Department of Electrical and Computer Engineering

*University of Wisconsin – Madison*

ECE 552 Introductions to Computer Architecture

Homework #5 (Due Wed. 2/28)

*For the question below (verilog based), paste your entire verilog code in a single text/doc file and submit the text/doc file on canvas. Note that these verilog questions need to follow the verilog rules specified in the 552 project/verilog rules document that can be found on Canvas.*

1. ***Register File***

You are required to design a register file with 16 registers, each 16-bits wide. The register file should have 2 read ports and one write port. The expected design is shown in the figure below:



* 1. From the figure, there are 2 4-bit source registers inputs and 1 4-bit destination register input to perform 2 reads and/or 1 write.
  2. The register file is made up of 16 registers and each register is made of 16 bit-cells.
  3. Each bit cell (as show above) consists of a D-Flip Flop and tri-state buffers. The D Flip-Flop is provided to you (refer DFF.v in the verilog section on canvas).
  4. The respective register information is decoded via the 4-16 decoders, which you are expected to implement using assign statement and operators (as opposed to an exhaustive case statement).
  5. Once a read register is decoded, the particular register i.e. a row of 16 bitcells, are enabled. The enabling is performed via the tri-state buffers (as shown in figure).
  6. Once a write register is decoded, the particular register (row of bitcells) is given the write enable and the DstData is written in.
  7. You are expected to design the register file to support internal bypassing. This means that your design should be able read the written value to a register in the same cycle as a write. Logic to perform bypassing is not shown in the figure.

The required modules are listed below:

module **ReadDecoder\_4\_16**(input [3:0] RegId, output [15:0] Wordline);

module **WriteDecoder\_4\_16**(input [3:0] RegId, input WriteReg, output [15:0] Wordline);

module **BitCell**( input clk,  input rst, input D, input WriteEnable, input ReadEnable1, input ReadEnable2, inout Bitline1, inout Bitline2);

module **Register**( input clk,  input rst, input [15:0] D, input WriteReg, input ReadEnable1, input Readenable2, inout [15:0] Bitline1, inout [15:0] Bitline2);

module **RegisterFile**(input clk, input rst, input [3:0] SrcReg1, input [3:0] SrcReg2, input [3:0] DstReg, input WriteReg, input [15:0] DstData, inout [15:0] SrcData1, inout [15:0] SrcData2);