**UNIVERSITY OF TEXAS at DALLAS**

**DEPARTMENT OF ELECTRICAL ENGINEERING**

**CE 6304.0012 EEDG 6304**

**10/17/2016**

**Group Number 37**

**VENKATA KARTHEEK MADHAVARAPU**

**VIKRAM SHIVAPRAKASH**

**Contribution Percentage:Kartheek 60% Vikram 40%**

**Project #1**

**CACHE DESIGN**



|  |  |  |
| --- | --- | --- |
| **S.No** | **Topic** | **Page Number** |
| **1** | Introduction | **3** |
| **2** | CPI of benchmarks | **5** |
| **3** | Optimization of CPI for each benchmark | **7** |
| **4** | Cost function | **12** |
| **5** | Optimizing Caches for Performance/Cost | **28** |

**Objective:**

1. To select an optimal cache configuration for each benchmark.
2. Identify the optimal design (cache configuration) for each benchmark in terms of CPI.
3. Present graphs showing the trade-off between CPI and cost for different designs.

**Introduction:**

Parameters needed to be provided in order to simulate the configuration

* **CPU Models**: TimingSimpleCPU (timing)

The timingSimpleCPU uses timing memory accesses .It stalls on cache accesses and waits for the memory system to respond prior to proceeding. The timing CPU is also derived from BaseSimpleCPU, and implements the same set of functions. It defines the port that is used to hook up to memory, and connects the CPU to cache.

In this project we have used only timing for CPU model.

* **Associativity:**

Caches can also take on a variety of forms and capacities. No matter how large or small they are, caches fall into one of three categories:

Direct mapped, n-way set associative, and fully associative

Higher associativity reduces miss rates. We are trying out 1way, 2 way, 4 way and 8 way associativity in our project.

Direct Mapped:

A cache block can only go in one spot in the cache. It makes a cache block very easy to

Find, but it‛s not very flexible about where to put the blocks.

2-Way set Associative

This cache is made up of sets that can fit two blocks each. The index is now used to

Find the set, and the tag helps find the block within the set.

4-Way Set Associative:

Each set here fits four blocks, so there are fewer sets. As such, fewer index bits are

Needed.

**Cache levels:**

Most CPUs have different independent caches, including instruction and data caches, where the data cache is usually organized as a hierarchy of more cache levels (L1, L2, etc.).

All current CPUs with caches have a split L1 cache and also have L2 caches. The L2 cache is usually not split and is usually shared between cores. Most modern desktop CPUs have at least three independent caches: an **instruction cache** to speed up executable instruction fetch, a **data cache** to speed up data fetch and store. The data cache is usually organized as a hierarchy of more cache levels (L1, L2, etc.)

**Cache Size:**

The size of the cache normally refers actually to the size of the data store, where the memory elements are actually stored. A typical PC level 2 cache is 512 KB, but can be as small as 64 KB on older machines, or as high as 1 MB or even 2 MB. For this project we are taking level 1 data and instruction cache ranging in size from 8 KB to 128 KB. The more cache the system has, the more likely it is to register a hit on a memory access, because fewer memory locations are forced to share the same cache line.

**Block Size:**

We have used the block size as 64 bytes and 32 Bytes for all benchmarks.

**PART2:Benchmarks:**

We have used 5 Benchmarks for this project:

401.bzip2

429.mcf

456.hmmer

458.sjeng

470.lbm

A sample command we used to run one of the combinations in 429.mcf benchmark is:

export GEM5\_DIR=/usr/local/gem5

export BENCHMARK=./src/benchmark

export ARGUMENT=./data/inp.in

time $GEM5\_DIR/build/X86/gem5.opt -d ~/m5out $GEM5\_DIR/configs/example/se.py -c $BENCHMARK -o $ARGUMENT -I 100000000 --cpu-type=atomic --caches --l2cache --l1d\_size=128kB --l1i\_size=128kB --l2\_size=1MB --l1d\_assoc=2 --l1i\_assoc=2 --l2\_assoc=1 –cacheline\_size=64

We obtain the stats file and note down readings of :

Total missed instruction l1 Icache: system.cpu.icache.overall\_misses::total 676 # number of overall misses

Total missed instuction l1 Dcache:

system.cpu.dcache.overall\_misses::total 10225987 # number of overall misses

Total missed instruction in l2 cache: system.l2.overall\_misses::total 6552130 # number of overall misses

The total Cycles Per Instruction is calculated using formula:

**CPI=1+[{(iL1.miss\_instruction\_total+dL1.miss\_instruction\_total)\*4+(L2\_miss\_instruction\_total)\*80}/{Total\_number\_of\_instruction.}]**

CPI=1+ [(676+10225987)\*4 + (6552130\*80)] / [5\*10^8] = 2.130

PART 3: OPTIMIZATION OF CPI FOR EACH BENCHMARK

* 401.bzip2

According to the statistics the best CPI was found for the following combination:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | cost |
| 401.bzip2\_64kB\_128kB\_4\_1MB\_4\_64B | 1.118426496 | 327.2 |

429.mcf

According to the statistics the best CPI was found for the following combination:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 429.mcf\_128kB\_128kB\_8\_1024kB\_8\_64B | 2.075993 | 353.2 |

458. sjeng

According to the statistics the best CPI was found for the following combination:

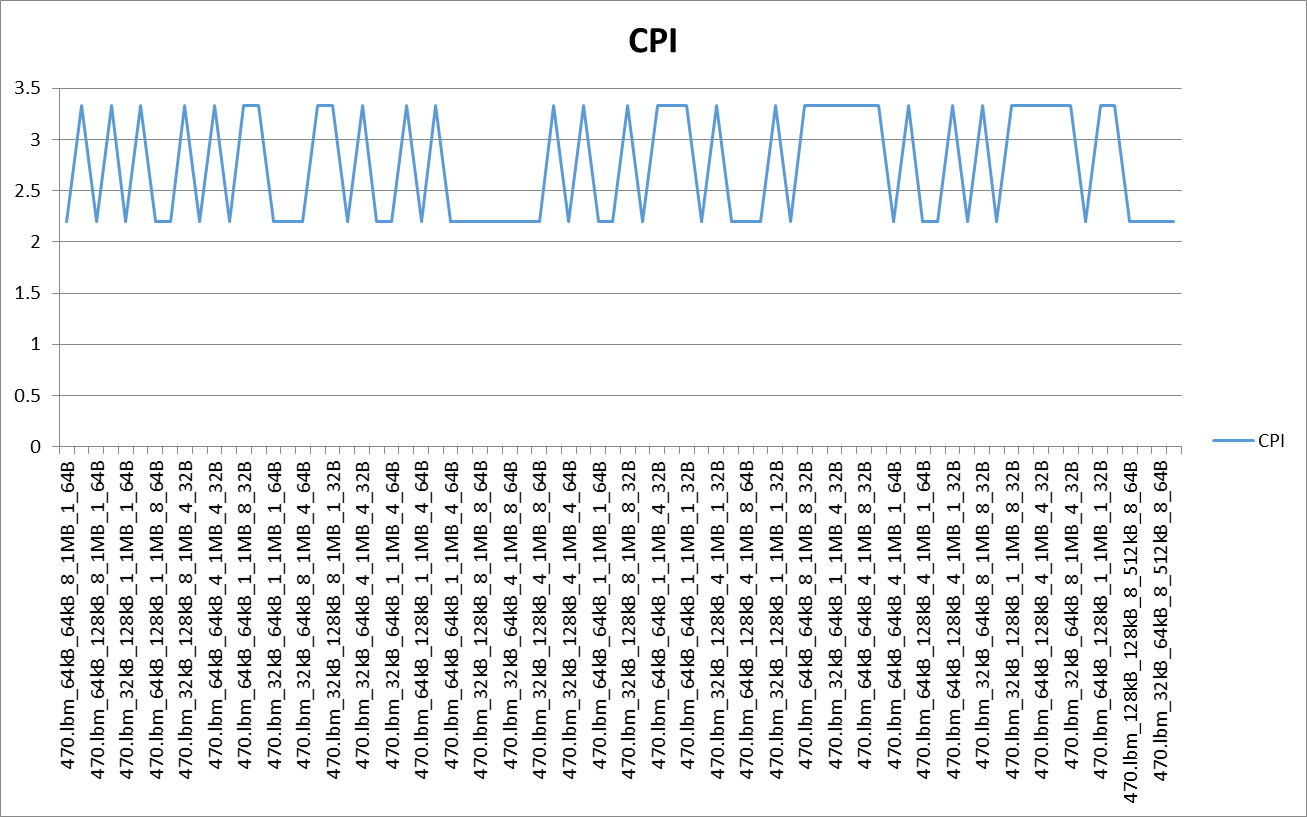
|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 458.sjeng\_64kB\_128kB\_4\_1MB\_4\_64B | 2.408137 | 327.2 |

456.hmmer

According to the statistics the best CPI was found for the following combination:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 456.hmmer\_64kB\_128kB\_4\_1MB\_4\_64B | 1.00171 | 327.2 |

470.lbm



According to the statistics the best CPI was found for the following combination:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 470.lbm\_64kB\_128kB\_4\_1MB\_8\_64B | 2.200961 | 327.4 |

**PART 4: Cost Function:**

Cost function plays a major role in determining parameters that give best CPI. So we need to define a cost function, which assists architect to design a cost efficient, and performance effective cache. It refrains us from choosing whatever design is giving the least CPI.

Cost function can be defined as below:

**Cost function=0.40\*(L1 cache size) + 0.25\*(L2 cache size) + 0.05\*(L1 associativity) + 0.05\*(L2 associativity) + 0\*(block size)**

Explanation of the cost function can be found in the below table:

|  |  |  |  |
| --- | --- | --- | --- |
| Cost Function | Weight | Overall Weight | Comment |
| L1 Cache size | .40 | 64KB:50units  128Kb:100units | If cache size doubles, cost also doubles. |
| L2 Cache size | .25 | 1 MB:250 units | If cache size doubles, cost also doubles. We have taken only 1MB for this project. |
| L1 Associativity | 0.05 | 1:2 units  2:4 units  4:8 units  8:16 units | Increasing associativity increases number of comparators on hardware, which in turn increases cost. |
| L2 Associativity | 0.05 | 1:2 units  2:4 units  4:8 units  8:16 units | Increasing associativity increases number of comparators on hardware, which in turn increases cost. |
| Block Size | 0 | 64 Bytes:0 units | With respect to hardware there will be no additional cost for change in block size. We are using 64 Bytes and 32 Bytes in this project |

401.BZIP2

According to the table the best configuration for lowest cost is:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | cost |
| 401.bzip2\_64kB\_64kB\_1\_512kB\_1\_64B | 1.19644928 | 176.3 |

429.MCF

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 429.mcf\_16kB\_64kB\_4\_1MB\_2\_64B | 2.116985 | 282.3 |
| 429.mcf\_16kB\_64kB\_4\_1MB\_1\_64B | 2.158766 | 282.25 |
| 429.mcf\_16kB\_64kB\_4\_1MB\_2\_32B | 2.739378 | 282.3 |
| 429.mcf\_16kB\_64kB\_1\_1MB\_1\_64B | 2.182483 | 282.1 |
| 429.mcf\_8kB\_32kB\_2\_1MB\_4\_32B | 2.778042 | 266.3 |
| 429.mcf\_8kB\_64kB\_2\_1MB\_2\_32B | 2.770118 | 279 |
| 429.mcf\_16kB\_32kB\_1\_1MB\_1\_64B | 2.210404 | 269.3 |
| 429.mcf\_16kB\_32kB\_2\_1MB\_2\_32B | 2.776236 | 269.4 |
| 429.mcf\_8kB\_32kB\_1\_1MB\_2\_32B | 2.843907 | 266.15 |
| 429.mcf\_8kB\_64kB\_1\_1MB\_2\_64B | 2.170489 | 278.95 |
| 429.mcf\_16kB\_64kB\_1\_1MB\_4\_32B | 2.748257 | 282.25 |
| 429.mcf\_8kB\_64kB\_4\_1MB\_2\_64B | 2.140811 | 279.1 |
| 429.mcf\_8kB\_64kB\_1\_1MB\_4\_64B | 2.154678 | 279.05 |
| 429.mcf\_8kB\_32kB\_2\_1MB\_2\_64B | 2.162266 | 266.2 |
| 429.mcf\_8kB\_64kB\_2\_1MB\_1\_64B | 2.189009 | 278.95 |
| 429.mcf\_16kB\_32kB\_1\_1MB\_2\_64B | 2.157398 | 269.35 |
| 429.mcf\_8kB\_64kB\_1\_1MB\_1\_32B | 2.885251 | 278.9 |
| 429.mcf\_8kB\_64kB\_4\_1MB\_1\_64B | 2.182648 | 279.05 |
| 429.mcf\_16kB\_64kB\_4\_1MB\_1\_32B | 2.798687 | 282.25 |
| 429.mcf\_16kB\_32kB\_2\_1MB\_4\_32B | 2.760234 | 269.5 |
| 429.mcf\_8kB\_32kB\_2\_1MB\_1\_32B | 2.867166 | 266.15 |
| 429.mcf\_16kB\_64kB\_1\_1MB\_2\_32B | 2.764521 | 282.15 |
| 429.mcf\_16kB\_64kB\_2\_1MB\_4\_32B | 2.736435 | 282.3 |
| 429.mcf\_8kB\_64kB\_4\_1MB\_1\_32B | 2.80863 | 279.05 |
| 429.mcf\_8kB\_64kB\_2\_1MB\_4\_32B | 2.754243 | 279.1 |
| 429.mcf\_8kB\_64kB\_4\_1MB\_4\_64B | 2.125802 | 279.2 |
| 429.mcf\_16kB\_32kB\_2\_1MB\_2\_64B | 2.142791 | 269.4 |
| 429.mcf\_16kB\_32kB\_1\_1MB\_1\_32B | 2.867643 | 269.3 |
| 429.mcf\_16kB\_32kB\_4\_1MB\_1\_64B | 2.186527 | 269.45 |
| 429.mcf\_8kB\_32kB\_2\_1MB\_1\_64B | 2.213458 | 266.15 |
| 429.mcf\_16kB\_64kB\_1\_1MB\_1\_32B | 2.833702 | 282.1 |
| 429.mcf\_16kB\_64kB\_2\_1MB\_1\_64B | 2.169472 | 282.15 |
| 429.mcf\_8kB\_64kB\_2\_1MB\_2\_64B | 2.145808 | 279 |
| 429.mcf\_8kB\_32kB\_2\_1MB\_4\_64B | 2.146729 | 266.3 |
| 429.mcf\_8kB\_32kB\_4\_1MB\_2\_64B | 2.158967 | 266.3 |
| 429.mcf\_16kB\_32kB\_2\_1MB\_1\_64B | 2.193922 | 269.35 |
| 429.mcf\_16kB\_32kB\_4\_1MB\_2\_32B | 2.768143 | 269.5 |
| 429.mcf\_16kB\_32kB\_1\_1MB\_4\_32B | 2.776317 | 269.45 |
| 429.mcf\_16kB\_32kB\_1\_1MB\_2\_32B | 2.792477 | 269.35 |
| 429.mcf\_16kB\_32kB\_4\_1MB\_1\_32B | 2.841319 | 269.45 |
| 429.mcf\_8kB\_64kB\_2\_1MB\_4\_64B | 2.130456 | 279.1 |
| 429.mcf\_8kB\_32kB\_1\_1MB\_2\_64B | 2.193073 | 266.15 |
| 429.mcf\_16kB\_64kB\_4\_1MB\_4\_32B | 2.724036 | 282.4 |
| 429.mcf\_16kB\_32kB\_4\_1MB\_4\_64B | 2.119637 | 269.6 |
| 429.mcf\_16kB\_64kB\_2\_1MB\_4\_64B | 2.110985 | 282.3 |
| 429.mcf\_8kB\_32kB\_4\_1MB\_1\_32B | 2.85126 | 266.25 |
| 429.mcf\_16kB\_32kB\_4\_1MB\_2\_64B | 2.135142 | 269.5 |
| 429.mcf\_8kB\_32kB\_2\_1MB\_2\_32B | 2.794046 | 266.2 |
| 429.mcf\_16kB\_64kB\_2\_1MB\_2\_32B | 2.752307 | 282.2 |
| 429.mcf\_8kB\_32kB\_4\_1MB\_1\_64B | 2.21041 | 266.25 |
| 429.mcf\_16kB\_32kB\_4\_1MB\_4\_32B | 2.752197 | 269.6 |
| 429.mcf\_16kB\_64kB\_2\_1MB\_1\_32B | 2.814419 | 282.15 |
| 429.mcf\_8kB\_64kB\_4\_1MB\_4\_32B | 2.733934 | 279.2 |
| 429.mcf\_8kB\_32kB\_4\_1MB\_2\_32B | 2.778043 | 266.3 |
| 429.mcf\_8kB\_32kB\_1\_1MB\_1\_64B | 2.246179 | 266.1 |
| 429.mcf\_16kB\_64kB\_2\_1MB\_2\_64B | 2.126332 | 282.2 |
| 429.mcf\_8kB\_32kB\_4\_1MB\_4\_64B | 2.143456 | 266.4 |
| 429.mcf\_16kB\_64kB\_1\_1MB\_2\_64B | 2.134809 | 282.15 |
| 429.mcf\_16kB\_32kB\_1\_1MB\_4\_64B | 2.14165 | 269.45 |
| 429.mcf\_16kB\_64kB\_4\_1MB\_4\_64B | 2.101983 | 282.4 |
| 429.mcf\_16kB\_32kB\_2\_1MB\_1\_32B | 2.849303 | 269.35 |
| 429.mcf\_8kB\_32kB\_4\_1MB\_4\_32B | 2.762095 | 266.4 |
| 429.mcf\_8kB\_32kB\_1\_1MB\_4\_32B | 2.827742 | 266.25 |
| 429.mcf\_8kB\_32kB\_1\_1MB\_4\_64B | 2.177316 | 266.25 |
| 429.mcf\_8kB\_64kB\_2\_1MB\_1\_32B | 2.832281 | 278.95 |
| 429.mcf\_16kB\_64kB\_1\_1MB\_4\_64B | 2.119012 | 282.25 |
| 429.mcf\_8kB\_32kB\_1\_1MB\_1\_32B | 2.919194 | 266.1 |
| 429.mcf\_8kB\_64kB\_1\_1MB\_1\_64B | 2.218254 | 278.9 |
| 429.mcf\_8kB\_64kB\_1\_1MB\_4\_32B | 2.799682 | 279.05 |
| 429.mcf\_16kB\_32kB\_2\_1MB\_4\_64B | 2.127258 | 269.5 |
| 429.mcf\_8kB\_64kB\_1\_1MB\_2\_32B | 2.815954 | 278.95 |
| 429.mcf\_8kB\_64kB\_4\_1MB\_2\_32B | 2.749278 | 279.1 |
| 429.mcf\_128kB\_128kB\_8\_512kB\_8\_64B | 2.183761 | 228.2 |
| 429.mcf\_128kB\_128kB\_8\_1024kB\_8\_64B | 2.075993 | 353.2 |
| 429.mcf\_32kB\_64kB\_8\_512kB\_8\_64B | 2.206449 | 164.2 |
| 429.mcf\_64kB\_64kB\_8\_512kB\_8\_64B | 2.206448 | 177 |

According to the table the best configuration for lowest cost is:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 429.mcf\_32kB\_64kB\_8\_512kB\_8\_64B | 2.206449 | 164.2 |

458.sjeng

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 458.sjeng\_64kB\_128kB\_4\_1MB\_2\_32B | 3.801471 | 327.1 |
| 458.sjeng\_64kB\_128kB\_2\_1MB\_4\_32B | 3.805094 | 327.1 |
| 458.sjeng\_64kB\_64kB\_1\_1MB\_1\_64B | 2.427848 | 301.3 |
| 458.sjeng\_32kB\_128kB\_2\_1MB\_2\_64B | 2.425298 | 314.2 |
| 458.sjeng\_32kB\_64kB\_2\_1MB\_1\_64B | 2.428479 | 288.55 |
| 458.sjeng\_32kB\_128kB\_2\_1MB\_4\_64B | 2.424989 | 314.3 |
| 458.sjeng\_64kB\_128kB\_1\_1MB\_1\_64B | 2.424943 | 326.9 |
| 458.sjeng\_64kB\_64kB\_2\_1MB\_4\_64B | 2.412334 | 301.5 |
| 458.sjeng\_64kB\_64kB\_4\_1MB\_2\_32B | 3.802325 | 301.5 |
| 458.sjeng\_32kB\_64kB\_4\_1MB\_1\_32B | 3.812695 | 288.65 |
| 458.sjeng\_64kB\_64kB\_4\_1MB\_4\_32B | 3.802071 | 301.6 |
| 458.sjeng\_32kB\_64kB\_2\_1MB\_4\_64B | 2.426029 | 288.7 |
| 458.sjeng\_32kB\_128kB\_2\_1MB\_2\_32B | 3.82371 | 314.2 |
| 458.sjeng\_64kB\_64kB\_2\_1MB\_1\_64B | 2.414162 | 301.35 |
| 458.sjeng\_64kB\_64kB\_2\_1MB\_2\_64B | 2.4126 | 301.4 |
| 458.sjeng\_64kB\_64kB\_1\_1MB\_4\_64B | 2.423732 | 301.45 |
| 458.sjeng\_32kB\_64kB\_2\_1MB\_4\_32B | 3.824416 | 288.7 |
| 458.sjeng\_64kB\_64kB\_1\_1MB\_2\_64B | 2.424042 | 301.35 |
| 458.sjeng\_64kB\_128kB\_2\_1MB\_1\_64B | 2.412273 | 326.95 |
| 458.sjeng\_64kB\_64kB\_4\_1MB\_4\_64B | 2.408878 | 301.6 |
| 458.sjeng\_32kB\_128kB\_4\_1MB\_2\_32B | 3.810191 | 314.3 |
| 458.sjeng\_64kB\_64kB\_1\_1MB\_1\_32B | 3.824117 | 301.3 |
| 458.sjeng\_32kB\_64kB\_4\_1MB\_4\_64B | 2.416205 | 288.8 |
| 458.sjeng\_32kB\_128kB\_1\_1MB\_4\_32B | 3.842494 | 314.25 |
| 458.sjeng\_64kB\_128kB\_1\_1MB\_2\_32B | 3.818235 | 326.95 |
| 458.sjeng\_32kB\_64kB\_4\_1MB\_2\_64B | 2.416485 | 288.7 |
| 458.sjeng\_32kB\_128kB\_1\_1MB\_2\_64B | 2.43829 | 314.15 |
| 458.sjeng\_32kB\_128kB\_4\_1MB\_1\_32B | 3.811029 | 314.25 |
| 458.sjeng\_64kB\_128kB\_4\_1MB\_4\_64B | 2.408137 | 327.2 |
| 458.sjeng\_32kB\_128kB\_4\_1MB\_2\_64B | 2.415727 | 314.3 |
| 458.sjeng\_64kB\_128kB\_2\_1MB\_4\_64B | 2.411288 | 327.1 |
| 458.sjeng\_32kB\_128kB\_2\_1MB\_1\_64B | 2.426373 | 314.15 |
| 458.sjeng\_32kB\_128kB\_4\_1MB\_4\_64B | 2.415469 | 314.4 |
| 458.sjeng\_64kB\_64kB\_2\_1MB\_1\_32B | 3.808053 | 301.35 |
| 458.sjeng\_64kB\_128kB\_4\_1MB\_1\_64B | 2.408837 | 327.05 |
| 458.sjeng\_64kB\_64kB\_1\_1MB\_2\_32B | 3.819962 | 301.35 |
| 458.sjeng\_64kB\_128kB\_4\_1MB\_1\_32B | 3.802014 | 327.05 |
| 458.sjeng\_32kB\_64kB\_4\_1MB\_2\_32B | 3.811037 | 288.7 |
| 458.sjeng\_64kB\_128kB\_1\_1MB\_4\_32B | 3.817836 | 327.05 |
| 458.sjeng\_32kB\_128kB\_1\_1MB\_1\_32B | 3.847083 | 314.1 |
| 458.sjeng\_64kB\_128kB\_4\_1MB\_2\_64B | 2.408341 | 327.1 |
| 458.sjeng\_64kB\_128kB\_1\_1MB\_2\_64B | 2.421767 | 326.95 |
| 458.sjeng\_64kB\_64kB\_4\_1MB\_1\_32B | 3.803568 | 301.45 |
| 458.sjeng\_32kB\_128kB\_2\_1MB\_4\_32B | 3.823402 | 314.3 |
| 458.sjeng\_32kB\_64kB\_1\_1MB\_4\_64B | 2.440153 | 288.65 |
| 458.sjeng\_64kB\_64kB\_4\_1MB\_1\_64B | 2.410222 | 301.45 |
| 458.sjeng\_32kB\_64kB\_4\_1MB\_1\_64B | 2.418088 | 288.65 |
| 458.sjeng\_64kB\_64kB\_2\_1MB\_4\_32B | 3.806115 | 301.5 |
| 458.sjeng\_32kB\_128kB\_4\_1MB\_1\_64B | 2.41655 | 314.25 |
| 458.sjeng\_64kB\_128kB\_1\_1MB\_4\_64B | 2.421373 | 327.05 |
| 458.sjeng\_64kB\_128kB\_2\_1MB\_1\_32B | 3.806128 | 326.95 |
| 458.sjeng\_32kB\_64kB\_1\_1MB\_4\_32B | 3.844287 | 288.65 |
| 458.sjeng\_64kB\_64kB\_2\_1MB\_2\_32B | 3.806403 | 301.4 |
| 458.sjeng\_32kB\_64kB\_1\_1MB\_1\_64B | 2.445636 | 288.5 |
| 458.sjeng\_32kB\_64kB\_2\_1MB\_2\_32B | 3.824737 | 288.6 |
| 458.sjeng\_32kB\_64kB\_1\_1MB\_2\_64B | 2.44054 | 288.55 |
| 458.sjeng\_64kB\_128kB\_2\_1MB\_2\_64B | 2.411544 | 327 |
| 458.sjeng\_32kB\_64kB\_2\_1MB\_1\_32B | 3.82679 | 288.55 |
| 458.sjeng\_32kB\_64kB\_2\_1MB\_2\_64B | 2.426346 | 288.6 |
| 458.sjeng\_64kB\_128kB\_1\_1MB\_1\_32B | 3.821748 | 326.9 |
| 458.sjeng\_64kB\_64kB\_1\_1MB\_4\_32B | 3.819634 | 301.45 |
| 458.sjeng\_32kB\_64kB\_1\_1MB\_2\_32B | 3.844668 | 288.55 |
| 458.sjeng\_64kB\_64kB\_4\_1MB\_2\_64B | 2.409105 | 301.5 |
| 458.sjeng\_32kB\_128kB\_2\_1MB\_1\_32B | 3.824768 | 314.15 |
| 458.sjeng\_32kB\_64kB\_4\_1MB\_4\_32B | 3.810751 | 288.8 |
| 458.sjeng\_32kB\_128kB\_1\_1MB\_4\_64B | 2.437795 | 314.25 |
| 458.sjeng\_32kB\_128kB\_1\_1MB\_2\_32B | 3.842964 | 314.15 |
| 458.sjeng\_64kB\_128kB\_4\_1MB\_4\_32B | 3.801251 | 327.2 |
| 458.sjeng\_64kB\_128kB\_2\_1MB\_2\_32B | 3.805366 | 327 |
| 458.sjeng\_32kB\_64kB\_1\_1MB\_1\_32B | 3.849591 | 288.5 |
| 458.sjeng\_32kB\_128kB\_1\_1MB\_1\_64B | 2.442207 | 314.1 |
| 458.sjeng\_32kB\_128kB\_4\_1MB\_4\_32B | 3.809936 | 314.4 |
| 458.sjeng\_64kB\_128kB\_2\_512kB\_2\_64B | 2.412648 | 202 |
| 458.sjeng\_128kB\_128kB\_2\_512kB\_2\_64B | 2.408714 | 227.6 |
| 458.sjeng\_32kB\_64kB\_2\_512kB\_2\_64B | 2.427826 | 163.6 |
| 458.sjeng\_64kB\_64kB\_2\_512kB\_2\_64B | 2.413766 | 176.4 |

According to the table, the best configuration for lowest cost:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 458.sjeng\_32kB\_64kB\_2\_512kB\_2\_64B | 2.427826 | 163.6 |

456. hmmer

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 456.hmmer\_64kB\_128kB\_4\_1MB\_1\_32B | 1.003211 | 327.05 |
| 456.hmmer\_32kB\_64kB\_1\_1MB\_1\_64B | 1.007519 | 288.5 |
| 456.hmmer\_32kB\_64kB\_2\_1MB\_1\_64B | 1.003875 | 288.55 |
| 456.hmmer\_32kB\_128kB\_2\_1MB\_4\_64B | 1.001958 | 314.3 |
| 456.hmmer\_32kB\_64kB\_1\_1MB\_4\_32B | 1.009953 | 288.65 |
| 456.hmmer\_32kB\_128kB\_2\_1MB\_1\_64B | 1.002022 | 314.15 |
| 456.hmmer\_32kB\_64kB\_1\_1MB\_2\_64B | 1.00739 | 288.55 |
| 456.hmmer\_32kB\_128kB\_2\_1MB\_2\_64B | 1.002019 | 314.2 |
| 456.hmmer\_64kB\_64kB\_1\_1MB\_1\_32B | 1.009865 | 301.3 |
| 456.hmmer\_32kB\_128kB\_4\_1MB\_2\_64B | 1.001711 | 314.3 |
| 456.hmmer\_32kB\_64kB\_1\_1MB\_1\_32B | 1.010394 | 288.5 |
| 456.hmmer\_32kB\_64kB\_4\_1MB\_1\_32B | 1.007534 | 288.65 |
| 456.hmmer\_64kB\_128kB\_2\_1MB\_4\_64B | 1.001957 | 327.1 |
| 456.hmmer\_64kB\_64kB\_4\_1MB\_1\_64B | 1.003897 | 301.45 |
| 456.hmmer\_32kB\_128kB\_4\_1MB\_1\_32B | 1.003214 | 314.25 |
| 456.hmmer\_64kB\_64kB\_1\_1MB\_4\_64B | 1.007206 | 301.45 |
| 456.hmmer\_32kB\_128kB\_1\_1MB\_4\_64B | 1.003619 | 314.25 |
| 456.hmmer\_64kB\_128kB\_1\_1MB\_4\_32B | 1.005708 | 327.05 |
| 456.hmmer\_32kB\_64kB\_4\_1MB\_4\_64B | 1.003897 | 288.8 |
| 456.hmmer\_64kB\_64kB\_1\_1MB\_1\_64B | 1.007208 | 301.3 |
| 456.hmmer\_64kB\_64kB\_1\_1MB\_4\_32B | 1.009862 | 301.45 |
| 456.hmmer\_64kB\_128kB\_1\_1MB\_4\_64B | 1.003559 | 327.05 |
| 456.hmmer\_32kB\_64kB\_2\_1MB\_2\_64B | 1.00387 | 288.6 |
| 456.hmmer\_32kB\_64kB\_4\_1MB\_1\_64B | 1.003899 | 288.65 |
| 456.hmmer\_32kB\_128kB\_1\_1MB\_1\_32B | 1.006047 | 314.1 |
| 456.hmmer\_64kB\_128kB\_2\_1MB\_1\_32B | 1.003799 | 326.95 |
| 456.hmmer\_64kB\_128kB\_2\_1MB\_1\_64B | 1.002019 | 326.95 |
| 456.hmmer\_64kB\_64kB\_2\_1MB\_4\_32B | 1.007226 | 301.5 |
| 456.hmmer\_64kB\_128kB\_1\_1MB\_2\_32B | 1.005708 | 326.95 |
| 456.hmmer\_32kB\_128kB\_4\_1MB\_4\_64B | 1.001711 | 314.4 |
| 456.hmmer\_64kB\_64kB\_1\_1MB\_2\_64B | 1.007206 | 301.35 |
| 456.hmmer\_32kB\_128kB\_2\_1MB\_1\_32B | 1.003802 | 314.15 |
| 456.hmmer\_32kB\_128kB\_4\_1MB\_1\_64B | 1.001713 | 314.25 |
| 456.hmmer\_64kB\_128kB\_2\_1MB\_2\_64B | 1.002018 | 327 |
| 456.hmmer\_64kB\_64kB\_2\_1MB\_1\_64B | 1.003872 | 301.35 |
| 456.hmmer\_32kB\_64kB\_4\_1MB\_2\_64B | 1.003897 | 288.7 |
| 456.hmmer\_64kB\_64kB\_2\_1MB\_1\_32B | 1.00743 | 301.35 |
| 456.hmmer\_32kB\_128kB\_1\_1MB\_2\_32B | 1.005895 | 314.15 |
| 456.hmmer\_64kB\_128kB\_4\_1MB\_1\_64B | 1.001711 | 327.05 |
| 456.hmmer\_64kB\_128kB\_4\_1MB\_4\_64B | 1.00171 | 327.2 |
| 456.hmmer\_32kB\_64kB\_2\_1MB\_4\_32B | 1.007228 | 288.7 |
| 456.hmmer\_64kB\_128kB\_1\_1MB\_1\_64B | 1.003561 | 326.9 |
| 456.hmmer\_64kB\_128kB\_1\_1MB\_2\_64B | 1.003559 | 326.95 |
| 456.hmmer\_64kB\_64kB\_4\_1MB\_4\_64B | 1.003896 | 301.6 |
| 456.hmmer\_64kB\_64kB\_4\_1MB\_1\_32B | 1.007532 | 301.45 |
| 456.hmmer\_32kB\_64kB\_4\_1MB\_2\_32B | 1.007531 | 288.7 |
| 456.hmmer\_32kB\_128kB\_2\_1MB\_2\_32B | 1.003746 | 314.2 |
| 456.hmmer\_64kB\_128kB\_2\_1MB\_4\_32B | 1.003676 | 327.1 |
| 456.hmmer\_32kB\_64kB\_1\_1MB\_2\_32B | 1.010083 | 288.55 |
| 456.hmmer\_64kB\_64kB\_1\_1MB\_2\_32B | 1.009862 | 301.35 |
| 456.hmmer\_32kB\_128kB\_4\_1MB\_2\_32B | 1.003211 | 314.3 |
| 456.hmmer\_64kB\_64kB\_2\_1MB\_2\_32B | 1.007334 | 301.4 |
| 456.hmmer\_64kB\_64kB\_4\_1MB\_2\_32B | 1.00753 | 301.5 |
| 456.hmmer\_32kB\_128kB\_2\_1MB\_4\_32B | 1.003677 | 314.3 |
| 456.hmmer\_32kB\_64kB\_4\_1MB\_4\_32B | 1.00753 | 288.8 |
| 456.hmmer\_64kB\_64kB\_4\_1MB\_4\_32B | 1.007529 | 301.6 |
| 456.hmmer\_64kB\_128kB\_1\_1MB\_1\_32B | 1.005716 | 326.9 |
| 456.hmmer\_64kB\_128kB\_4\_1MB\_4\_32B | 1.00321 | 327.2 |
| 456.hmmer\_32kB\_128kB\_1\_1MB\_4\_32B | 1.005799 | 314.25 |
| 456.hmmer\_64kB\_128kB\_4\_1MB\_2\_64B | 1.00171 | 327.1 |
| 456.hmmer\_64kB\_64kB\_2\_1MB\_2\_64B | 1.003868 | 301.4 |
| 456.hmmer\_64kB\_128kB\_4\_1MB\_2\_32B | 1.00321 | 327.1 |
| 456.hmmer\_64kB\_64kB\_4\_1MB\_2\_64B | 1.003896 | 301.5 |
| 456.hmmer\_32kB\_64kB\_2\_1MB\_2\_32B | 1.007335 | 288.6 |
| 456.hmmer\_64kB\_64kB\_2\_1MB\_4\_64B | 1.003768 | 301.5 |
| 456.hmmer\_32kB\_128kB\_4\_1MB\_4\_32B | 1.003211 | 314.4 |
| 456.hmmer\_32kB\_64kB\_2\_1MB\_4\_64B | 1.003769 | 288.7 |
| 456.hmmer\_32kB\_128kB\_1\_1MB\_1\_64B | 1.003776 | 314.1 |
| 456.hmmer\_64kB\_128kB\_2\_1MB\_2\_32B | 1.003745 | 327 |
| 456.hmmer\_32kB\_64kB\_2\_1MB\_1\_32B | 1.007435 | 288.55 |
| 456.hmmer\_32kB\_64kB\_1\_1MB\_4\_64B | 1.007266 | 288.65 |
| 456.hmmer\_32kB\_128kB\_1\_1MB\_2\_64B | 1.003712 | 314.15 |
| 456.hmmer\_64kB\_128kB\_1\_512kB\_1\_64B | 1.003874 | 201.9 |
| 456.hmmer\_128kB\_128kB\_1\_512kB\_1\_64B | 1.003822 | 227.5 |
| 456.hmmer\_32kB\_64kB\_1\_512kB\_1\_64B | 1.008266 | 163.5 |
| 456.hmmer\_64kB\_64kB\_1\_512kB\_1\_64B | 1.00787 | 176.3 |

According to table the configuration for lowest cost is:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 456.hmmer\_32kB\_64kB\_1\_512kB\_1\_64B | 1.008266 | 163.5 |

470.LBM

According to table the best configuration for lowest cost is:

|  |  |  |
| --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost |
| 470.lbm\_32kB\_64kB\_8\_512kB\_8\_64B | 2.20098 | 164.2 |

**PART 5:Optimizing Caches for Performance/Cost**

401.bzip2

According to table the best configuration for best performance is :

|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | cost | CPI\*COST |
| 401.bzip2\_64kB\_64kB\_1\_512kB\_1\_64B | 1.19644928 | 176.3 | 210.934 |

429.mcf

According to table the best configuration for best performance is :

|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost | CPI\*COST |
| 429.mcf\_32kB\_64kB\_8\_512kB\_8\_64B | 2.206449 | 164.2 | 362.2989 |

458. sjeng

According to table the best configuration for best performance is :

|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost | CPI\*COST |
| 458.sjeng\_32kB\_64kB\_2\_512kB\_2\_64B | 2.427826 | 163.6 | 397.1923 |

456.hmmer

According to table the best configuration for best performance is:

|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost | CPI\*COST |
| 456.hmmer\_32kB\_64kB\_1\_512kB\_1\_64B | 1.008266 | 163.5 | 164.8516 |

470.lbm

According to table the best configuration for best performance is:

|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark\_icachesize\_dcachesize\_l1associativity\_l2size\_l2associativity\_Cacheline size | CPI | Cost | CPI\*COST |
| 470.lbm\_32kB\_64kB\_8\_512kB\_8\_64B | 2.20098 | 164.2 | 361.4009 |

The best configuration involving all the benchmarks combined is:

The CPI and cost are two main factors while deciding the configuration and the way to find out a good balance is taking the product of cost and CPI. The best configuration is shown below

32kB\_64kB\_1\_512kB\_1\_64B

|  |  |
| --- | --- |
| Benchmark | CPI |
|  |  |
| 401.bzip2 | 1.196449 |
| 429.mcf | 2.206 |
| 458.sjeng | 2.427 |
| 456.hmmer | 1.008 |
| 470.lbm | 2.2009 |

* CPU type: Timing
* L1\_I\_Cache: 32KB
* L1\_D\_Cache:64KB
* L2\_Cache:512KB
* Block size:64bytes
* L1\_I and L1\_D Associativity:1
* L2 associativity: 1