EEL 4783: Spring 2024

Implementing a Multi-Cycle Processor in Verilog

1. **Problem Description**:

   In this project, you will implement a simplified multi-cycle CPU using Verilog or systemverilog. The multi-cycle CPU should be able to perform arithmetic, memory, and control-related instructions. By completing this project, you will gain hands-on experience in building a fundamental component of modern computer systems.

2. **Instruction Set Architecture**

   Memory Instructions:

31-29	28-24	23-16	15-0
opcode	reg_addr_0	-	Addr

   2.1. lw: Loads a 32-bit value from memory and writes it in register
   regFile[reg_addr_0] = memory[Addr]

   2.2. sw: Stored a 32-bit value from register file and writes it to memory
   memory[Addr] = regFile[reg_addr_0]

   Control Instructions:

31-29	28-24	23-19	18-16	14-0
opcode	reg_addr_0	reg_addr_1	-	Addr

   2.3. beq: Changes program counter (PC) when register values are equal
   if(regFile[reg_addr_0] == regFile[reg_addr_1]):
   PC = Addr

   2.4. blt: Changes PC when a register value is less than the other
   if(regFile[reg_addr_0] < regFile[reg_addr_1]):
   PC = Addr

   Arithmetic Instructions:

31-29	28-24	23-19	18-14	13-0
opcode	reg_addr_0	reg_addr_1	reg_addr_2	-

2.5. add: Perform addition operation

regFile[reg_addr_2] = regFile[reg_addr_0] + regFile[reg_addr_1]

2.6. sub: Perform subtraction operation

regFile[reg_addr_2] = regFile[reg_addr_0] - regFile[reg_addr_1]

2.7. and: Perform bitwise AND operation

regFile[reg_addr_2] = regFile[reg_addr_0] & regFile[reg_addr_1]

2.8. or: Perform bitwise OR operation

regFile[reg_addr_2] = regFile[reg_addr_0] | regFile[reg_addr_1]

3. CPU Components:

Write a verilog module for each of the CPU components. Implement the decoder and ALU with combination logic. Implement the memory and register file using sequential logic.

3.1. Memory (Instruction, Data):

The memory module implements a 256x32 bits single read/write port RAM. The module takes in a 16-bit read address, 32-bit write data and a write enable signal as input. It also produces 32-bit read data as output.

3.2. Register File:

The register file implements a 16x32 bit dual read/ single write port RAM. The module takes in two 16-bit read addresses, one 16-bit write address, and a write enable signal as input. It produces two 32-bit read data as output.

3.3. Decoder:

The decoder modules take the 32-bit instruction as an input and splits the instruction into its constituent signals based on the ISA.

3.4. ALU:

The ALU module performs arithmetic operations based on the opcode. It takes in two 32-bit input operands and produces a single 32-bit output.

4. **CPU execution stages**:

Write a CPU top module instantiating all the CPU components. You will write a state machine that goes through each of the 4 stages of CPU execution.

4.1. Fetch: During the fetch stage, a new instruction is read from the instruction memory and the program counter is incremented.

4.2. Decode: In the decode stage, the instruction is decoded, and signals are established to assist its execution. At the same time, the register file is read to supply the operands to the ALU

4.3. Execute: Here, all the arithmetic operations are performed, and data is stored back to the register file. For a branch operation, the PC is changed

4.4. Memory: During this stage, the register file and data memory are accessed to perform load or store operations

5. **Testing plan**:

   Write a testbench for each of the 4 CPU components described in Section 3. The testbench must test the entire functionality of each of the CPU components.

   Write a testbench for the entire CPU going through each of the eight instructions.

6. **What to Submit on Webcourses:**
   - All the source code and testbench files
   - Report:
     - Show the simulation waveform of each individual module (section 3) along with explanation describing its functionality based on the waveform.
     - Show the simulation of entire CPU for each of the eight instructions on example instructions.
     - Write down some of the challenges and how you overcame them during the project.
7. **Grading Policy**:

   The grading will be done using the following rubric

Points (100)	**Task**
25	Design source files of CPU
25	Testbench source files of CPU.
25	Report showing simulation waveforms
25	Fully functioning CPU

8. **Additional Help**:
   Reach out to the TA (Sanjay) for any questions

Email: sanjay.gandham@ucf.edu
TA office hours: Tu 3-4pm, Th 3-4pm and by email appointment

9. **References:**
    - https://github.com/GandhamSanjay/EEL4783-Example-Projects
    - https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec07.pdf