CS250 - Computer Organization and Architecture

RISC-V Processor in Verilog - HDL Assignment

Points to Note

- Several Verilog tutorials are available online. One of them is here: Verilog Playlist
- In the comments section of each HDL code file, include a short description of the module, your name, roll number, date of writing the program and other information you deem relevant.
- Your submission is an archive containing the following: One README file + One directory per ALU design. For both the ALU designs include the following: High level block diagram(s), Module level block diagrams, HDL code, snapshots of the simulator at key execution stages, and other relevant information.
- This is a team assignment. One submission per team. A team consists of up to 2 students.
- Each Q has a different submission date (all are midnight deadlines). Pack your report, code, screenshots and other files in an archive.

Design the following units in HDL for the RISC-V Processor.

1. *Deadline: 12-March.* **64-bit Integer ALU Design.** Design a 1-bit ALU. Extend to 64-bit ALU. The tasks to be supported are in the video lecture (and the textbook).

   Hint: For the block diagrams and design of the 64-bit Integer ALU, refer Section A.5 of *Patterson and Hennesy, Computer Organization and Design, RISC-V edition, MK.*

2. *Deadline: 12-March.* **ALU Control.** Design the ALU control for the above 64-bit Integer ALU. The control unit takes in 2 inputs:

   - A 2-bit input that distinguishes the nature of operation to perform – R-type (10), Memory Access (00), Condition evaluation for Control transfer (01).
   - A 6-bit `funct` input from the instruction encoding. The `funct` field decides which exact operation will be done by the ALU.

   Hint: This is solved in the video lectures. Also, check Section 4.4 in the textbook.

3. *Deadline: 19-March.* **Instruction Fetch Hardware** (Figs. 4.5 and 4.6, Pg. 244-245). Implement a combinational module that fetches the instruction corresponding to the address in the PC from the Instruction memory. The PC is updated to PC+4 in the next clock cycle. Implement the Instruction memory using addresses from the RISC-V memory layout (Fig. 2.13, Pg. 106). A Reset signal to this block resets the value of the PC to the first instruction to be fetched.

4. *Deadline: 28-March.* **ALU and ALU-Immediate Instructions.** Use the register file and ALU (datapath + control) and implement the datapath for an R-type, I-type RISC-V ALU instructions. The entire block is combinational. The results should be verified by the testbench by reading out the output register. The block diagram of the instruction datapath is in the video lecture. The control inputs can be hardwired into the muxes, and Register files.

5. *Deadline: 28-March.* **Load/Store Instructions, BEQ Instructions**. Implement the datapath for an load or store instruction execution. The datapath block diagram is in Fig 4.11, Pg. 250. The ALU operation input can be hardwired to the appropriate value (or can be fed from a constant register) such that the ALU performs effective address calculation at all times. The Data memory can be modeled after the RISC-V memory layout. Assume that Data memory accesses only the data portion of the memory layout. The control inputs can be hardwired into the muxes, and Register files.

6. *Deadline: 02-April.* The RISC-V Processor Pipeline Up to this point, you have designed the required modules for the pipeline. In this assignment your task is to integrate all these modules and build a working RISC-V pipeline (Fig. 4.17, Pg. 257). The control inputs will be output by the Control Unit (CU). The CU takes in the `opcode` and `funct` fields of the RISC-V instruction to output the corresponding control signals.

Testing the Pipeline

- Show the working of the pipeline by executing individual instructions. The instructions must be stored in the memory. Follow the memory layout of the RISC-V program.
- Demonstrate the working of the pipeline with a sequence of instructions. The sequence could be a sample, random set of instructions that contain atleast one instruction of each type. By the end of the sequence all the blocks of the pipeline are tasked.
- Demo the working of a simple, meaningful program. A small program such as the sum of the elements of an integer array should execute on your pipeline.