CMPEN 431

Project 1

Kyle Ostrowski and Sarah Tickner

# Provided Framework

The provided framework and its components enable design space exploration by easily allowing us to change cache sizes and subsequent latencies. Although adjusting every parameter and running tests on it is not possible because of timing constraints, the heuristic we used allowed us to look at different parameters to find a good design by testing them in a consistent way.

# Chosen Design Point

Our Penn State IDs are 907056774 and 931701543 which added together and mod by 24 is 21 so the order in which we adjusted the different configurations was Floating Point Unit, Cache, Core, and finally Branch Predictor. Using this order of exploration, the design space chosen by our DSE for performance was configuration 0 0 2 2 0 6 0 2 3 1 0 0 4 2 3 1 5 4 with the best EDP=3.91484e-08 and the best time=0.000194057. The design space chosen by our DSE for energy was configuration 0 0 2 2 0 5 0 1 3 1 0 0 3 3 2 1 4 3 with the best EDP=3.84011e-08. and the best Time=0.000197122.

# Plots

Plot A took the data output by the execution time and energy efficiency logs and plotted the Normalized Geometric Mean Execution Time (NGET) for each design space that was explored. The execution time data is shown in orange while the energy efficiency data is shown in blue. The energy efficiency consistently shows lower execution time.

Plot B shows the normalized geometric mean of the Energy-Delay product for each design space that was explored. Again this graph pulled data from both the execution time and energy efficiency logs. Again the energy efficiency data has a consistently lower EDP.

Plot C shows the normalized execution time compared to the benchmark for the final chosen design to optimize the performance.

Plot D shows the normalized EDP to the benchmark for the final design to optimize EDP.

# Table

|  |  |  |
| --- | --- | --- |
| Parameter | Performance | EDP |
| Width | Value = 0  Why = | Value = 0  Why = |
| Scheduling | Value = 0  Why = | Value = 0  Why = |
| L1block | Value = 2  Why = | Value = 2  Why = |
| Dl1sets | Value = 2  Why = | Value = 2  Why = |
| Dl1assoc | Value = 0  Why = | Value = 0  Why = |
| Il1sets | Value = 6  Why = | Value = 5  Why = |
| Il1assoc | Value = 0  Why = | Value = 0  Why = |
| Ul2sets | Value = 2  Why = | Value = 1  Why = |
| Ul2block | Value = 3  Why = | Value = 3  Why = |
| Ul2assoc | Value = 1  Why = | Value = 1  Why = |
| Replacepolicy | Value = 0  Why = | Value = 0  Why = |
| Fpwidth | Value = 0  Why = | Value = 0  Why = |
| branchsettings | Value = 4  Why = | Value = 3  Why = |
| Ras (return address stack) | Value = 2  Why = | Value = 3  Why = |
| Btb (branch target buffer) | Value = 3  Why = | Value = 2  Why = |
| Dl1lat | Value = 1  Why = | Value = 1  Why = |
| Il1lat | Value = 5  Why = | Value = 4  Why = |
| Ul2lat | Value = 4  Why = | Value = 3  Why = |

# More Sophisticated Heuristic

A better heuristic to more efficiently explore the design space might be

* Increase each value until it does not make it better and then stop
  + Instead of testing each one in each section
* Btb
  + For both performance and energy this was the most influencial for the timing

# Insights

Some insights we gained while working on this project is how long it can take to run these tests. When designing hardware, these are important considerations to creating a reliable and consistent

# Resources

Resources we used are

* I just want to find some stuff here because it will look more like we kinda know what we are doing

# Closing Comments

The code for our design space exploration was correct, but due to some instability in the simulation software, it yielded some odd results.