Iru

ECE 351- Spring 2020
Homework #2

Submit your deliverables to your Homework #2 dropbox by 10:00 PM on Monday, 04-May-2020. We will only grade the submission with the latest date stamp. NO LATE ASSIGNMENTS WILL BE ACCEPTED AFTER NOON ON TUESDAY, 05-MAY-2020 BECAUSE I'D LIKE TO REVIEW THE SOLUTION IN CLASS BEFORE THE EXAM NEXT THURSDAY.

Source code should be structured and commented with meaningful variables. Include a header at the top of each file listing the author and a description. You may use the following template:

```
//////////////////////////////////////////////////////////////
// <filename>.sv - <one line description>
//
// Author:  <your name> (<your email address>)
// Date:    <date you created the code>
//
// Description:
// ------------
// <text description of what function the module performs>
///////////////////////////////////////////////////////////////
```

Files to submit:

- `hw2_prob1.sv` (your model for problem 1)
- `tb_hw2_prob1.sv` (your testbench for problem 1)
- `results_hw2_prob1.txt` (your transcript from a successful simulation of problem 1)
- `hw2_prob2.sv` (your model for problem 2)
- `tb_hw2_prob2.sv` (your testbench for problem 2)
- `results_hw2_prob2.txt` (your transcript from a successful simulation of problem 2)
- `hw2_prob3_alu.sv (your model for the ALU)`
- `hw_prob3_dut.sv` (your top level model including the ALU and register file)
- `results_hw2_prob3.txt` (your transcript from a successful simulation of problem 3)

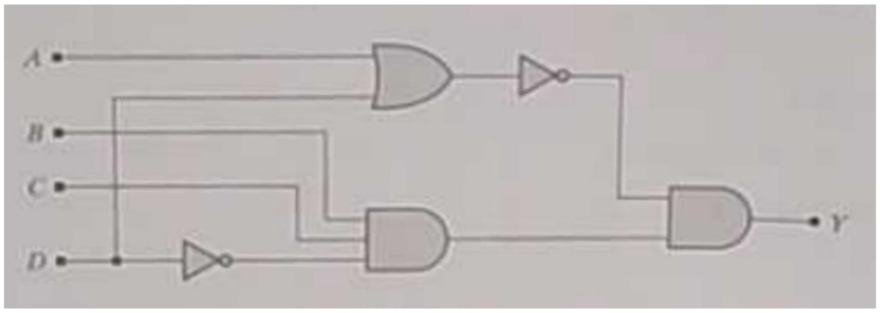
Note: the name of the file should match the name of the module. The file should have a .sv extension for a SystemVerilog source code file or package.

Create a single .zip or .rar file containing your source code files and transcripts showing that your implementations simulates correctly. Name the file `<yourname>_hw2.zip` (ex: rkravitz_hw2.zip).

Acknowledgement: These problems are based on an exercises from *Advanced Digital Design with the Verilog HDL: 2e* by Michael D. Ciletti, Pearson India Education Services Pvt. Ltd, 2017. Modifications for SystemVerilog designed by Roy Kravitz, 2020.

Problem 1 (25 pts)

A. (5 pts) Write a SystemVerilog module that implements the schematic below using continuous assignment(s). Note that there are no delays in the circuit.



The signature for this module is:

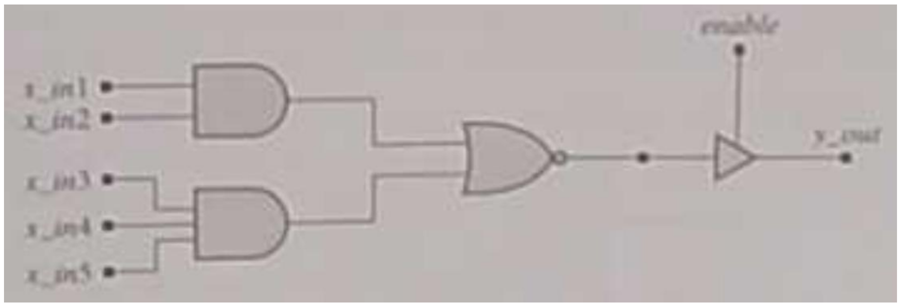
```
module hw2_prob1 (
   input  logic  A, B, C, D,
   output logic  Y
);
```

B. (10 pts) Write a testbench to verify the functionality of your model. The testbench for this circuit should set up a `$monitor()` statement in one initial block and generate all of the possible input combinations with a `#5 ns` delay between changing the inputs.

C. (10 pts) Simulate your model and your testbench using ModelSim or QuestaSim to demonstrate that your design is correct. Submit a transcript of a successful simulation

Problem 2 (25 pts)

A. (5 pts) Write a SystemVerilog model of the following circuit. Note that the buffer is a tri-state buffer. Model the combinational logic using an `always_comb` block and the tri-state buffer using continuous assign statement(s):



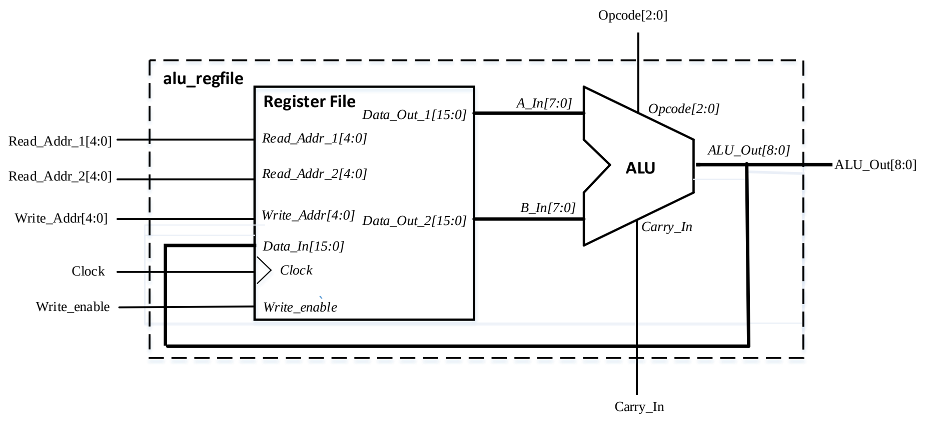
The name of your module should be `hw2_prob2`. You are the engineer for this project so we are leaving the signature of the module to you. Note: the inputs are `x_in1`..`x_in5`. The tri-state output is `y_out`.

B. (10 pts) Write a testbench to verify the functionality of your model. The testbench for this circuit should set up a `$monitor()` statement in one initial block and generate all of the possible input combinations with a `#5 ns` delay between changing the inputs. Be sure to check that `y_out` is `‘z` whenever enable = 0. *Hint: Declare* `y_out` *as a tri s that it is capable of be driven by more than one source.*

C. (10 pts) Simulate your model and your testbench using ModelSim or QuestaSim to demonstrate that your design is correct. Submit a transcript of a successful simulation.

Problem 3 (50 pts)

For this homework problem we are going to create a SystemVerilog model for a simple digital system the includes an ALU (Arithmetic Logic Unit) with inputs taken from a 32 entry x 16 bit wide Register File and the results written back to the Register file.   A block diagram of the system is shown below:



A. (20 pts) Write a SystemVerilog module that implements an 8-bit ALU. The ALU is a block of combinational logic that implements the following functionality (ex: ADD Opcode = 3'b000, EXNOR Opcode = 3'b111):

Opcode	Operation	Function
3'b000	ADD	a + b + c_in
3'b001	SUBTRACT	a + ~b + c_in
3'b010	SUBTRACT_a	a + ~a + ~c_in
3'b011	OR_ab	{1'b0, a | b}
3'b100	AND_ab	{1'b0, a & b}
3'b101	NOT_ab	{1'b0, (~a) & b}
3'b110	EXOR	{1'b0, a ^ b}
3'b111	EXNOR	{1'b0, a ~^ b}

Make use of SystemVerilog constructs as appropriate. The signature for the module is:

```
import ALU_REGFILE_defs::*;

module hw2_prob3_alu (
   input logic [ALU_INPUT_WIDTH-1:0]   A_In, B_In,  // A and B operands
   input logic                         Carry_In,    // Carry In
   input aluop_t                       Opcode,      // operation to perform
   output logic [ALU_OUTPUT_WIDTH-1:0] ALU_Out      // ALU result(extended by 1 bit
                                                    // to preserve Carry_Out from
                                                    // Sum/Diff)
);
```

Note that the ALU Output is one bit wider than the inputs. This is to preserve the Carry Out. Note also the use of a `typedef` enum for the ALU opcode (operation to perform) in the ALU_REGFILE_defs package.

B. (15 pts) Create a model of the DUT called hw2_prob3_dut that instantiates instances of your ALU module and the register file module provided in the release. Pad the extra bits from the ALU to the register file with 0's. The signature for the module is:

```
import ALU_REGFILE_defs::*;

module hw2_prob3_dut (

   // register file interface
   input  logic [REGFILE_ADDR_WIDTH-1:0]      Read_Addr_1, // read port addresses
                                              Read_Addr_2,
   input  logic [REGFILE_ADDR_WIDTH-1:0]      Write_Addr,  // write port address
   input  logic                               Write_enable,// write enable (1 to
                                                           // write)
   input  logic [REGFILE_WIDTH-1:0]           Write_data,  // data to write into the
                                                           // register file

   // ALU interface.  Data to the ALU comes from the register file
   input  logic                               Carry_In,    // Carry In
   input  aluop_t                             Opcode,      // operation to perform
   output logic [ALU_OUTPUT_WIDTH-1:0]        ALU_Out,     // ALU result

   // system-wide signals
   input  logic                               Clock        // system clock
);
```

C. (15 pts) Simulate your source code with QuestaSim using the provided testbench to demonstrate that your design is correct. Submit a transcript of a successful simulation. You will be graded on the correctness of your design.