**Lab 3A – Register File**

We want to simulate the Register File shown below (similar to Figure 4.7a on p.265). This has a collection of 32 registers (R0 to R31). Hence the address is 5 bits. Registers are 64 bits wide and so data in and data out are 64 bits.

RegWrite

clock

5

Read register 1

(Address)

64

5

Read

data1 (to A input of ALU)

Read

Register 2

(Address)

5

Write

register

(Address)

64

Read

Data2 (to B input of ALU)

Write

Data

64

The Verilog code for the module is given in Figure A.8.11 (p. A. 56) (beginning with the line “module registerfile ….”). It has some errors. The register file declaration should be “reg [63:0] RF **[0:31]**;” Note the change in the order of indices in the second set of parentheses. We do it when we want to declare *how many* registers of size [63:0] (first set of parentheses) are there. Also the always statement should be

“always @ (posedge clock)

If (RegWrite = = 1) RF [WriteReg] <= WriteData;”

Develop a Verilog Test Fixture. In the beginning in the test bench, you may initialize two registers by first writing to them. Choose Registers 5 and 10 and write to them 64’h5555555555555555 and 64’haaaaaaaaaaaaaaaa. Remember writing can take place only on the positive edge of the clock and so clock must be zero initially and then become 1 for *each* write operation. Also the register address and register data must be **set up** **prior** to the arrival of the clock edge. All changes must occur before the clock edge. Values must be stable before and after the clock edge (for what are called “set up time” and “hold time”). So you may make changes to address and data at 50 and let the clock edge occur at 100 and undo changes if necessary at 150 and so on. It is a good idea to *plan* and **draw a timing diagram** to indicate when some signal will go up and then come down with relation to others. The timing diagram will help a lot in writing the test module. Show your plan of timing diagram to the instructor at the time of demo.

In the test bench, after writing, we can read them and verify that they come out right. (Initially both data outputs will be all zeros since we are reading register 0 as Read address is 0 (in fact, all registers are also zero initially).

**Lab 3B – RF-ALU Combined**

Now we want to combine RF of Lab 3A with ALUwithControl of Lab 2C as shown below:

Read

data1

5

Read register 1

5

5

64

A

64

4

Zero

64

Read

Register 2

64

ALU result

Read

Data2

Write

register

(Address)

B

2

ALUOp

Write

Data

Opcode field

116

ALU operation

RegWrite

clock

Now two more signals will become ‘wire’. See how we “instantiated” the two modules in Lab 2C when we combined them. Plan and draw a timing diagram. Develop a Verilog Test Fixture to verify the operation. We need to write two register values and then read them out to ALU and with appropriate ALU operation developed one after another, ALU result should be correct.

Include ‘Wires’ A and B and ALU operation in the waveform.

mc, f16