

Date: 11/17/2020

Q1. Circuit diagram for a register file with four 2-bit registers.

The rest of this lab is about a register file with eight 4-bit registers.

Q2. Write the Verilog code for a 4-bit 8-to-1 multiplexer below.

```
module Mux8_4b(S2, S1, S0, W0, W1, W2, W3, W4, W5, W6, W7, F);
    input S2, S1, S0;
    input [3:0] W0, W1, W2, W3, W4, W5, W6, W7;
    output [3:0] F;

    reg[3:0] F;

    always @(W0 or W1 or W2 or W3 or W4 or W5 or W6 or W7)

    begin
        case (S2, S1, S0)
            3'b000: F = w0;
            3'b001: F = w1;
            3'b010: F = w2;
            3'b011: F = w3;
            3'b100: F = w4;
            3'b101: F = w5;
            3'b110: F = w6;
            3'b111: F = w7;
        endcase
    end
end
```

Q3. Write the Verilog code for a 3-to-8 decoder in the space below.

```
module Decoder3to8(EN, W2, W1, W0, Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7);
    input EN, W2, W1, W0;
    output Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;

    always @ (EN, W2, W1, W0)

    begin
        reg Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;
        initial begin
            {Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7} = 8'b0;
        end

        if (EN == 1'b1)

        begin
            case
                3'b000: Y0 = 1;
                3'b001: Y1 = 1;
                3'b010: Y2 = 1;
                3'b011: Y3 = 1;
                3'b100: Y4 = 1;
                3'b101: Y5 = 1;
                3'b110: Y6 = 1;
                3'b111: Y7 = 1;
            endcase
        end
    end
endmodule
```

Q4. Using copies of the decoder, multiplexer, and 4-bit register from the previous steps, write the Verilog code that will provide the functionality of a register file with eight 4-bit registers. Your code should contain the decoder, the 4-bit registers, the multiplexers, and additional connections to make the whole circuit operational.

```
Module Dflip_4bit(Q, D, EN, RST, CLK);
    input CLK, RST, EN;
    input [3:0] D;

    output [3:0] Q;

    wire [3:0] W1;
endmodule

module regfile(DATAP3, DATAP2, DATAP1, DATAP0, DATAQ3, DATAQ2, DATAQ1, DATAQ0,
    RP2, RP1, RP0, RQ2, RQ1, RQ0, WA2, WA1, WA0, LD_DATA, WR, CLRN, CLK);

    // address and control ports
    input RP2, RP1, RP0, RQ2, RQ1, RQ0, WA2, WA1, WA0, WR, CLRN, CLK;

    // input data port
    input [3:0] LD_DATA;

    // output data ports
    output DATAP3, DATAP2, DATAP1, DATAP0, DATAQ3, DATAQ2, DATAQ1, DATAQ0;

    // wire declarations
    wire [3:0] DATAP, DATAQ;

    wire EN0, EN1, EN3, EN4, EN5, EN6, EN7;
    wire[3:0] DATA0, DATA1, DATA2, DATA3, DATA4, DATA5, DATA6, DATA7

    Decoder3to8 Decode (.EN(1'b1), .W1(WA2), .W0(WA1), .W0(WA0), .Y0(EN0),
        .Y1(EN1), .Y2(EN2), .Y3(EN3), .Y4(EN4), .Y5(EN5), .Y6(EN6), .Y7(EN7));

    Mux8_4b PMUX (.S2(RP2), .S1(RP1), .S0(RP0), .W0(DATA0), .W1(DATA1),
        .W2(DATA2), .W3(DATA3), .W4(DATA4), .W5(DATA5), .W6(DATA6), .W7(DATA7),
        .F(DATAP));

    Mux8_4b QMUX (.S2(RQ2), .S1(RQ1), .S0(RQ0), .W0(DATA0), .W1(DATA1),
        .W2(DATA2), .W3(DATA3), .W4(DATA4), .W5(DATA5), .W6(DATA6), .W7(DATA7),
        .F(DATAQ));

    assign DATAP0 = DATAP[0];
    assign DATAP1 = DATAP[1];
    assign DATAP2 = DATAP[2];
    assign DATAP3 = DATAP[3];

    assign DATAQ0 = DATAQ[0];
    assign DATAQ1 = DATAQ[1];
    assign DATAQ2 = DATAQ[2];
    assign DATAQ3 = DATAQ[3];
endmodule
```

Prelab TA Initials: _____

LAB:

Fill in the characteristic table for the one-bit parallel access register.

In	Load	Out
0	0	Prev Value of Out
1	0	Prev Value of Out
0	1	0
1	1	1

Fill in the table with steps that will load the registers as follows: Reg[0]=F, Reg[1]=A.
 Reg[2]=C. Reg[3]=E, Reg[4]=2, Reg[5]=7, Reg[6]=6, and Reg[7]=1.

LD_DATA	WA
1111	000
1110	001
1101	010
1100	011
1011	100
1010	101
1001	110
1000	111

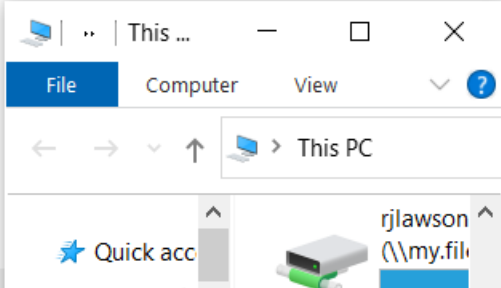
4.0 Register File with Eight 4-bit Registers

Screenshots:

```
module mux8_4b(S2, S1, S0, W0, W1, W2, W3, W4, W5, W6, W7, F)
    input S2, S1, S0;
    input [3:0] W0, W1, W2, W3, W4, W5, W6, W7;
    output [3:0] F;

    reg[3:0] F;

    always @(W0 or W1 or W2 or W3 or W4 or W5 or W6 or W7)
    begin
        case ({S2, S1, S0})
            3'b000 : F = W0;
            3'b001 : F = W1;
            3'b010 : F = W2;
            3'b011 : F = W3;
            3'b100 : F = W4;
            3'b101 : F = W5;
            3'b110 : F = W6;
            3'b111 : F = W7;
        endcase
    end
endmodule
```



Fill in the table below with the result produced by the register file (with CLRN=1).

LD_DATA	Sel	WA	RP	RQ	CTRL	WR	Effect
0110	0	111	111	111	0	1	Reg[7] ← 6
0011	0	110	110	111	0	1	Reg[6] ← 3
0010	0	101	101	110	1	1	Reg[5] ← 2
0100	0	100	100	101	1	1	Reg[4] ← 4
0101	0	011	011	100	0	1	Reg[3] ← 5
0001	0	010	010	011	0	1	Reg[2] ← 1
0111	0	001	001	010	1	1	Reg[1] ← 7
1000	0	000	000	001	1	1	Reg[0] ← -8 Register File Contents:

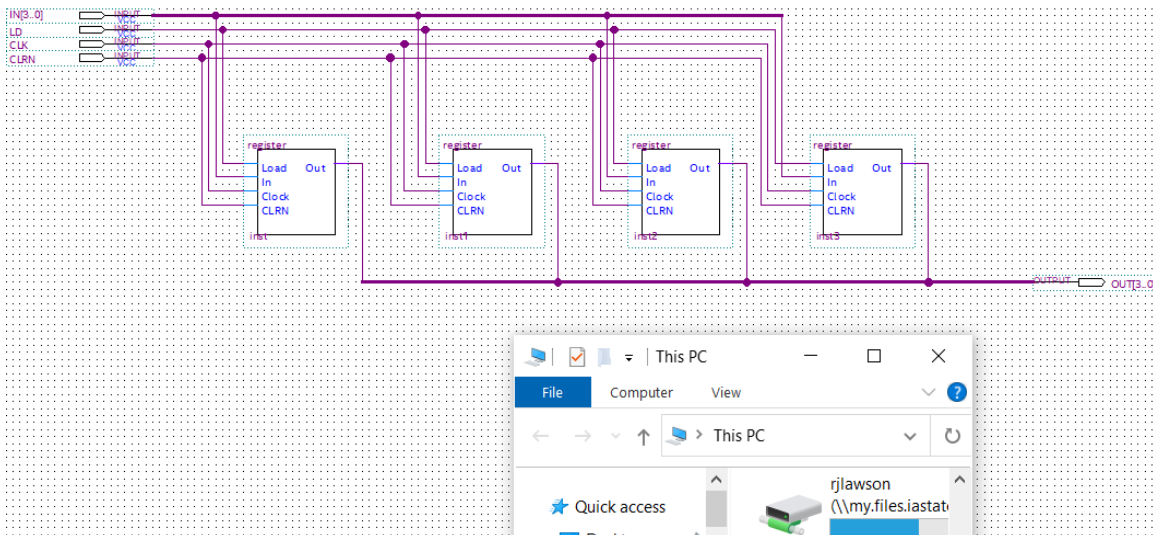
Lab 12 Answer Sheet

0000	1	001	000	001	0	1	Reg[1] \leftarrow Reg[0] + Reg[1] = -1
0001	1	000	010	011	0	1	Reg[0] \leftarrow Reg[2] + Reg[3] = 6
1111	1	010	100	101	1	1	Reg[2] \leftarrow Reg[4] + Reg[5] = 2
1001	1	101	110	111	1	1	Reg[5] \leftarrow Reg[6] + Reg[7] = -3

0100	1	010	010	101	1	0	<p>Register File Contents:</p> <p>Reg[7] \leftarrow 6</p> <p>Reg[6] \leftarrow 3</p> <p>Reg[5] \leftarrow -3</p> <p>Reg[4] \leftarrow 4</p> <p>Reg[3] \leftarrow 5</p> <p>Reg[2] \leftarrow 2</p> <p>Reg[1] \leftarrow -1</p> <p>Reg[0] \leftarrow 6</p>
------	---	-----	-----	-----	---	---	--

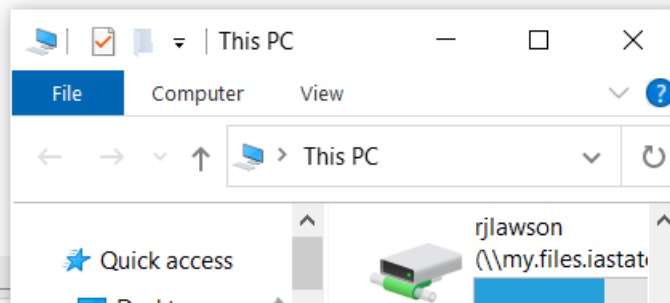
5.0 Register File with Adder and 7-Seg Displays

Screenshots



Lab 12 Answer Sheet

```
module Decoder_3to8(EN, W2, W1, W0, Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7);  
    input EN, W2, W1, W0;  
    output reg Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;  
  
    always @ (EN, W2, W1, W0)  
    begin  
        //reg Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;  
  
        //initial begin  
            {Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7} = 8'b0;  
        //end  
  
        if (EN)  
        begin  
            case ({W2, W1, W0})  
                3'b000: Y0 = 1'b1;  
                3'b001: Y1 = 1'b1;  
                3'b010: Y2 = 1'b1;  
                3'b011: Y3 = 1'b1;  
                3'b100: Y4 = 1'b1;  
                3'b101: Y5 = 1'b1;  
                3'b110: Y6 = 1'b1;  
                3'b111: Y7 = 1'b1;  
            endcase  
        end  
    end  
endmodule
```



Cpr E 281 LAB
EXAM
 ELECTRICAL AND COMPUTER
 ENGINEERING
 IOWA STATE UNIVERSITY

Lab 12 Answer Sheet

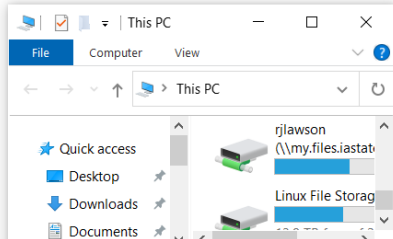
```
module regfile(DATAP3, DATAP2, DATAP1, DATAP0, DATAQ3, DATAQ2, DATAQ1, DATAQ0,
    RP2, RP1, RP0, RQ2, RQ1, RQ0, WA2, WA1, WA0, LD_DATA, WR, CLRN, CLK);
    // address and control ports
    input RP2, RP1, RP0, RQ2, RQ1, RQ0, WA2, WA1, WA0, WR, CLRN, CLK;
    // input data port
    input [3:0] LD_DATA;
    // output data ports
    output DATAP3, DATAP2, DATAP1, DATAP0, DATAQ3, DATAQ2, DATAQ1, DATAQ0;
    // wire declarations
    wire [3:0] DATAP, DATAQ;

    wire EN0, EN1, EN3, EN4, EN5, EN6, EN7;
    wire[3:0] DATA0, DATA1, DATA2, DATA3, DATA4, DATA5, DATA6, DATA7

    Decoder3to8 Decode (.EN(1'b1), .W1(WA2), .W0(WA1), .W0(WA0), .Y0(EN0), .Y1(EN1), .Y2(EN2), .Y3(EN3), .Y4(EN4), .Y5(EN5), .
    Mux8_4b PMUX (.S2(RP2), .S1(RP1), .S0(RP0), .W0(DATA0), .W1(DATA1), .W2(DATA2), .W3(DATA3), .W4(DATA4), .W5(DATA5), .W6(DATA6),
    Mux8_4b QMUX (.S2(RQ2), .S1(RQ1), .S0(RQ0), .W0(DATA0), .W1(DATA1), .W2(DATA2), .W3(DATA3), .W4(DATA4), .W5(DATA5), .W6(DATA6)
```

```
    assign DATAP0 = DATAP[0];
    assign DATAP1 = DATAP[1];
    assign DATAP2 = DATAP[2];
    assign DATAP3 = DATAP[3];

    assign DATAQ0 = DATAQ[0];
    assign DATAQ1 = DATAQ[1];
    assign DATAQ2 = DATAQ[2];
    assign DATAQ3 = DATAQ[3];
endmodule
```



```
module mux8_4b(S2, S1, S0, W0, W1, W2, W3, W4, W5, W6, W7, F)
    input S2, S1, S0;
    input [3:0] W0, W1, W2, W3, W4, W5, W6, W7;
    output [3:0] F;

    reg[3:0] F;

    always @(W0 or W1 or W2 or W3 or W4 or W5 or W6 or W7)

    begin
        case ({S2, S1, S0})
            3'b000 : F = W0;
            3'b001 : F = W1;
            3'b010 : F = W2;
            3'b011 : F = W3;
            3'b100 : F = W4;
            3'b101 : F = W5;
            3'b110 : F = W6;
            3'b111 : F = W7;
        endcase
    end
endmodule
```

