Alex Oswald
EECE 2323
Due: 22 Sep 2020

Pre-Lab Assignment 4

Hi all,
For Prelab 4, please submit a single document which contains:
1) reg_file.v
2) reg_file_tb.v
3) Waveform of reg_file_tb.v
4) Create Test Vectors for the ALU-Regfile module in lab 4 (You do not need to code or simulate it, so please consider every necessary cases)

reg_file.v

```
1.  `timescale 1ns / 1ps
2.
3.  module reg_file(
4.      //inputs
5.      input rst, // rst=reset OR clr=clear
6.      input clk,
7.      input wr_en,
8.      input [1:0] rd0_addr,
9.      input [1:0] rd1_addr,
10.     input [1:0] wr_addr,
11.     input [8:0] wr_data, //piped from ALU
12.     //outputs
13.     output reg [8:0] rd0_data,
14.     output reg [8:0] rd1_data);
15.
16.     reg [8:0] mem[3:0];
17.     integer i;
18.
19.     always@(rst, negedge clk)
20.     begin
21.         if(rst)
22.             for (i=0; i<4; i=i+1)
23.                 mem [i] = 0;
24.         else if(wr_en)
25.             mem[wr_addr] = wr_data;
26.         else
27.           begin
28.             rd0_data = mem[rd0_addr];
29.             rd1_data = mem[rd1_addr];
30.           end
31.     end
32.
33. endmodule
```

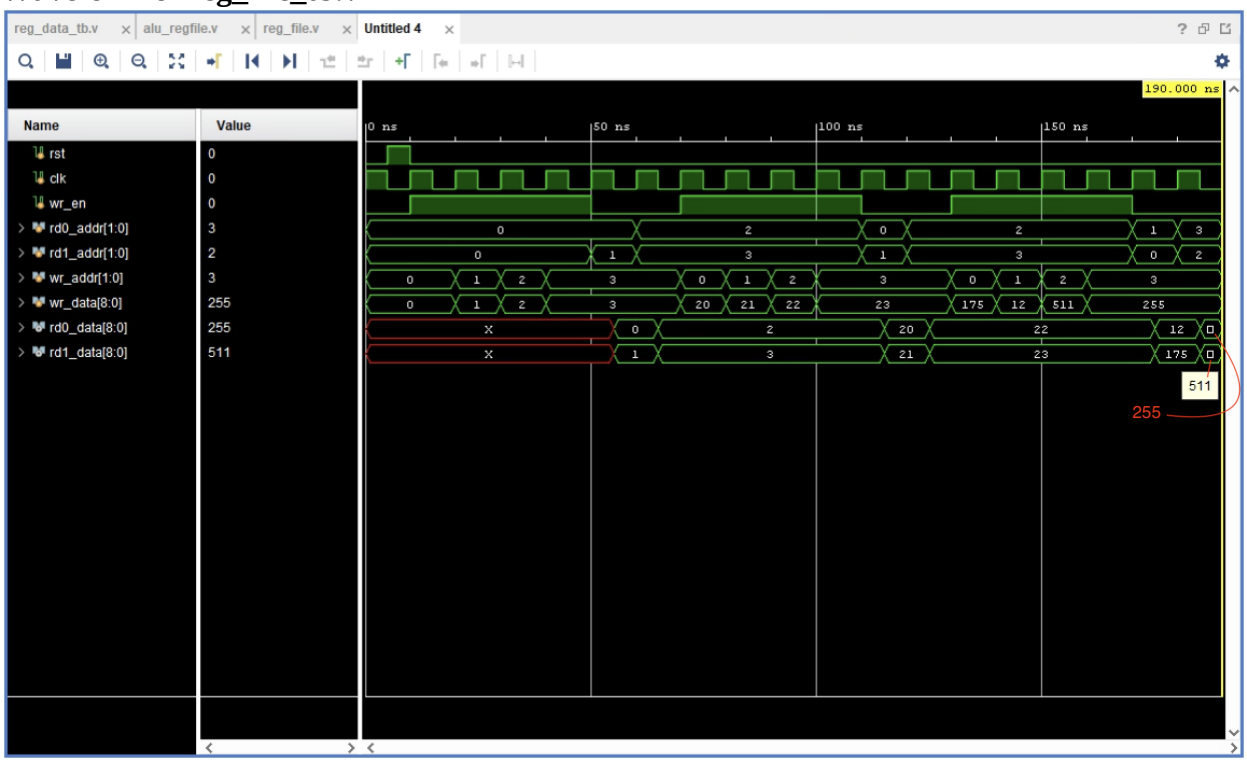
reg_file_tb.v

```
1.  `timescale 1ns / 1ps
2.
3.  module reg_file_tb();
```

```
    // Inputs
    reg rst, clk, wr_en;
    reg [1:0] rd0_addr, rd1_addr, wr_addr;
    reg [8:0] wr_data; //piped from alu
    // Outputs
    wire [8:0] rd0_data, rd1_data;

    // Initiate the Unit Under Test (UUT)
    reg_file uut (
        .rst(rst),
        .clk(clk),
        .wr_en(wr_en),
        .rd0_addr(rd0_addr),
        .rd1_addr(rd1_addr),
        .wr_addr(wr_addr),
        .wr_data(wr_data), //piped from ALU
        .rd0_data(rd0_data),
        .rd1_data(rd1_data));

    // Initialize Inputs (stimulus)
    initial
    // 1) Initial pulse for rst (clr?)
    begin
        #0
        rst = 0;
        #5
        rst = 1;
        #5
        rst = 0;
    end

    // 2) Clock signal
    initial
    begin
        clk  = 1;
        forever #5  clk = ~clk;
    end

    // 3) Values for other inputs
    initial
    begin
        $monitor("%d  ", $time, , " rst=%b ", rst, " clk=%b ", clk, " wr_en=%b ", wr_en
    , " rd0_addr=%d ", rd0_addr, " rd1_addr=%d ", rd1_addr, " wr_addr=%d ", wr_addr, " wr_d
    ata=%d ", wr_data, " rd0_data=%d ", rd0_data, " rd1_data=%d ", rd1_data);
        #0 // Initially
        rd0_addr = 0;
        rd1_addr = 0;
        wr_en = 0;
        wr_addr = 0;
        wr_data = 9'd0;

        // Test Things
        #10 wr_en = 1;  wr_addr = 0;    wr_data = 9'd0;
        #10 wr_en = 1;  wr_addr = 1;    wr_data = 9'd1;
        #10 wr_en = 1;  wr_addr = 2;    wr_data = 9'd2;
        #10 wr_en = 1;  wr_addr = 3;    wr_data = 9'd3;
        #10 wr_en = 0;  rd0_addr = 0;   rd1_addr = 1;
        #10 wr_en = 0;  rd0_addr = 2;   rd1_addr = 3;

        #10 wr_en = 1;  wr_addr = 0;    wr_data = 9'd20;
        #10 wr_en = 1;  wr_addr = 1;    wr_data = 9'd21;
```

```
63.         #10 wr_en = 1;  wr_addr = 2;    wr_data = 9'd22;
64.         #10 wr_en = 1;  wr_addr = 3;    wr_data = 9'd23;
65.         #10 wr_en = 0;  rd0_addr = 0;   rd1_addr = 1;
66.         #10 wr_en = 0;  rd0_addr = 2;   rd1_addr = 3;
67.
68.         #10 wr_en = 1;  wr_addr = 0;    wr_data = 9'd175;
69.         #10 wr_en = 1;  wr_addr = 1;    wr_data = 9'd12;
70.         #10 wr_en = 1;  wr_addr = 2;    wr_data = 9'd511;
71.         #10 wr_en = 1;  wr_addr = 3;    wr_data = 9'd255;
72.         #10 wr_en = 0;  rd0_addr = 1;   rd1_addr = 0;
73.         #10 wr_en = 0;  rd0_addr = 3;   rd1_addr = 2;
74.
75.         #10 $finish;
76.     end
77.
78. endmodule
```

Waveform of reg_file_tb.v



Test Vectors for the ALU-Regfile module in Lab 4

```
1.  ALUsrc1=0; ALUsrc2=0;  // ReadData1 && ReadData2
2.   ALUop = 3'b000; // a + b;
3.  #10 RegWrite = 1;   WriteAddr = 1;  WriteData = 9'd1;
4.  #10 RegWrite = 1;   WriteAddr = 2;  WriteData = 9'd2;
5.  #10 RegWrite = 1;   WriteAddr = 3;  WriteData = 9'd3;
6.  #10 RegWrite = 0;   ReadAddr1 = 0;  ReadAddr2 = 1;
7.  #10 RegWrite = 0;   ReadAddr1 = 2;  ReadAddr2 = 3;
8.
9.   ALUop = 3'b001; // ~b;
10. #10 RegWrite = 1;   WriteAddr = 0;  WriteData = 9'd20;
11. #10 RegWrite = 1;   WriteAddr = 1;  WriteData = 9'd21;
12. #10 RegWrite = 1;   WriteAddr = 2;  WriteData = 9'd22;
```

```
#10 RegWrite = 1;   WriteAddr = 3;  WriteData = 9'd23;
#10 RegWrite = 0;   ReadAddr1 = 0;  ReadAddr2 = 1;
#10 RegWrite = 0;   ReadAddr1 = 2;  ReadAddr2 = 3;

 ALUop = 3'b010; // a & b;
#10 RegWrite = 1;   WriteAddr = 0;  WriteData = 9'd175;
#10 RegWrite = 1;   WriteAddr = 1;  WriteData = 9'd12;
#10 RegWrite = 1;   WriteAddr = 2;  WriteData = 9'd511;
#10 RegWrite = 1;   WriteAddr = 3;  WriteData = 9'd255;
#10 RegWrite = 0;   ReadAddr1 = 1;  ReadAddr2 = 0;
#10 RegWrite = 0;   ReadAddr1 = 3;  ReadAddr2 = 2;

ALUsrc1=0; ALUsrc2=1; // ReadData1 && Instr_i
 ALUop = 3'b011; // a | b;
#10 RegWrite = 1;   WriteAddr = 0;  WriteData = 9'd399;
#10 RegWrite = 0;   ReadAddr1 = 0;  Instr_i = 8'd42;

 ALUop = 3'b100; // a >>> 1;
#10 RegWrite = 1;   WriteAddr = 1;  WriteData = 9'd425;
#10 RegWrite = 0;   ReadAddr1 = 1;  Instr_i = 8'd200;

 ALUop = 3'b101; // a << 1;
#10 RegWrite = 1;   WriteAddr = 2;  WriteData = 9'd1;
#10 RegWrite = 0;   ReadAddr1 = 2;  Instr_i = 8'd23;

 ALUop = 3'b110; // 0;
#10 RegWrite = 1;   WriteAddr = 3;  WriteData = 9'd1;
#10 RegWrite = 0;   ReadAddr1 = 3;  Instr_i = 8'd23;

#10 ALUsrc1=1; ALUsrc2=0; // zero_register && ReadData2
 ALUop = 3'b110; // 0;
#10 Instr_i = 8'd255;

 ALUop = 3'b111; // 0;
#10 Instr_i = 8'd255;

#10 $finish;
```