Chase McFarlane & Andre Mitrik

CMPEN 431 – Project 1 Report

1. **Describe in 100 words or less how the provided framework and its components enable a design space exploration.**

SimpleScalar provides a model allowing virtual testing of components (e.g., CPU, Caches, and Memory Hierarchies) within an architecture to maximize performance based on criteria. Design space exploration is facilitated by given constraints about component parameters (e.g., sizes and associativity for caches) being used in conjunction with logical decision parameters (e.g., scheduling format and replacement policies) to determine the workflow for an optimized framework. These parameters are tested and calculated in a defined exploration order to fit specified criteria (performance vs energy optimization). Ultimately, the goal is to maximize the efficiency ratio of cache sizes to latency.

1. **List the design points chosen by your DSE.**

\*DPI = Design Point Index

|  |  |  |
| --- | --- | --- |
| **Parameter** | **\*DPI** | **Value & Description** |
| Width | 0 | 1 |
| Scheduling | 0 | “-issue:inorder true -issue: wrongpath false” |
| L1 Block Size | 2 | 32 |
| L1 Data Sets | 2 | 128 |
| L1 Data Associativity | 0 | 1 |
| L1 Instruction Sets | 6 | 2048 |
| L1 Instruction Associativity | 0 | 1 |
| Unified L2 Sets | 2 | 1024 |
| Unified L2 Block Size | 3 | 128 |
| Unified L2 Associativity | 1 | 2 |
| Cache & TLB Replacement Policy | 0 | “l” (LRU) |
| Floating Point Unit Width | 0 | 1 |
| Branch Predictor Choice | 4 | “-bpred comb -bpred:comb 1024” |
| Return Address Stack Size | 3 | 8 |
| Branch Target Buffer | 0 | 128 16 <number of sets> <associativity> |
| L1 D$ Latency | 1 | 2 |
| L1 I$ Latency | 5 | 6 |
| Unified L2 Latency | 4 | 9 |

\*DPI = Design Point Index

|  |  |  |
| --- | --- | --- |
| **Parameter** | **\*DPI** | **Value & Description** |
| Width | 0 | 1 |
| Scheduling | 0 | “-issue:inorder true -issue: wrongpath false” |
| L1 Block Size | 2 | 32 |
| L1 Data Sets | 2 | 128 |
| L1 Data Associativity | 0 | 1 |
| L1 Instruction Sets | 5 | 1024 |
| L1 Instruction Associativity | 0 | 1 |
| Unified L2 Sets | 1 | 512 |
| Unified L2 Block Size | 3 | 128 |
| Unified L2 Associativity | 1 | 2 |
| Cache & TLB Replacement Policy | 0 | “l” (LRU) |
| Floating Point Unit Width | 0 | 1 |
| Branch Predictor Choice | 4 | “-bpred comb -bpred:comb 1024” |
| Return Address Stack Size | 3 | 8 |
| Branch Target Buffer | 0 | 128 16 <number of sets> <associativity> |
| L1 D$ Latency | 1 | 2 |
| L1 I$ Latency | 4 | 5 |
| Unified L2 Latency | 3 | 8 |

1. **Fill out the following table as detailed below.**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Performance** | **EDP** |
| Width | Value = 0  Why = | Value = 0  Why = |
| Scheduling | Value = 0  Why = | Value = 0  Why = |
| L1 Block Size | Value = 2  Why = | Value = 2  Why = |
| L1 Data Sets | Value = 2  Why = | Value = 2  Why = |
| L1 Data Associativity | Value = 0  Why = | Value = 0  Why = |
| L1 Instruction Sets | Value = 6  Why = | Value = 5  Why = |
| L1 Instruction Associativity | Value = 0  Why = | Value = 0  Why = |
| Unified L2 Sets | Value = 2  Why = | Value = 1  Why = |
| Unified L2 Block Size | Value = 3  Why = | Value = 3  Why = |
| Unified L2 Associativity | Value = 1  Why = | Value = 1  Why = |
| Cache & TLB Replacement Policy | Value = 0  Why = | Value = 0  Why = |
| Floating Point Unit Width | Value = 0  Why = | Value = 0  Why = |
| Branch Predictor Choice | Value = 4  Why = | Value = 4  Why = |
| Return Address Stack Size | Value = 3  Why = | Value = 3  Why = |
| Branch Target Buffer | Value = 0  Why = | Value = 0  Why = |
| L1 D$ Latency | Value = 1  Why = | Value = 1  Why = |
| L1 I$ Latency | Value = 5  Why = | Value = 4  Why = |
| Unified L2 Latency | Value = 4  Why = | Value = 3  Why = |

1. **Plots as defined below.**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

1. **Describe a more sophisticated heuristic which you expect will perform design space exploration (limited by 1000 design points) more effectively to find a better performing design (with respect to execution time).**
2. **Elaborate on any 2 new insights you gained while working on this project.**

SimpleScalar is a simulator for computer architectures that can emulate the performance of an architecture without having to build the physical chips or components. This virtualization facilitates the host machine to keep operating under its current architecture (e.g., x86) while simulating another ISA (e.g., MIPS). It can be similarly compared to how virtual machines allow different operating systems to run on top of another unrelated operating system. Given this description of the overbearing architecture, two insights our group gained working on this project include:

1. The importance of virtualized simulator hardware components to save on R&D costs and decrease dangers.
2. How to develop an effective way to approach a potential architecture’s component design given a list of input parameters/constraints. Additionally, valuable insight was provided depicting the importance of the sequence of how these constraints are used and the effect the sequence has on the final output.
3. **List of additional resources used (optional).**

* TA Office Hours
* C++ math library (for log2 function)
* https://www.geeksforgeeks.org/c-plus-plus/

1. **Additional information or comments (optional)**

N/A