CS4223: Multi-Core Architectures
Assignment 1 (20 marks)
Deadline: Friday, September 8th 2023, 11:59pm

The goal of this assignment is to identify ILP exploitable through out-of-order superscalar processor architectures and explore the area-performance design tradeoff. This is individual assignment. You are encouraged to discuss your questions in the discussion forum.

The second assignment (mini-project) will be in groups of 2 students each (Deadline Week 13, 11th of November). You may want to select your partners early.

Infrastructure:

You can use an account on the Compute Cluster on run your experiments (a Linux-based Docker image or VM will also work). You should have automatically received SoC account when you registered for this module. If you do not have SoC account, you can enable account access here https://dochub.comp.nus.edu.sg/cf/guides/compute-cluster/enable-disable-access. If there are issues, please contact Technical helpdesk in Level 1 COM1. Next you need to activate "SoC Compute Cluster" access through MySoC. Please follow the instructions below to access the cluster.

https://dochub.comp.nus.edu.sg/cf/guides/compute-cluster/access

Examples of cluster nodes are xcna0 - xcna15, xcnb0 - xcnb19, etc., all end in .comp.nus.edu.sg: xcnb0.comp.nus.edu.sg. Note that some machines might be reserved for specific classes, so trying a few machines to use might be needed.

You will use Simplescalar architectural simulator for your experiments. An architectural simulator models an architecture in software, executes your application and provides detailed timing information. Simplescalar simulator you will be using can execute binaries in PISA instruction set (which is very similar to MIPS and other RISC instruction sets) and can model in-order as well as out-of-order processor cores with detailed cache and branch prediction mechanisms.
http://www.simplescalar.com/

For your convenience SimpleScalar has been installed on the cluster. Go to /home/course/cs4223/assignments/assignment1/simplesim-3.0
You will be using sim-safe, sim-profile and sim-outorder simulators from this directory. Feel free to copy the applications to your home directory if desired.

You will use three SPEC95 benchmarks (compress, go, and gcc; binaries are compress95.ss, go.ss and cc1.ss) for this experiment. The pre-compiled

benchmarks for PISA instruction set are available from /home/course/cs4223/assignments/assignment1/benchmarks
Feel free to copy this entire directory to your own home directory on the cluster. Also see the README file in the benchmarks directory to see how to run these files.

Before running SimpleScalar, please export environment variable as below:
$ export PATH=$PATH:/home/course/cs4223/assignments/assignment1/simplesim-3.0

First follow the instructions in the README file in CS4223 directory to run each benchmark using sim-safe simulator and check the correctness of the output. sim-safe is a functional simulator which simply executes the benchmark but does not model the micro-architecture and hence cannot provide any timing information.

The benchmark you choose depends on your student ID as follows:
Student ID modulo 3 = 0: Choose compress
Student ID modulo 3 = 1: Choose go
Student ID modulo 3 = 2: Choose gcc

To make the simulation time more fair between benchmarks, please first validate that your benchmark is running correctly (See the README in the benchmark directory to compare simulation output). When verified, one can use the first 80M instructions (SimpleScalar option -max:inst 80000000) to speed up development.

Design Problem:

When designing a processor you have to make trade-offs in order to achieve the best performance for a given cost. Choose one benchmark and optimize for this chosen benchmark your configuration of the pipeline functional units, pipeline width etc. using chip area as the cost of interest according to the table below.

Resource	Area	Values
Pipeline width W	1.8 x W	1-4
Integer ALU	2	1-4
Integer Multiplier	3	1-4
FP ALU	4	1-4
FP Multiplier	5	1-4
Out-of-order issue	1.4 x (above area)	Yes, No
Reorder Buffer (RUU)	0.5xRUU	8,16,32
Load/Store Queue (LSQ)	0.5xLSQ	4,8,16

Note that area estimates are gross approximations and that they do vary widely with the technology used to implement the chip. Using the table, you can calculate the area of your design by just adding the contributions from the units.

For example a design with a pipeline width of 2, 4 integer ALUs, 1 integer multiplier, 1 floating point ALU, 1 floating point multiplier would require an area of (2*1.8 + 4*2 + 3 + 4 + 5) = 23.6 for the first 5 rows. With out-of-order execution, 8 RUU and 4 LSQ would require 23.6 x 1.4 + 4 + 2 = 39.04 unit area. For in-order processor, the last three rows will contribute to zero area.

Clearly, if you choose the maximum value of each parameter you get the best performance (IPC). But the area requirement might be too high.

You will be now using sim-outorder simulators for the rest of this assignment. To understand the meaning of the different configuration choices please read SimpleScalar user-guide available from
http://www.simplescalar.com/docs/users_guide_v2.pdf

Use the default configuration for sim-outorder given in default.cfg file in CS4223 directory and modify the pipeline parameters. ***Note that you can use the sim-outorder simulator to model in-order processor by setting issue:inorder to true.***

(A) Design the **best performing architecture under the constraint that the area is less than or equal to 60.** Use simulations both to guide your design and verify your results. Note that simulating all possible reasonable configurations would take several hundreds of simulations, which is not realistic. You need a smart mechanism to navigate the design space.

You may want to run the sim-profile profiling simulator on your benchmark to find out the distribution of different instruction classes: load, store, unconditional branch, conditional branch, integer computation, and floating point computation. You can use this information in intelligently designing your architecture. Remember: "Common cases first and fast".

(B) Now assume that the power consumption for your architecture is proportional to the total area. For example, the power consumption for the processor with 39.04 unit area is 39.04 watt. Therefore, performance per watt for this architecture will be IPC/watt. **Design the architecture with best performance per watt value (higher is better) under the constraint that the area is less than or equal to 60.**

(C) After you get your optimal design point in terms of performance (as in Question (A)) you need configure the size of the L1 instruction and data caches. In the default.cfg you can see the parameter '–cache:dl1' and '–cache:il1' with the same configuration value '128:64:1:1'. (You could find the meaning of each number in the SimpleScalar user-guide.) Essentially the default configuration is 64-byte block size and 128 sets in a direct-mapped cache. We would like to keep the block size constant at 64-byte and associativity equal to 1, that is, direct-mapped cache. But our objective is to

coordinate L1 instruction cache size and L1 data cache size to improve the IPC. The choices for number of cache sets are given below in the table. Your goal is to find the optimal design point (best IPC) among these choices.

Choice number	1	2	3	4	5	6	7
L1 ICache # of sets	1024	512	256	128	64	32	16
L1 DCache # of sets	16	32	64	128	256	512	1024

Your report should not exceed 5 pages. The key points you should mention are the bottlenecks in the pipeline, how you identify them, and how you overcome them through your design choices. Please provide an introduction to this assignment, and an overview of the completed tasks and results as an introduction section (also include a short conclusion after you describe your results). Describe your optimal design point, the process you employed to reach this design point and the configuration of L1 cache. Note that your comprehension of the architecture and your interpretation of the experimental results for different design choices are far more important than reaching the "optimal" design point.

- Upload your report to Luminus Files / Submissions / Assignment 1 - ILP.

Grading Criteria:

1. Optimal Pipeline Configuration: 8 Points
2. Optimal Performance/Watt Configuration: 4 Points
3. Optimal Cache Configuration: 4 Points
4. Report and Analysis: 4 Points

SIMULATION TAKES TIME. START EARLY.

Runtime Examples (single run)
==> sim-safe-compress.time <==
real 8.29s

==> sim-profile-compress.time <==
real 8.26s

==> sim-outorder-compress.time <==
real 92.78s