**CS 6290 High Performance Computer Architecture**

**Lab 5 Report**

Power Simulation and Perceptron Branch Predictor

**Name** : Raghavendra Vinayak Belapure

**T-Square Account Name** : rbelapure3

# Power Simulation

The aim of the experiment is to generate the values of different power parameters using McPat Power simulator. The default values chosen are as follows.

|  |  |  |
| --- | --- | --- |
| G-share History Len : 12 | Cache Size : 1 MB | Type : 8 way set associative |
| D-cache latency : 5 cycles | DRAM Row Buffer Hit latency : 100 | DRAM Row Buffer Miss latency : 200 |
| Hardware Threads : 1 | Clock Frequency : 1200MHz | Trace File : test2.pzip |

## 1.1 Stack Column Graphs

First, we will draw stack column graph for runtime dynamic power and energy consumed that shows the breakdown of power for Instruction fetch unit, Load & Store unit, memory management unit, execution unit, L2, First level directory, NoCs and Memory Controllers. The exact values are shown below in Table.

|  |  |  |
| --- | --- | --- |
|  | Runtime Dynamic Power | Energy |
| Instruction Fetch Unit | 5.28593 | 0.043820157 |
| Load Store Unit | 3.39639 | 0.028155943 |
| Memory Management Unit | 0.163693 | 0.001357009 |
| Execution Unit | 0.258174 | 0.002140253 |
| L2 | 1.25E-08 | 1.03298E-10 |
| First Level Directory | 0 | 0 |
| Memory Controller | 0.155358 | 0.001287912 |
| NOC | 0.00177379 | 1.47047E-05 |

It can be seen that Instruction Fetch unit and Load Store unit are the major components that consume 94% of the power. Number of cycles taken to run test2.pzip with the default configuration is 9947954 and the processor runs at 1200MHz. Thus, the time taken to execute the program is 9947954 / 1200 Mhz = 0.008289962 sec. Energy is calculated as Energy = Power x Time. Now, let us draw a detailed power consumption graph for various subcomponents of the processor. They show most power consuming members of each subcomponent.

|  |  |
| --- | --- |
|  | Instruction Fetch Unit |
| Instruction Cache | 5.12631 |
| Instruction Buffer | 0.002336 |
| Instruction Decoder | 0.05715 |

|  |  |
| --- | --- |
|  | Load Store Unit |
| Data Cache | 3.28224 |
| Load Store Queue | 0.014021 |

|  |  |
| --- | --- |
|  | Memory Management Unit |
| ITLB | 0.03966 |
| DTLB | 0.023896 |

|  |  |
| --- | --- |
|  | Execution Unit |
| Register Files | 0.001653 |
| Instruction Scheduler | 0 |
| Integer ALUs | 0.023263 |
| Floating Point Units | 1.02E-07 |
| Complex ALUs | 0.119974 |
| Results Broadcast Bus | 0.013146 |

## 1.2 Clock Frequency Effects

We now observe the effects of increasing the frequency. Power is directly proportional to frequency. As a result increase in frequency will cause increase in power. The result is shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Frequency | IPC | Power | Time | Energy | EDP |
| 900 | 0.337231 | 6.9744 | 0.011053 | 0.07709 | 0.000852 |
| 1200 | 0.337231 | 9.26133 | 0.00829 | 0.076776 | 0.000636 |
| 1500 | 0.337231 | 11.5482 | 0.006632 | 0.076587 | 0.000508 |

## 1.3 Branch Prediction Effects

Increasing history length will result in increase in branch prediction accuracy. The IPC is improving as there are less pipeline flushes due to wrong path execution. But, the power consumption increases due to increase in branch prediction hardware.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hist Len | IPC | Power | Time | Energy | EDP |
| 4 | 0.331357 | 9.10869 | 0.008289 | 0.075502 | 0.000626 |
| 8 | 0.335605 | 9.21917 | 0.008289 | 0.076418 | 0.000633 |
| 12 | 0.337231 | 9.26133 | 0.008289 | 0.076767 | 0.000636 |

## 1.4 Cache Size Effects

Increase in cache size increases the cache hardware, added tag matching and increase in latency. This causes increase in power consumption. IPC should ideally decrease, but as the working set fits within 1MB cache, we are not able to observe any change in IPC.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cache Size | IPC | Power | Time | Energy | EDP |
| 1 MB | 0.337231 | 9.26133 | 0.008289 | 0.076767 | 0.000636 |
| 2 MB | 0.337231 | 11.0357 | 0.008289 | 0.091475 | 0.000758 |
| 4 MB | 0.337231 | 14.3949 | 0.008289 | 0.119319 | 0.000989 |

## 1.5 MT Simulations

Adding the multithreading can have several implications. If the working set of all the threads fits in cache and if the two threads do not have adverse effect on each other in Global History Register, it will result in increase in IPC. But, increasing threads will have extra MT hardware in each stage of pipeline, which will cause power to increase.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Threads count | IPC | Power | Time | Energy | EDP |
| 1 | 0.326321 | 11.9908 | 0.008289 | 0.099392 | 0.000824 |
| 2 | 0.332954 | 12.3479 | 0.008289 | 0.102352 | 0.000848 |
| 4 | 0.317087 | 12.7254 | 0.008289 | 0.105481 | 0.000874 |

## 1.6 Power Counters

The values of power counters for the default configuration are as follows.

|  |  |
| --- | --- |
| Floating point instructions : 4401  Integer instructions : 1759153  Load INstructions497051  Store Instructions : 400692  Branch Instructions : 696486  Integer register reads : 682248  Floating point register reads : 1  Integer Register writes : 533457 | Floating Point Register writes : 0  Multiply Instructions : 1759252  Dcache Reads : 1039102  Dcahe Writes : 982186  Num Sched Accesses : 3987419  Num Mem Accesses : 7202  Dcache Read misses : 4071  Dcache Write Misses : 48370 |

# Perceptron Branch Predictor

The aim of this experiment is to implement the perceptron branch predictor and compare branch prediction accuracy results with the g-share branch predictor.

## 2.1 Varying Storage Budget

We vary the storage budget for the branch predictor to 1KB, 4KB, 8KB and 16 KB. We calculate the history length from the following relationships.

For G-share predictor, the storage budget = (number of PHT entries) x 2 bit counter = 2 x (2^hist\_len)

Thus, .

In case of Perceptron predictor,

Storage budget = (Number of counter bits for weight) x (Number of perceptrons) x (hist\_length + 1)

Thus, the calculated history length for perceptron and g-share predictor for default number of perceptrons (i.e. 512) is shown below.

|  |  |  |
| --- | --- | --- |
| Storage Budget | G-share History length | Perceptron History Length |
| 1 KB | 9 | 1 |
| 4 KB | 11 | 7 |
| 8 KB | 12 | 15 |
| 16 KB | 13 | 31 |

Now, we compare performance of G-share branch predictor with the perceptron predictor. As we increase the storage budget, we can see increase in branch prediction accuracy. This can be seen from the fact that misses per kilo instructions (MPKI) decreases. The results show that G-share performs better than perceptron predictor. This could be due to several reasons.

The accuracy of perceptron predictor is more if the branch function is *linearly separable*. If the trace is not linearly separable, the g-share accuracy will be more as it is a history based predictor, unlike perceptron, which makes correlations between branches and GHR bits.

Also, it can be seen that for the given storage budget, G-share can have more history bits than perceptron. As a result, G-share is more accurate than perceptron in these cases.

Moreover, we are using default threshold value of zero in this experiment. Using a good threshold value for training will have a significant effect on accuracy of perceptron.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trace | Storage Budget | Number of Instructions | G-share miss count | G-share MPKI | Perceptron miss count | Perceptron MPKI |
| Long-1 | 1 KB | 19742295 | 255541 | 12.94383454 | 787362 | 39.8819894 |
| Long-1 | 4 KB | 19742295 | 158644 | 8.035742552 | 394343 | 19.97452677 |
| Long-1 | 8 KB | 19742295 | 158634 | 8.035236025 | 244418 | 12.38042487 |
| Long-1 | 16 KB | 19742295 | 145148 | 7.352134086 | 92098 | 4.665009818 |
|  |  |  |  |  |  |  |
| Long-2 | 1 KB | 26790387 | 12699 | 0.474013309 | 24196 | 0.903159779 |
| Long-2 | 4 KB | 26790387 | 12709 | 0.474386578 | 23091 | 0.861913641 |
| Long-2 | 8 KB | 26790387 | 12679 | 0.473266773 | 23163 | 0.864601172 |
| Long-2 | 16 KB | 26790387 | 12721 | 0.474834499 | 22996 | 0.858367593 |

## 2.2 Varying Perceptron Table Entries

We vary the number of table entries for perceptron to 128, 256, and 512. We consider the storage budget as 4KB and 8KB. From the relation given above, we calculate the history lengths as 31, 15, 7 for 4K storage budget and 63, 31 and 15 bits for 8K storage budget. The results are as follows.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trace | Storage Budget | History Length | Table Entries | Number of Instructions | Perceptron miss count | Perceptron MPKI |
| Long-1 | 4 KB | 31 | 128 | 19742295 | 92035 | 4.661818699 |
| Long-1 | 4 KB | 15 | 256 | 19742295 | 244460 | 12.38255228 |
| Long-1 | 4 KB | 7 | 512 | 19742295 | 394343 | 19.97452677 |
| Long-1 | 8 KB | 63 | 128 | 19742295 | 91955 | 4.657766486 |
| Long-1 | 8 KB | 31 | 256 | 19742295 | 92106 | 4.665415039 |
| Long-1 | 8 KB | 15 | 512 | 19742295 | 244418 | 12.38042487 |
|  |  |  |  |  |  |  |
| Long-2 | 4 KB | 31 | 128 | 26790387 | 23112 | 0.862697504 |
| Long-2 | 4 KB | 15 | 256 | 26790387 | 23160 | 0.864489192 |
| Long-2 | 4 KB | 7 | 512 | 26790387 | 23091 | 0.861913641 |
| Long-2 | 8 KB | 63 | 128 | 26790387 | 23112 | 0.862697504 |
| Long-2 | 8 KB | 31 | 256 | 26790387 | 23024 | 0.859412744 |
| Long-2 | 8 KB | 15 | 512 | 26790387 | 23163 | 0.864601172 |

In the first trace, the MPKI is very high and it increases as we increase the number of entries in the table, the accuracy degrades. On the other hand, in case of second trace, MPKI value is very low. Also, MPKI decreases as we increase the number of table entries. This could be due to the fact that, the perceptron takes a large amount of time to learn. The training is complete in the second trace as the number of instructions in trace2 is more than those in trace 1 and hence, the accuracy increases.

## 2.3 Varying the threshold

The ideal value of threshold for training is floor function of (1.93 \* hist\_len + 14). We vary the threshold and run the traces for threshold values of 0, 13. Also, we use the ideal value of 27 for threshold which can be obtained when history length is 12. Thus, from the results, it can be concluded that, as we approach the ideal value of threshold, MPKI improves. But as we cross the ideal value and go beyond it, the accuracy decreases drastically.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trace | Threshold | Number of Instructions | Perceptron miss count | Perceptron MPKI |
| Long-1 | 0 | 19742295 | 394343 | 19.97452677 |
| Long-1 | 13 | 19742295 | 314911 | 15.9510837 |
| Long-1 | 27 | 19742295 | 268612 | 13.60591562 |
| Long-1 | 39 | 19742295 | 279580 | 14.161474135 |
| Long-2 | 0 | 26790387 | 23091 | 0.861913641 |
| Long-2 | 13 | 26790387 | 12108 | 0.451953158 |
| Long-2 | 27 | 26790387 | 12152 | 0.453595538 |
| Long-2 | 39 | 26790387 | 12193 | 4.478456176 |