ECE 201 Spring 2022 - Project 2

Assigned by Prof. Xiaochen Guo

Due Mar. 3rd, 2022 23:59pm

In this project, you will design and test a datapath that includes an ALU and a register file. This project is broken down into three parts.

- Please finish one by one in sequence.
- Keep all your source codes in the workstation, and include your running directory in the submitted report (e.g. **/proj/ece201-spring2022/XXX/Proj2)**. TA and instructors will go to your running directory and conduct simulations for grading purposes.
- Submit a project report (pdf format, one file including all parts) through coursesite.

1. **(20 points)** Implement and test a two-read-port one-write-port Register file with 32 64-bit entries (sequential logic). The Verilog code has been given on the textbook Figure A.8.11. However, there is a typo (not a critical bug, the code still works without fixing the typo). Please first identify and fix the typo and then write a Verilog testbench to test the code. Add a $readmemh system call in the register file module to initialize the content to non-zero values from a file. The stimulus in the testbench should first use the two read ports (one read port reads all odd entries, the other reads all even entries) to read each location of the register file in 16 cycles, and then clear the register file to all zeros in 32 cycles through the write port, then read each location of the register file again in 16 cycles.

In the project report, please include (1) the corrected line of code for the register file; (2) the testbench code; (3) the waveform of the simulation result.

2. **(40 points)** Design, implement, and test a 64-bit ALU (combinational logic). Design an ALU by writing **behavioral** Verilog. The ALU should be able to execute the following functions:

AND, ADD, ADDS, EOR, SUB, SUBS, LSL, ORR.

Function	ALUctl	Function	ALUctl
AND	0000	*SUB*	0110
ADD	0010	*SUBS*	0111
ADDS	0011	*LSL*	1001
EOR	1100	*ORR*	0001

The inputs of this ALU module should include ALUctl (listed above), two 64-bit source operands A and B, one 64-bit destination operand ALUOut, and four flags N, Z, V, C. An example of a simpler ALU design can be found on the textbook (Figure A.4.3). Write a testbench to test each function. You will decide the values of the inputs in the testbench. For ADDS and SUBS, use different combinations of inputs to test the four flags. For other functions, use one pair of inputs for each function.

In the project report, please include (1) the Verilog code of the ALU design; (2) a list of cases to be tested; (3) the testbench code; (4) the waveform of the simulation result.

3. **(40 points)** Design, implement, and test a datapath that combines the above register file and ALU to execute the following R-type instructions:

AND, ADD, ORR, ADDS, EOR, SUB, LSL, SUBS.

Please use **hierarchical design** and instantiate the above modules in the datapath. The input of this module should include a 32-bit instruction, clock; and the output of this module should include the four flags N, Z, V, C. The register file should be initialized from a file. The results of execution should be written back to the register file. Write a testbench to test the following instructions in sequence and print the values of flags after ADDS and SUBS. Print the computation result after executing each instruction. To validate your results, execute each instruction by hand (e.g., calculate bitwise AND of 0xFF and 0xF0F0 F0F0 F0F0 F0F0 to validate the result of the first instruction) and compare the results with print out.

Instruction	Opcode	Rm	Shamt	Rn	Rd
AND	10001010000	00001	000000	00010	01000
ADD	10001011000	00001	000000	00010	01001
ORR	10101010000	00000	000000	00010	01010
ADDS	10101011000	00000	000000	00011	01011
EOR	11001010000	00000	000000	00010	01100
SUB	11001011000	00101	000000	00100	01101
LSL	11010011011	00000	000011	00001	01110
SUBS	11101011000	00100	000000	00101	01111

The non-zero values of the initial register file are given below (all other registers are zeros).

X0	0xFFFF FFFF FFFF FFFF
X1	0xFF
X2	0xF0F0 F0F0 F0F0 F0F0
X3	0x8000 0000 0000 0000
X4	0x8
X5	0x1

In the project report, please include (1) the Verilog code of the top module (and any additional sub modules beyond the register file and ALU in the previous two parts); (2) the testbench code; (3) the values of the four flags after ADDS and SUBS are executed; (4) the computation result after each instruction is executed; and (5) a hand calculation for each instruction to validate the results.