Computer Science and Engineering 431
Computer Architecture
Spring 2018

Course Project

Distributed: April 12, 2018 **Due:** May 2nd, 2018 **Points**: 65

In this final project you are going to use SimpleScalar as the evaluation engine to perform a design space exploration, using the provided framework code, over an 18-dimensional processor pipeline and memory hierarchy design space. You will use a 5-benchmark suite as the workload. **This is an individual project**, and submitted artifacts include your code implementations of stub functions within the provided framework, as well as a project report discussing your chosen heuristics for exploring the design space, as well as summarizing your findings and how they confirm or run counter to intuitions developed in the discussion of material in class and assigned readings.

All allowed configuration parameters for each dimension of the design space are described in the provided shell script wrapper for executing SimpleScalar. The shell script takes 18 integer arguments, one for each configuration dimension, which expresses the index of the selected parameter for each dimension. All reported results should be normalized against a baseline design with configuration parameters: 0, 0, 0, 0, 0, 0, 5, 0, 5, 0, 2, 2, 2, 3, 0, 0, 3, 0. Note that not all possible parameter settings represent a valid combination. One of your tasks will be to write a configuration validation function based upon restrictions described later in this document.

Your assignment is to traverse the design space in order to select the best performing design under a set of different optimization functions. These include "best" performing overall design (in term of the geometric mean of normalized execution time normalized across all benchmarks), and the most energy-efficient design (as measured by the lowest geometric mean of normalized energy-delay product [units of energy delay product are joule-seconds] across all benchmarks). **In all there are 2 "best" designs to produce, 1 for performance and 1 for efficiency.**

Modeling Considerations:

The IC (instruction count) for each benchmark is a constant. Thus, for performance, you will be trying to optimize `sim_IPC` and the clock cycle time. We will use the following very simplistic model for clock cycle time: The clock cycle time is determined by the fetch width and whether the machine is in-order, or dynamic as follows:

Dynamic, fetch width = 1 means a 115 ps clock cycle
Static, fetch width = 1 means a 100 ps clock cycle
Dynamic, fetch width = 2 means a 125 ps clock cycle
Static, fetch width = 2 means a 120 ps clock cycle
Dynamic, fetch width = 4 means a 150 ps clock cycle
Static, fetch width = 4 means a 140 ps clock cycle
Dynamic, fetch width = 8 means a 175 ps clock cycle
Static, fetch width = 8 means a 165 ps clock cycle

Power and energy settings are as follows:

Leakage power:

Dynamic, fetch width = 1 → 1.5 mW
Static, fetch width = 1 → 1mW
Dynamic, fetch width = 2 → 2 mW
Static, fetch width = 2 → 1.5 mW
Dynamic, fetch width = 4 → 8 mW
Static, fetch width = 4 → 7 mW
Dynamic, fetch width = 8 → 32 mW
Static, fetch width = 8 → 30 mW

Cache and memory access energy, leakage/refresh power
(don't forget instruction fetch when calculating access energy!):

8KB → 20pJ, 0.125mW
16KB → 28pJ, 0.25mW
32KB → 40pJ, 0.5mW
64KB → 56pJ, 1mW
128KB → 80pJ, 2mW
256KB → 112pJ, 4mW
512KB → 160pJ, 8mW
1024KB → 224pJ, 16mW
2048KB → 360pJ, 32mW
Main Memory → 2nJ, 512mW

Energy per committed instruction:

Dynamic, fetch width = 1 → 10pJ
Static, fetch width = 1 → 8pJ
Dynamic, fetch width = 2 → 12pJ
Static, fetch width = 2 → 10pJ
Dynamic, fetch width = 4 → 18pJ
Static, fetch width = 4 → 14pJ
Dynamic, fetch width = 8 → 27pJ
Static, fetch width = 8 → 20pJ

Other setting constraints (for validation) are as follows:

1. The il1 block size must match the ifq size (e.g., for the baseline machine the ifqsize is set to 1 word (8B) then the il1 block size is also set to 8B). The dl1 should have the same block size as your il1.
2. The ul2 block size must be at least twice your il1 (dl1) block size with a maximum block size of 128B. Your ul2 must be at least as large as il1+dl1 in order to be inclusive.
3. The il1 sizes and il1 latencies are linked as follows (the same linkages hold for the dl1 size and dl1 latency):
   il1 = 8 KB (baseline, minimum size) means il1lat = 1
   il1 = 16 KB means il1lat = 2
   il1 = 32 KB means il1lat = 3
   il1 = 64 KB (maximum size) means il1lat = 4
   The above are for direct mapped caches. For 2-way set associative add an additional cycle of latency to each of the above; for 4-way (maximum) add two additional cycles.
4. The ul2 sizes and ul2 latencies are linked as follows:
   ul2 = 128 KB (minimum size) means ul2lat = 7
   ul2 = 256 KB (baseline) means ul2lat = 8
   ul2 = 512 KB means ul2 lat = 9
   ul2 = 1024 KB (1 MB) means ul2lat = 10
   ul2 = 2 MB (maximum size) means ul2lat = 11
   The above are for 4-way set associative caches. For 8-way set associative add one additional cycle of latency to each of the above; for 16-way (maximum) add two additional cycles; for 2-way set associative subtract 1 cycle; for direct mapped subtract 2 cycles.

5. Miscellaneous

mplat is fixed at 3
fetch:speed ratio of no more than 2
ifqsize can be set to a maximum of 8 words (64B)
decode:width and issue:width equal to your fetch:ifqsize
mem:width is fixed at 8B (memory bus width)
memport can be set to a maximum of 2
mem:lat is fixed at 51 + 7 cycles for 8 word
tlb:lat is fixed at 30, maximum tlb size of 512 entries for a 4-way set associative tlb
ruu:size maximum of 128 (assume must be a power of two)
lsq:size maximum of 32 (assume must be a power of two)

The framework code will evaluate a fixed number of design points per run. **This parameter cannot be changed.** The key part of your task in this project is to develop the (pair of/parameterized) heuristic search function(s) that select(s) the next design point to consider, given either a performance, or an energy efficiency goal. The framework code must be run once for each of the two optimization function options.

Your report for the course Project is due (submitted through the Canvas dropbox) at 11:59pm on May 2nd. PDF only.

The report should be a minimum of 4, maximum of 10, typed, single space pages in 12 point font for text and tables. The report should include, at minimum, the following four plots: 1 line plot of the normalized geomean execution time (y axis) for each considered design point vs. number of designs considered (x axis); 1 line plot of the normalized geomean of energy-delay product (y axis) vs number of designs considered; 1 bar chart showing normalized per-benchmark execution time and geomean normalized execution time for the best performing design; 1 bar chart showing per-benchmark normalized energy-delay product and geomean normalized energy delay product for the most energy-efficient design found. You will not be penalized for not finding the absolute best design points. However, you must produce a reasonable design space sequencing heuristic, and defend/explain both the heuristic and its results in your report (i.e. if you claim that the highest performing configuration is the one with minimum resources in all dimensions, and list "magic" as the reason, you will not receive full points). Points will also be assigned for following the guidelines and adhering to appropriate levels of clarity, and style (and spelling, grammar, etc.) for a technical document (e.g. where possible, avoid tangential discourse on whether a task was difficult or whether a result was, for instance, exciting – focus on analysis within the scope of the design space explored, how past information was used to generate the next configuration, the reasons that particular configuration combinations are sensible, etc.).