIFO |
| 108 | 64 | 1 | FIFO | 64 | 1 | FIFO | 64 | 8 | FIFO | 64 | 8 | FIFO |
| 109 | 32 | 1 | FIFO | 32 | 1 | FIFO | 32 | full | FIFO | 32 | full | FIFO |
| 110 | 64 | 1 | FIFO | 64 | 1 | FIFO | 64 | full | FIFO | 64 | full | FIFO |
| 111 | 32 | 2 | FIFO | 32 | 2 | FIFO | 32 | 1 | FIFO | 32 | 1 | FIFO |
| 112 | 64 | 2 | FIFO | 64 | 2 | FIFO | 64 | 1 | FIFO | 64 | 1 | FIFO |
| 113 | 32 | 2 | FIFO | 32 | 2 | FIFO | 32 | 2 | FIFO | 32 | 2 | FIFO |
| 114 | 64 | 2 | FIFO | 64 | 2 | FIFO | 64 | 2 | FIFO | 64 | 2 | FIFO |
| 115 | 32 | 2 | FIFO | 32 | 2 | FIFO | 32 | 4 | FIFO | 32 | 4 | FIFO |
| 116 | 64 | 2 | FIFO | 64 | 2 | FIFO | 64 | 4 | FIFO | 64 | 4 | FIFO |
| 117 | 32 | 2 | FIFO | 32 | 2 | FIFO | 32 | 8 | FIFO | 32 | 8 | FIFO |
| 118 | 64 | 2 | FIFO | 64 | 2 | FIFO | 64 | 8 | FIFO | 64 | 8 | FIFO |
| 119 | 32 | 2 | FIFO | 32 | 2 | FIFO | 32 | full | FIFO | 32 | full | FIFO |
| 120 | 64 | 2 | FIFO | 64 | 2 | FIFO | 64 | full | FIFO | 64 | full | FIFO |
| 121 | 32 | 4 | FIFO | 32 | 4 | FIFO | 32 | 1 | FIFO | 32 | 1 | FIFO |
| 122 | 64 | 4 | FIFO | 64 | 4 | FIFO | 64 | 1 | FIFO | 64 | 1 | FIFO |
| 123 | 32 | 4 | FIFO | 32 | 4 | FIFO | 32 | 2 | FIFO | 32 | 2 | FIFO |
| 124 | 64 | 4 | FIFO | 64 | 4 | FIFO | 64 | 2 | FIFO | 64 | 2 | FIFO |
| 125 | 32 | 4 | FIFO | 32 | 4 | FIFO | 32 | 4 | FIFO | 32 | 4 | FIFO |
| 126 | 64 | 4 | FIFO | 64 | 4 | FIFO | 64 | 4 | FIFO | 64 | 4 | FIFO |
| 127 | 32 | 4 | FIFO | 32 | 4 | FIFO | 32 | 8 | FIFO | 32 | 8 | FIFO |
| 128 | 64 | 4 | FIFO | 64 | 4 | FIFO | 64 | 8 | FIFO | 64 | 8 | FIFO |
| 129 | 32 | 4 | FIFO | 32 | 4 | FIFO | 32 | full | FIFO | 32 | full | FIFO |
| 130 | 64 | 4 | FIFO | 64 | 4 | FIFO | 64 | full | FIFO | 64 | full | FIFO |
| 131 | 32 | 8 | FIFO | 32 | 8 | FIFO | 32 | 1 | FIFO | 32 | 1 | FIFO |
| 132 | 64 | 8 | FIFO | 64 | 8 | FIFO | 64 | 1 | FIFO | 64 | 1 | FIFO |
| 133 | 32 | 8 | FIFO | 32 | 8 | FIFO | 32 | 2 | FIFO | 32 | 2 | FIFO |
| 134 | 64 | 8 | FIFO | 64 | 8 | FIFO | 64 | 2 | FIFO | 64 | 2 | FIFO |
| 135 | 32 | 8 | FIFO | 32 | 8 | FIFO | 32 | 4 | FIFO | 32 | 4 | FIFO |
| 136 | 64 | 8 | FIFO | 64 | 8 | FIFO | 64 | 4 | FIFO | 64 | 4 | FIFO |
| 137 | 32 | 8 | FIFO | 32 | 8 | FIFO | 32 | 8 | FIFO | 32 | 8 | FIFO |
| 138 | 64 | 8 | FIFO | 64 | 8 | FIFO | 64 | 8 | FIFO | 64 | 8 | FIFO |
| 139 | 32 | 8 | FIFO | 32 | 8 | FIFO | 32 | full | FIFO | 32 | full | FIFO |
| 140 | 64 | 8 | FIFO | 64 | 8 | FIFO | 64 | full | FIFO | 64 | full | FIFO |
| 141 | 32 | full | FIFO | 32 | full | FIFO | 32 | 1 | FIFO | 32 | 1 | FIFO |
| 142 | 64 | full | FIFO | 64 | full | FIFO | 64 | 1 | FIFO | 64 | 1 | FIFO |
| 143 | 32 | full | FIFO | 32 | full | FIFO | 32 | 2 | FIFO | 32 | 2 | FIFO |
| 144 | 64 | full | FIFO | 64 | full | FIFO | 64 | 2 | FIFO | 64 | 2 | FIFO |
| 145 | 32 | full | FIFO | 32 | full | FIFO | 32 | 4 | FIFO | 32 | 4 | FIFO |
| 146 | 64 | full | FIFO | 64 | full | FIFO | 64 | 4 | FIFO | 64 | 4 | FIFO |
| 147 | 32 | full | FIFO | 32 | full | FIFO | 32 | 8 | FIFO | 32 | 8 | FIFO |
| 148 | 64 | full | FIFO | 64 | full | FIFO | 64 | 8 | FIFO | 64 | 8 | FIFO |
| 149 | 32 | full | FIFO | 32 | full | FIFO | 32 | full | FIFO | 32 | full | FIFO |
| 150 | 64 | full | FIFO | 64 | full | FIFO | 64 | full | FIFO | 64 | full | FIFO |

The plots for each of the benchmarks are shown in the next section.

**GCC Benchmark :**

**L1 Separate; L2 Unified**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | UL2 Associativity | Replacement policy | CPI |
| 64 | Full | Full | 8 | LRU | 1.04656503 |
| 32 | Full | Full | Full | LRU | 1.0636472 |
| **64** | **Full** | **Full** | **Full** | **LRU** | **1.04198861** |
| 32 | 1 | 1 | 1 | FIFO | 1.17201767 |
| 64 | 1 | 1 | 1 | FIFO | 1.1250448 |

**L1 Unified; L2 Unified**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as

shown below :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Block Size | UL1 Associativity | UL2 Associativity | Replacement policy | CPI |
| 64 | Full | 8 | L | 1.041287628 |
| 32 | Full | Full | L | 1.056013642 |
| **64** | **Full** | **Full** | **L** | **1.036672268** |
| 32 | 1 | 1 | FIFO | 1.190592389 |
| 64 | 1 | 1 | FIFO | 1.145796657 |

**L1 Separate; L2 Separate**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | DL2 Associativity | IL2 Associativity | Replacement policy | CPI |
| 64 | Full | Full | 8 | 8 | LRU | 1.048158129 |
| 32 | Full | Full | Full | Full | LRU | 1.061719301 |
| **64** | **Full** | **Full** | **Full** | **Full** | **LRU** | **1.040812957** |
| 32 | 1 | 1 | 1 | 1 | FIFO | 1.205204367 |
| 64 | 1 | 1 | 1 | 1 | FIFO | 1.151150287 |

**Choice of configuration:**

According to the data above, the lowest CPI is achieved for the configuration shown below:

Unified L1 and Unified L2

**L1 Block Size = 64, L2 Block Size = 64, L1 fully associative, L2 fully associative, LRU replacement policy for both levels**

Minimum **CPI = 1.036672268**

**Anagram Benchmark:**

**L1 Separate; L2 Unified**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | UL2 Associativity | Replacement policy | CPI |
| 64 | 8 | 8 | Full | LRU | 1.075945363 |
| 32 | Full | Full | 1 | LRU | 1.112400056 |
| **64** | **Full** | **Full** | **1** | **LRU** | **1.056552935** |
| 32 | Full | Full | 2 | LRU | 1.124545035 |
| 64 | Full | Full | 2 | LRU | 1.062619732 |

**L1 Unified; L2 Unified**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Block Size | UL1 Associativity | UL2 Associativity | Replacement policy | CPI |
| 64 | 1 | Full | LRU | 1.080038 |
| 32 | 2 | 1 | LRU | 1.11346 |
| **64** | **2** | **1** | **LRU** | **1.057471** |
| 32 | 2 | 2 | LRU | 1.124925 |
| 64 | 2 | 2 | LRU | 1.063124 |

**L1 Separate; L2 Separate**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | DL2 Associativity | IL2 Associativity | Replacement policy | CPI |
| 64 | 2 | 2 | 4 | 4 | LRU | 1.075953718 |
| 32 | 2 | 2 | 8 | 8 | LRU | 1.151218278 |
| **64** | **2** | **2** | **8** | **8** | **LRU** | **1.075939423** |
| 32 | 2 | 2 | Full | Full | LRU | 1.151218278 |
| 64 | 2 | 2 | Full | Full | LRU | 1.075939423 |

**Choice of configuration :**

According to the data above, the lowest CPI is achieved for the configuration shown below:

Separate L1 and unified L2

**L1 Instruction cache Block Size = 64, L1 Data cache Block Size =64, L2 Block Size = 64, L1 Data cache fully associative, L1 instruction cache fully associative, L2 fully associative, LRU replacement policy for both levels**

Minimum **CPI = 1.056552935**

**Go Benchmark**

**L1 Separate; L2 Unified**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | UL2 Associativity | Replacement policy | CPI |
| 64 | Full | Full | 2 | Random | 1.010746759 |
| 32 | Full | Full | 4 | Random | 1.018992127 |
| **64** | **Full** | **Full** | **4** | **Random** | **1.010628669** |
| 32 | Full | Full | 8 | Random | 1.019093373 |
| 64 | Full | Full | 8 | Random | 1.01067135 |

**L1 Unified; L2 Unified**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Block Size | UL1 Associativity | UL2 Associativity | Replacement policy | CPI |
| 64 | 4 | 2 | LRU | 1.009184253 |
| 32 | 4 | 4 | LRU | 1.015156897 |
| **64** | **4** | **4** | **LRU** | **1.009092051** |
| 32 | 4 | 8 | LRU | 1.015926553 |
| 64 | 4 | 8 | LRU | 1.00948391 |

**L1 Separate; L2 Separate**

CPI vs Configuration number:

The Snapshot of the configurations around the optimum case for the current design scenario is as shown below :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | DL2 Associativity | IL2 Associativity | Replacement policy | CPI |
| 64 | Full | Full | 1 | 1 | Random | 1.075953718 |
| 32 | Full | Full | 2 | 2 | Random | 1.151218278 |
| **64** | **Full** | **Full** | **2** | **2** | **Random** | **1.075939423** |
| 32 | Full | Full | 4 | 4 | Random | 1.151218278 |
| 64 | Full | Full | 4 | 4 | Random | 1.075939423 |

**Choice of configuration :**

According to the data above, the lowest CPI is achieved for the configuration shown below:

Unified L1 and unified L2

**L1 Block Size = 64, L2 Block Size = 64, L1 fully associative, L2 fully associative, LRU replacement policy for both levels**

Minimum **CPI = 1.010628669**

The Following Table gives the Optimum Cache Configurations for each of the benchmarks considered :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Benchmark** | **Configuration** | | | **CPI** |
| **GCC** |  | **UL1** | | 1.036672268 |
| Sets | 1 | |
| Block Size | 64 | |
| Associativity | Full | |
| Replacement Policy | LRU | |
|  | **UL2** | |
| Sets | 1 | |
| Block Size | 64 | |
| Associativity | Full | |
| Replacement Policy | LRU | |
| **Anagram** |  | **IL1** | **DL1** | 1.057471094 |
| Sets | 1 | 1 |
| Block Size | 64 | 64 |
| Associativity | Full | Full |
| Replacement Policy | LRU | LRU |
|  | **UL2** | |
| Sets | 2048 | |
| Block Size | 64 | |
| Associativity | 1 | |
| Replacement Policy | LRU | |
| **GO** |  | **UL1** | | 1.009184253 |
| Sets | 512 | |
| Block Size | 64 | |
| Associativity | 4 | |
| Replacement Policy | LRU | |
|  | **UL2** | |
| Sets | 512 | |
| Block Size | 64 | |
| Associativity | 2 | |
| Replacement Policy | LRU | |

1. **Part 4: Define Cost Function**

While defining the cost function we have developed few references. The reference cache with its parameters are as follows:

* L2 cache size of 1KB
* Direct-Mapped Associativity
* Block Size of 4 Bytes
* Random Replacement Policy

Here, Block size is assumed to affect the cost because, throughout our tests, we have assumed the miss penalty to be constant irrespective of the block size. But as the block size increases, to keep the miss penalty constant we would need additional hardware as more bytes of data need to be moved in the same time.

While designing a cache, we have considered the cost of split or unified cache separately for each cache level. When the cache is split, the size of the cache may reduce, but the memory bandwidth of the processor increases as additional hardware must be employed to fetch from the instruction cache and the data cache separately. Along with this, since L1 cache is faster than L2 cache, the cost factor for L1 split cache is higher than the L2 split cache in order to support faster access.

The cost of our cache design has been calculated with respect to the reference cache design mentioned above. The cost of any cache was assumed to be expressed in terms of the cost of reference cache as follows:

**CostOfAnyCache/CostOfReferenceCache = CostFactor**

**CostOfAnyCache = CostFactor \* CostOfReferenceCache**

As we have considered the cost of reference cache to be the basic unit, its value is taken to be equal to 1 and therefore CostOfReferenceCache becomes our unit.

Thus, **CostOfAnyCache = CostFactor**

Where CostFactor is a ratio determined by the different parameters of our cache. CostFactor is given as follows:

**CostFactor = CacheSizeRatio\*CacheSizeCostFactor**

**+ BlockSizeRatio\*BlockSizeCostFactor**

**+ AssociativityCostFactor(Associativity)**

**+ ReplacementCostFactor(ReplacementPolicy)**

**+ L1splitCostFactor(Split/Unified) + L2splitCostFactor(Split/Unified)**

Here:

**CacheSizeRatio** = Size of present cache / Size of reference cache

**BlockSizeRatio** = Block size of present cache / Block Size of reference cache

**CacheSizeCostFactor** is a scaling factor which determines the impact of the cache size on the cost.

**BlockSizeCostFactor** is a scaling factor which determines the impact of the block size on the cost.

**AssociativityCostFactor** is a factor which varies according to the associativity

**ReplacementCostFactor** is a factor which varies according to the replacement policy used.

**L1splitCostFactor** is a factor which varies according to the cache kind used that is either split or unified.

**L2splitCostFactor** is a factor which varies according to the cache kind used that is either split or unified.

Furthermore, this reference cache is assumed to be at the L2 level.

The values of these factors have been chosen such that if we substitute the parameters of the reference cache in the equation for CostFactor, we get a value of 1. The values we have chosen are as follows:

CacheSizeCostFactor = 0.5 for L2 cache

= 0.5\*5 for L1 cache (since L1 caches are more expensive here we have assumed that an L1 cache is 5 times as expensive as an L2 cache)

BlockSizeCostFactor = 0.1

AssociativityCostFactor(1-way) = 0.35

AssociativityCostFactor(2-way) = 0.55

AssociativityCostFactor(4-way) = 0.75

AssociativityCostFactor(8-way) = 0.95

AssociativityCostFactor(Fully associative) = 1.15

ReplacementCostFactor(Random) = 0.15

ReplacementCostFactor(FIFO) = 0.25

ReplacementCostFactor(LRU) = 0.45

L1splitCostFactor(Split) = 0.4

L1splitCostFactor(Unified) = 0

L2splitCostFactor(Split) = 0.2

L2splitCostFactor(Unified) = 0

From the equation above we have said that

**Cost Of Cache = CostFactor** in reference units

Thus **Cost Of Cache = CacheSizeRatio\*CacheSizeCostFactor + BlockSizeRatio\*BlockSizeCostFactor + AssociativityCostFactor(Associativity) + ReplacementCostFactor(ReplacementPolicy) + L1splitCostFactor(Split/Unified) + L2splitCostFactor(Split/Unified)**

For a 2 level cache, we calculate the cost of each cache level separately and just add them up.

We arrived at the cost effective optimum configuration of cache by calculating the product of CPI and Cost Function for each design.

1. **Part 5 : Optimize caches for performance/cost**

To achieve optimization for both performance and cost, we calculate the product of CPI and Cost for each of the configurations. The optimal configuration for a particular benchmark is the one which has the least CPI-Cost product.

**GCC Benchmark :**

**L1 Separate; L2 Unified**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | UL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | Full | Full | 8 | LRU | 1.046565025 | 838.8 | 877.8587433 |
| 32 | Full | Full | Full | LRU | 1.063647198 | 837.8 | 891.1236222 |
| **64** | **Full** | **Full** | **Full** | **LRU** | **1.041988607** | **839** | **874.2284409** |
| 32 | 1 | 1 | 1 | FIFO | 1.172017666 | 836 | 979.8067691 |
| 64 | 1 | 1 | 1 | FIFO | 1.125044796 | 837.2 | 941.8875033 |

**L1 Unified; L2 Unified**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Block Size | UL1 Associativity | UL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | Full | 8 | LRU | 1.041287628 | 836.2 | 870.7247149 |
| 32 | Full | Full | LRU | 1.056013642 | 835.6 | 882.4049993 |
| **64** | **Full** | **Full** | **LRU** | **1.036672268** | **836.4** | **867.0726851** |
| 32 | 1 | 1 | FIFO | 1.190592389 | 834.4 | 993.4302896 |
| 64 | 1 | 1 | FIFO | 1.145796657 | 835.2 | 956.9693678 |

**L1 Separate; L2 Separate**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | DL2 Associativity | IL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | Full | Full | 8 | 8 | LRU | 1.048158129 | 841 | 881.5009866 |
| 32 | Full | Full | Full | Full | LRU | 1.061719301 | 839.8 | 891.631869 |
| **64** | **Full** | **Full** | **Full** | **Full** | **LRU** | **1.040812957** | **841.4** | **875.7400221** |
| 32 | 1 | 1 | 1 | 1 | FIFO | 1.205204367 | 837.4 | 1009.238137 |
| 64 | 1 | 1 | 1 | 1 | FIFO | 1.151150287 | 839 | 965.8150909 |

**Optimal configuration for CPI\*Cost :**

The optimal configuration for this benchmark is:

L1 Unified, L2 Unified

L1 fully associative, L2 fully associative, LRU replacement policy at both L1 and L2

**Anagram Benchmark :**

**L1 Separate; L2 Unified**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | UL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | 1 | 1 | 8 | Random | 1.077211841 | 837.1 | 901.7340325 |
| 32 | 2 | 2 | Full | Random | 1.112769561 | 835.5 | 929.718968 |
| **64** | **2** | **2** | **Full** | **Random** | **1.056883279** | **836.7** | **884.2942399** |
| 32 | 2 | 2 | 1 | Random | 1.125010107 | 835.7 | 940.170946 |
| 64 | 2 | 2 | 1 | Random | 1.06295866 | 836.9 | 889.5901029 |

**L1 Unified; L2 Unified**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Block Size | UL1 Associativity | UL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | 1 | 8 | LRU | 1.080038067 | 835.6 | 902.4798091 |
| 32 | 2 | Full | LRU | 1.113459967 | 834.2 | 928.8483046 |
| **64** | **2** | **Full** | **LRU** | **1.057471094** | **835** | **882.9883631** |
| 32 | 2 | 1 | LRU | 1.124925395 | 834.4 | 938.6377493 |
| 64 | 2 | 1 | LRU | 1.063124094 | 835.2 | 887.9212435 |

**L1 Separate; L2 Separate**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | DL2 Associativity | IL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | 1 | 1 | Full | Full | Random | 1.093251493 | 839.4 | 917.6753033 |
| 32 | 2 | 2 | 1 | 1 | Random | 1.151917639 | 836.6 | 963.6942964 |
| **64** | **2** | **2** | **1** | **1** | **Random** | **1.076544112** | **838.2** | **902.3592745** |
| 32 | 2 | 2 | 2 | 2 | Random | 1.15142883 | 837 | 963.7459311 |
| 64 | 2 | 2 | 2 | 2 | Random | 1.076249811 | 838.6 | 902.5430915 |

**Optimal configuration for CPI\*Cost :**

The optimal configuration for this benchmark is:

L1 Unified, L2 Unified

L1 2-way set associative, L2 Direct mapped, LRU replacement policy at both L1 and L2

**Go Benchmark :**

**L1 Separate; L2 Unified**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | UL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | 4 | 4 | 1 | Random | 1.012976533 | 837.1 | 847.9626557 |
| 32 | 4 | 4 | 2 | Random | 1.018776019 | 836.1 | 851.7986298 |
| **64** | **4** | **4** | **2** | **Random** | **1.010971991** | **837.3** | **846.4868477** |
| 32 | 4 | 4 | 4 | Random | 1.018674475 | 836.3 | 851.9174635 |
| 64 | 4 | 4 | 4 | Random | 1.010830325 | 837.5 | 846.5703974 |

**L1 Unified; L2 Unified**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Block Size | UL1 Associativity | UL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | 4 | 1 | LRU | 1.015252308 | 835.2 | 847.9387273 |
| 32 | 4 | 2 | LRU | 1.015083596 | 834.6 | 847.1887696 |
| **64** | **4** | **2** | **LRU** | **1.009184253** | **835.4** | **843.0725253** |
| 32 | 4 | 4 | LRU | 1.015156897 | 834.8 | 847.4529776 |
| 64 | 4 | 4 | LRU | 1.009092051 | 835.6 | 843.1973181 |

**L1 Separate; L2 Separate**

CPI\*Cost vs Configuration number:

The Snapshot of the configurations around the optimum case for the product CPI\*Cost is as shown below :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Block Size | DL1 Associativity | IL1 Associativity | DL2 Associativity | IL2 Associativity | Replacement policy | CPI | Cost | CPI\*Cost |
| 64 | 4 | 4 | 1 | 1 | Random | 1.01195082 | 838.6 | 848.6219577 |
| 32 | 4 | 4 | 2 | 2 | Random | 1.019574206 | 837.4 | 853.7914404 |
| **64** | **4** | **4** | **2** | **2** | **Random** | **1.01129593** | **839** | **848.4772852** |
| 32 | 4 | 4 | 4 | 4 | Random | 1.019634485 | 837.8 | 854.2497717 |
| 64 | 4 | 4 | 4 | 4 | Random | 1.011291196 | 839.4 | 848.8778301 |

**Optimal configuration for CPI\*Cost:**

The optimal configuration for this benchmark is:

L1 Unified, L2 Unified

L1 4-way set associative, L2 2-way set associative, LRU replacement policy at both L1 and L2

The following table shows the Cache Configuration for the minimum product CPI\*Cost for each of the benchmarks:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Benchmark** | **Configuration** | | | **CPI** | **Cost** | **CPI\*Cost** |
| **GCC** |  | **UL1** | | 1.036672268 | 836.4 | 867.0726851 |
| Sets | 1 | |
| Block Size | 64 | |
| Associativity | Full | |
| Replacement Policy | LRU | |
|  | **UL2** | |
| Sets | 1 | |
| Block Size | 64 | |
| Associativity | Full | |
| Replacement Policy | LRU | |
| **Anagram** |  | **UL1** | | 1.057471094 | 835 | 882.9883631 |
| Sets | 1024 | |
| Block Size | 64 | |
| Associativity | 2 | |
| Replacement Policy | LRU | |
|  | **UL2** | |
| Sets | 2048 | |
| Block Size | 64 | |
| Associativity | 1 | |
| Replacement Policy | LRU | |
| **GO** |  | | **UL1** | 1.009184253 | 835.4 | 843.0725253 |
| Sets | | 512 |
| Block Size | | 64 |
| Associativity | | 4 |
| Replacement Policy | | LRU |
|  | | **UL2** |
| Sets | | 1024 |
| Block Size | | 64 |
| Associativity | | 2 |
| Replacement Policy | | LRU |

**Optimum configuration for all benchmarks considering CPI\*Cost**

The above table indicates that all the benchmarks need the 2 cache levels to be unified. Therefore for the optimum configuration for the all the benchmarks taken together would be equal to the one which has the lowest product of CPI\*Cost. The CPI\*Cost is the least for the GO benchmark for the following configuration:

|  |  |
| --- | --- |
|  | **UL1** |
| Sets | 512 |
| Block Size | 64 |
| Associativity | 4 |
| Replacement Policy | LRU |
|  | **UL2** |
| Sets | 1024 |
| Block Size | 64 |
| Associativity | 2 |
| Replacement Policy | LRU |

**Optimum configuration for all benchmarks considering only CPI**

Among all the benchmarks, the CPI is minimum for the GO benchmark for the following configuration:

L1 Unified and L2 unified with the following details:

|  |  |
| --- | --- |
|  | **UL1** |
| Sets | 1 |
| Block Size | 64 |
| Associativity | Full |
| Replacement Policy | LRU |
|  | **UL2** |
| Sets | 1 |
| Block Size | 64 |
| Associativity | Full |
| Replacement Policy | LRU |

1. **Appendix: Scripts**
2. **TCL Script to generate different cache configurations for Anagram Benchmark for L1 Separate and L2 separate.**

This script generates all the 150 configurations required for a particular benchmark and executes the simplescalar tool by using the script given in the next part.

#!/usr/bin/tclsh

if { [catch exec rm /home/004/a/ak/aks130730/SimpleScalar/simplesim-3.0/results/result\_SS\_ANA] } {}

set params\_separate\_separate [list]

#set Block\_Size 64

puts "L1 separate and L2 separate"

#set L1\_size {{65536 65536} {32768 98304} {98304 32768}}

set L1\_size [expr 64\*1024]

#set L2\_size {{524288 524288} {262144 786432} {786432 262144}}

set L2\_size [expr 512\*1024]

set i 0

foreach repla\_policy {r l f} {

foreach L1\_associativity {1 2 4 8 FA} {

foreach L2\_associativity {1 2 4 8 FA} {

foreach Block\_Size {32 64} {

if { $L1\_associativity == "FA" && $L2\_associativity != "FA" } {

set L1\_ins\_sets 1

set L1\_data\_sets 1

set L1\_associativity\_temp [expr ($L1\_size/$Block\_Size)]

set L2\_associativity\_temp $L2\_associativity

set L2\_ins\_sets [expr ($L2\_size/($Block\_Size\*$L2\_associativity\_temp))]

set L2\_data\_sets [expr ($L2\_size/($Block\_Size\*$L2\_associativity\_temp))]

} elseif { $L2\_associativity == "FA" && $L1\_associativity != "FA" } {

set L2\_ins\_sets 1

set L2\_data\_sets 1

set L2\_associativity\_temp [expr ($L2\_size/$Block\_Size)]

set L1\_associativity\_temp $L1\_associativity

set L1\_ins\_sets [expr ($L1\_size/($Block\_Size\*$L1\_associativity\_temp))]

set L1\_data\_sets [expr ($L1\_size/($Block\_Size\*$L1\_associativity\_temp))]

} elseif { $L1\_associativity == "FA" && $L2\_associativity == "FA" } {

set L1\_ins\_sets 1

set L1\_data\_sets 1

set L2\_ins\_sets 1

set L2\_data\_sets 1

set L1\_associativity\_temp [expr ($L1\_size/$Block\_Size)]

set L2\_associativity\_temp [expr ($L2\_size/$Block\_Size)]

} else {

set L1\_associativity\_temp $L1\_associativity

set L2\_associativity\_temp $L2\_associativity

set L1\_ins\_sets [expr ($L1\_size/($Block\_Size\*$L1\_associativity\_temp))]

set L1\_data\_sets [expr ($L1\_size/($Block\_Size\*$L1\_associativity\_temp))]

set L2\_ins\_sets [expr ($L2\_size/($Block\_Size\*$L2\_associativity\_temp))]

set L2\_data\_sets [expr ($L2\_size/($Block\_Size\*$L2\_associativity\_temp))]

}

lappend params\_separate\_separate "$L1\_ins\_sets $Block\_Size $L1\_associativity\_temp $repla\_policy $L1\_data\_sets $Block\_Size $L1\_associativity\_temp $repla\_policy $L2\_ins\_sets $Block\_Size $L2\_associativity\_temp $repla\_policy $L2\_data\_sets $Block\_Size $L2\_associativity\_temp $repla\_policy"

puts "$L1\_ins\_sets $Block\_Size $L1\_associativity\_temp $repla\_policy $L1\_data\_sets $Block\_Size $L1\_associativity\_temp $repla\_policy $L2\_ins\_sets $Block\_Size $L2\_associativity\_temp $repla\_policy $L2\_data\_sets $Block\_Size $L2\_associativity\_temp $repla\_policy"

incr i

puts $i

}

}

}

}

foreach param $params\_separate\_separate {

puts "Now running for [lindex $param 0] [lindex $param 1] [lindex $param 2] [lindex $param 3] [lindex $param 4] [lindex $param 5] [lindex $param 6] [lindex $param 7] [lindex $param 8] [lindex $param 9] [lindex $param 10] [lindex $param 11] [lindex $param 12] [lindex $param 13] [lindex $param 14] [lindex $param 15]"

exec /home/004/a/ak/aks130730/SimpleScalar/simplesim-3.0/part3.cpi\_SS\_ANA [lindex $param 0] [lindex $param 1] [lindex $param 2] [lindex $param 3] [lindex $param 4] [lindex $param 5] [lindex $param 6] [lindex $param 7] [lindex $param 8] [lindex $param 9] [lindex $param 10] [lindex $param 11] [lindex $param 12] [lindex $param 13] [lindex $param 14] [lindex $param 15]

}

6 scripts in total, similar to this were used were used for each benchmark and each of the cache configurations: L1 Separate, L2 Unified and L1 Unified and L2 Unified cases

1. **Shell script to run simplescalar tool**

This script runs the simplescalar tool for a given configuration, it is called by the previous script for each configuration.

#!/bin/ksh

echo 'date'

./sim-cache -cache:il1 il1:$1:$2:$3:$4 -cache:dl1 dl1:$5:$6:$7:$8 -cache:il2 il2:$9:${10}:${11}:${12} -cache:dl2 dl2:${13}:${14}:${15}:${16} -tlb:itlb none -tlb:dtlb none -redir:sim ./results/data.out\_ANA\_SS ./benchmarks/anagram.alpha ./benchmarks/words < ./benchmarks/anagram.in

#./sim-cache -cache:dl1 dl1:512:64:2:l -cache:il1 il1:512:64:2:l -cache:il2 dl2 -cache:dl2 dl2:16384:64:1:l -tlb:itlb none -tlb:dtlb none ./benchmarks/cc1.alpha -O ./benchmarks/1stmt.i

#./sim-cache -cache:dl1 dl1:512:64:2:l -cache:il1 il1:512:64:2:l -cache:il2 dl2 -cache:dl2 dl2:16384:64:1:l -tlb:itlb none -tlb:dtlb none ./benchmarks/cc1.alpha -O ./benchmarks/1stmt.i

#./sim-cache -cache:dl1 dl1:$1:$2:$3:l -cache:il1 il1:$4:$5:$6:l -cache:il2 dl2 -cache:dl2 dl2:$7:$8:$9:l -tlb:itlb none -tlb:dtlb none -redir:sim ./results/data.out\_ANA\_SS ./benchmarks/cc1.alpha -O ./benchmarks/1stmt.i

echo "il1 : Sets = $1; Block Size = $2; Associativity = $3; Replacement policy = $4" >> ./results/result\_ANA\_SS

echo "dl1 : Sets = $5; Block Size = $6; Associativity = $7; Replacement policy = $8" >> ./results/result\_ANA\_SS

echo "il2 : Sets = $9; Block Size = ${10}; Associativity = ${11}; Replacement policy = ${12}" >> ./results/result\_ANA\_SS

echo "dl1 : Sets = ${13}; Block Size = ${14}; Associativity = ${15}; Replacement policy = ${16}" >> ./results/result\_ANA\_SS

grep sim\_num\_insn ./results/data.out\_ANA\_SS >> ./results/result\_ANA\_SS

grep miss\_rate ./results/data.out\_ANA\_SS >> ./results/result\_ANA\_SS

grep il1.accesses ./results/data.out\_ANA\_SS >> ./results/result\_ANA\_SS

grep dl1.accesses ./results/data.out\_ANA\_SS >> ./results/result\_ANA\_SS

grep il2.accesses ./results/data.out\_ANA\_SS >> ./results/result\_ANA\_SS

grep dl2.accesses ./results/data.out\_ANA\_SS >> ./results/result\_ANA\_SS

rm -rf ./results/data.out\_ANA\_SS

echo "----------------------------------------------------------------------" >> ./results/result\_ANA\_SS

6 scripts in total, similar to this were used were used for each benchmark and each of the cache configurations: L1 Separate, L2 Unified and L1 Unified and L2 Unified cases

1. **Python script to format the data generated by the above scripts**

This script reads the data generated from the above scripts and extracts the required values(Block size, associativity etc.), calculates the CPI and CPI\*Cost for each configuration and then copies the data to a .csv file, using which further analysis on the data was done in MS excel.

import csv

in\_file = open('../results\_for\_read/result\_ANA\_SS', 'r')

in\_file\_name = in\_file.name

DataFileName = 'Data\_file\_' + in\_file\_name[27:] + '.csv'

res\_file = open('./CPI\_results', 'w')

File\_data = open(DataFileName, 'w')

Ref\_size = 1024

DL1\_size = 65536

IL1\_size = 65536

IDL1\_size = 131072

DL2\_size = 524288

IL2\_size = 524288

IDL2\_size = 1048576

MPL1 = 5

MPL2 = 40

CostAssociativity = {'1':0.35 , '2':0.55, '4':0.75 , '8':0.95 , 'f':1.15 }

CostReplacement = {'r':0.15 , 'l':0.25 , 'f':0.45}

Il1\_miss\_rate = 'NA'

Dl1\_miss\_rate = 'NA'

Il2\_miss\_rate = 'NA'

Dl2\_miss\_rate = 'NA'

L1unified = 0

L2unified = 0

#DL1\_AccessPerInst = 0.36790047030

#DL2\_AccessPerInst = 0.012067

#MemAccessPerInst = 0.36790047030 #for gcc only fr now

DL1FoundAlready = 0

for line in in\_file:

if 'Sets' in line:

print >> res\_file, line

cache\_params = line.split(' ')

if 'dl1' in line:

if DL1FoundAlready == 0:

DL1FoundAlready = 1

if 'il1' in line:

#L1 unified

IDl1\_block\_size = cache\_params[8]

IDl1\_block\_size = IDl1\_block\_size[:-1]

IDl1\_block\_size = int(IDl1\_block\_size)

IDL1\_Associativity = cache\_params[11]

IDL1\_Associativity = IDL1\_Associativity[:-1]

#print line

#print IDL1\_Associativity

IDL1\_Repl = cache\_params[15]

IDL1\_Repl = IDL1\_Repl[:-1]

#print IDl1\_block\_size

else:

Dl1\_block\_size = cache\_params[8]

Dl1\_block\_size = Dl1\_block\_size[:-1]

Dl1\_block\_size = int(Dl1\_block\_size)

DL1\_Associativity = cache\_params[11]

DL1\_Associativity = DL1\_Associativity[:-1]

DL1\_Repl = cache\_params[15]

DL1\_Repl = DL1\_Repl[:-1]

print Dl1\_block\_size

else:

DL1FoundAlready = 0

Dl2\_block\_size = cache\_params[8]

Dl2\_block\_size = Dl2\_block\_size[:-1]

Dl2\_block\_size = int(Dl2\_block\_size)

DL2\_Associativity = cache\_params[11]

DL2\_Associativity = DL2\_Associativity[:-1]

DL2\_Repl = cache\_params[15]

DL2\_Repl = DL2\_Repl[:-1]

elif 'il1' in line:

Il1\_block\_size = cache\_params[8]

Il1\_block\_size = Il1\_block\_size[:-1]

Il1\_block\_size = int(Il1\_block\_size)

IL1\_Associativity = cache\_params[11]

IL1\_Associativity = IL1\_Associativity[:-1]

IL1\_Repl = cache\_params[15]

IL1\_Repl = IL1\_Repl[:-1]

#print Il1\_block\_size

elif 'dl2' in line:

if 'il2' in line:

IDl2\_block\_size = cache\_params[8]

IDl2\_block\_size = IDl2\_block\_size[:-1]

IDl2\_block\_size = int(IDl2\_block\_size)

IDL2\_Associativity = cache\_params[11]

IDL2\_Associativity = IDL2\_Associativity[:-1]

IDL2\_Repl = cache\_params[15]

IDL2\_Repl = IDL2\_Repl[:-1]

#print IDl2\_block\_size

else:

Dl2\_block\_size = cache\_params[8]

Dl2\_block\_size = Dl2\_block\_size[:-1]

Dl2\_block\_size = int(Dl2\_block\_size)

DL2\_Associativity = cache\_params[11]

DL2\_Associativity = DL2\_Associativity[:-1]

DL2\_Repl = cache\_params[15]

DL2\_Repl = DL2\_Repl[:-1]

#print Dl2\_block\_size

elif 'il2' in line:

Il2\_block\_size = cache\_params[8]

Il2\_block\_size = Il2\_block\_size[:-1]

Il2\_block\_size = int(Il2\_block\_size)

IL2\_Associativity = cache\_params[11]

IL2\_Associativity = IL2\_Associativity[:-1]

IL2\_Repl = cache\_params[15]

IL2\_Repl = IL2\_Repl[:-1]

#print Il2\_block\_size

if '.miss\_rate' in line:

Line\_entries = line.split()

if Line\_entries[0] == 'il1.miss\_rate':

Il1\_miss\_rate = float(Line\_entries[1])

#print Il1\_miss\_rate

elif Line\_entries[0] == 'dl1.miss\_rate':

Dl1\_miss\_rate = float(Line\_entries[1])

elif Line\_entries[0] == 'il2.miss\_rate':

Il2\_miss\_rate = float(Line\_entries[1])

elif Line\_entries[0] == 'dl2.miss\_rate':

Dl2\_miss\_rate = float(Line\_entries[1])

elif 'sim\_num\_insn' in line:

Line\_entries = line.split()

Num\_of\_insts = float(Line\_entries[1])

elif 'dl1.accesses' in line:

Line\_entries = line.split()

Num\_of\_memaccesses = float(Line\_entries[1])

DL1\_AccessPerInst = Num\_of\_memaccesses/Num\_of\_insts

# print "MemAccessPerInst=", MemAccessPerInst

elif 'dl2.accesses' in line:

Line\_entries = line.split()

Num\_of\_Dl2Accesses = float(Line\_entries[1])

DL2\_AccessPerInst = Num\_of\_Dl2Accesses/Num\_of\_insts

elif 'il2.accesses' in line:

Line\_entries = line.split()

Num\_of\_Il2Accesses = float(Line\_entries[1])

Il2\_AccessesPerInst = Num\_of\_Il2Accesses/Num\_of\_insts

#last\_pos = in\_file.tell()

#print Il1\_miss\_rate,Dl1\_miss\_rate,Il2\_miss\_rate,Dl2\_miss\_rate

if '---------' in line: #All values read

DL1FoundAlready = 0

if Il1\_miss\_rate == 'NA': #L1 unified

Il1\_miss\_rate = Dl1\_miss\_rate

L1unified = 1

elif Dl1\_miss\_rate == 'NA': #L1 unified

Dl1\_miss\_rate = Il1\_miss\_rate

L1unified = 1

if Il2\_miss\_rate == 'NA': #L2 unified

Il2\_miss\_rate = Dl2\_miss\_rate

L2unified = 1

elif Dl2\_miss\_rate == 'NA': #L2 unified

Dl2\_miss\_rate = Il2\_miss\_rate

L2unified = 1

#print '-----------------'

#print Il1\_miss\_rate,Dl1\_miss\_rate,Il2\_miss\_rate,Dl2\_miss\_rate,Num\_of\_Dl2Accesses,Num\_of\_memaccesses,Num\_of\_Il2Accesses

#print L1unified

#print L2unified

if (L1unified == 0)&(L2unified ==1):

#print Dl1\_block\_size

CPI = 1 + 1\*Il1\_miss\_rate\*MPL1 + DL1\_AccessPerInst\*Dl1\_miss\_rate\*MPL1 + DL2\_AccessPerInst\*Dl2\_miss\_rate\*MPL2

CPI = str(CPI)

if(int(DL1\_Associativity) > 10):

DL1\_Associativity = 'f'

if(int(IL1\_Associativity) > 10):

IL1\_Associativity = 'f'

if(int(IDL2\_Associativity) > 10):

IDL2\_Associativity = 'f'

Data\_string = str(Dl1\_block\_size) + ',' + DL1\_Associativity + ',' + DL1\_Repl + ',' + str(Il1\_block\_size) + ',' + IL1\_Associativity + ',' + IL1\_Repl + ',' + str(IDl2\_block\_size) + ',' + IDL2\_Associativity + ',' + IDL2\_Repl + ',' + CPI

Cost = DL1\_size/Ref\_size\*0.5\*5 + (Dl1\_block\_size/4)\*0.05 + CostAssociativity[DL1\_Associativity] + CostReplacement[DL1\_Repl] + IL1\_size/Ref\_size\*0.5\*5 + (Il1\_block\_size/4)\*0.05 + CostAssociativity[IL1\_Associativity] + CostReplacement[IL1\_Repl] + (IDL2\_size/Ref\_size)\*0.5 + (IDl2\_block\_size/4)\*0.05 + CostAssociativity[IDL2\_Associativity] + CostReplacement[IDL2\_Repl] + 0.4

Product = Cost \* float(CPI)

Data\_string = Data\_string + ',' + str(Cost) + ',' + str(Product)

elif (L1unified == 1)&(L2unified ==1):

CPI = 1 + DL1\_AccessPerInst\*Dl1\_miss\_rate\*MPL1 + DL2\_AccessPerInst\*Dl2\_miss\_rate\*MPL2

CPI = str(CPI)

#print IDL1\_Associativity

if(int(IDL1\_Associativity) > 10):

IDL1\_Associativity = 'f'

if(int(IDL2\_Associativity) > 10):

IDL2\_Associativity = 'f'

#print (CostAssociativity[IDL1\_Associativity], CostReplacement[IDL1\_Repl]

str(IDl1\_block\_size)

Data\_string = str(IDl1\_block\_size) + ',' + IDL1\_Associativity + ',' + IDL1\_Repl + ',' + str(IDl2\_block\_size) + ',' + IDL2\_Associativity + ',' + IDL2\_Repl + ',' + CPI

print Data\_string

Cost = IDL1\_size/Ref\_size\*0.5\*5 + (IDl1\_block\_size/4)\*0.05 + CostAssociativity[IDL1\_Associativity] + CostReplacement[IDL1\_Repl] + (IDL2\_size/Ref\_size)\*0.5 + (IDl2\_block\_size/4)\*0.05 + CostAssociativity[IDL2\_Associativity] + CostReplacement[IDL2\_Repl]

Product = Cost \* float(CPI)

Data\_string = Data\_string + ',' + str(Cost) + ',' + str(Product)

print Cost

#print Data\_string

elif (L1unified == 0)&(L2unified ==0):

if(int(IL1\_Associativity) > 10):

IL1\_Associativity = 'f'

if(int(IL2\_Associativity) > 10):

IL2\_Associativity = 'f'

if(int(DL1\_Associativity) > 10):

DL1\_Associativity = 'f'

if(int(DL2\_Associativity) > 10):

DL2\_Associativity = 'f'

CPI = 1 + 1\*Il1\_miss\_rate\*MPL1 + DL1\_AccessPerInst\*Dl1\_miss\_rate\*MPL1 + DL2\_AccessPerInst\*Dl2\_miss\_rate\*MPL2 + Il2\_AccessesPerInst\*Il2\_miss\_rate\*MPL2

CPI = str(CPI)

Data\_string = str(Dl1\_block\_size) + ',' + DL1\_Associativity + ',' + DL1\_Repl + ',' + str(Il1\_block\_size) + ',' + IL1\_Associativity + ',' + IL1\_Repl + ',' + str(Il2\_block\_size) + ',' + IL2\_Associativity + ',' + IL2\_Repl + ',' + str(Dl2\_block\_size) +',' + DL2\_Associativity +',' + DL2\_Repl +',' + CPI

Cost = DL1\_size/Ref\_size\*0.5\*5 + (Dl1\_block\_size/4)\*0.05 + CostAssociativity[DL1\_Associativity] + CostReplacement[DL1\_Repl] + IL1\_size/Ref\_size\*0.5\*5 + (Il1\_block\_size/4)\*0.05 + CostAssociativity[IL1\_Associativity] + CostReplacement[IL1\_Repl] + (DL2\_size/Ref\_size)\*0.5 + (Dl2\_block\_size/4)\*0.05 + CostAssociativity[DL2\_Associativity] + CostReplacement[DL2\_Repl] + (IL2\_size/Ref\_size)\*0.5 + (Il2\_block\_size/4)\*0.05 + CostAssociativity[IL2\_Associativity] + CostReplacement[IL2\_Repl] + 0.4 + 0.2

Product = Cost \* float(CPI)

Data\_string = Data\_string + ',' + str(Cost) + ',' + str(Product)

print Data\_string

#print CPI

#in\_file.seek(last\_pos)

#print >> res\_file, "\t\t\t\t\t\t\t\t\t\t",CPI

#print >>res\_file, CPI

#print >> res\_file,'------------------------------------------------'

#CPI = str(CPI)

#print (Il1\_miss\_rate, DL1\_AccessPerInst, Dl1\_miss\_rate, Dl2\_miss\_rate, CPI)

#Data\_string = Dl1\_block\_size + ',' + DL1\_Associativity + ',' + DL1\_Repl + ',' + Il1\_block\_size + ',' + IL1\_Associativity + ',' + IL1\_Repl + ',' + IDl2\_block\_size + ',' + IDL2\_Associativity + ',' + IDL2\_Repl + ',' + CPI

#print Data\_string

print '-----------------'

File\_data.write(Data\_string)

File\_data.write('\n')

Il1\_miss\_rate = 'NA'

Dl1\_miss\_rate = 'NA'

Il2\_miss\_rate = 'NA'

Dl2\_miss\_rate = 'NA'

L1unified = 0

L2unified = 0

in\_file.close()