HW #1: 32-bit ALU Design

Chun-Jen Tsai

NYCU

02/29/2024

HW 1: Design a 32-bit ALU

- Goal: design an ALU module using Verilog.
  - An Arithmetic and Logic Unit (ALU) is the circuit module of a CPU that performs calculations on registers
  - This ALU will become part of a mini processor for future HW
  - Verilator and GTKWave will be used as logic simulation tools

- The deadline of the HW is on 3/14, by 5:00pm.

The Port Definition of the ALU

- The ALU module is a combinational circuit with the I/O ports defined as follows:

```
module alu #(parameter DWIDTH = 32)
(
  input  [3 : 0]        op,   // Operation to perform.
  input  [DWIDTH-1 : 0] rs1,  // Input data #1.
  input  [DWIDTH-1 : 0] rs2,  // Input data #2.

  output [DWIDTH-1 : 0] rd,   // Result of computation.
  output zero,                // zero = 1 if rd is 0, 0 otherwise.
  output overflow             // overflow = 1 if overflow happens.
);
```

  - `op` triggers one of the six operations shown in the next slide
  - All output signals shall be set to `0` if `op` is an invalid bit pattern

Operation Table of the ALU

- You must implement 6 operations in the ALU

ALU operation	Function	op
`and`	rd ← rs1 and rs2	0000
`or`	rd ← rs1 or rs2	0001
`add`	Signed addition: rd ← rs1 + rs2	0010
`sub`	Singed subtraction: rd ← rs1 - rs2	0110
`nor`	rd ← ~(rs1 or rs2)	1100
`slt`	rd ← (rs1 < rs2)? 32'h1 : 32'h0;	0111

  - Any input bit patterns to the port `op` not listed in the table will be considered invalid.

Coding Comments

- You don’t have to implement the 32-bit adder and subtractor at gate-level. You can simply use the Verilog operators + and – to do the job

- For the implementation of the `overflow` flag, please refer to section 3.2 of the textbook
  - Note that integers are represented in 2’s complement here
  - Overflow conditions for additions and subtractions:

Operation	Operand A	Operand B	Result indicating overflow
$A + B$	≥ 0	≥ 0	< 0
$A + B$	< 0	< 0	≥ 0
$A - B$	≥ 0	< 0	< 0
$A - B$	< 0	≥ 0	≥ 0

Guide to the Simulation Tools

- For this course, we will use Verilator as the waveform simulator and GTKWave as the waveform viewer for digital circuit designs

- A user's guide on how to install and get started with the tools under Linux or Windows Subsystem for Linux (WSL) is available in the following link:

  https://hackmd.io/SQHK4lmSR1-Ob9V9Osrz5A

  - Note that the Verilator installed using the command:
    `$ sudo apt install verilator`
    under ubuntu may be too old, so you have to build your own Verilator from the source

HW 1 Grading Guide

- You should upload your `alu.v` to E3 by the deadline
- The port declaration of your ALU module must follow the specification in this HW precisely
- A sample testbench, `HW1_tb.tgz`, is available on E3
  - Please read the `readme.txt` file in the package carefully
- For grading, TAs will use a more thorough testbench to test your design
  - Correctness of `rd` account for 70%
  - Correctness of `zero` and `overflow` account for 30%