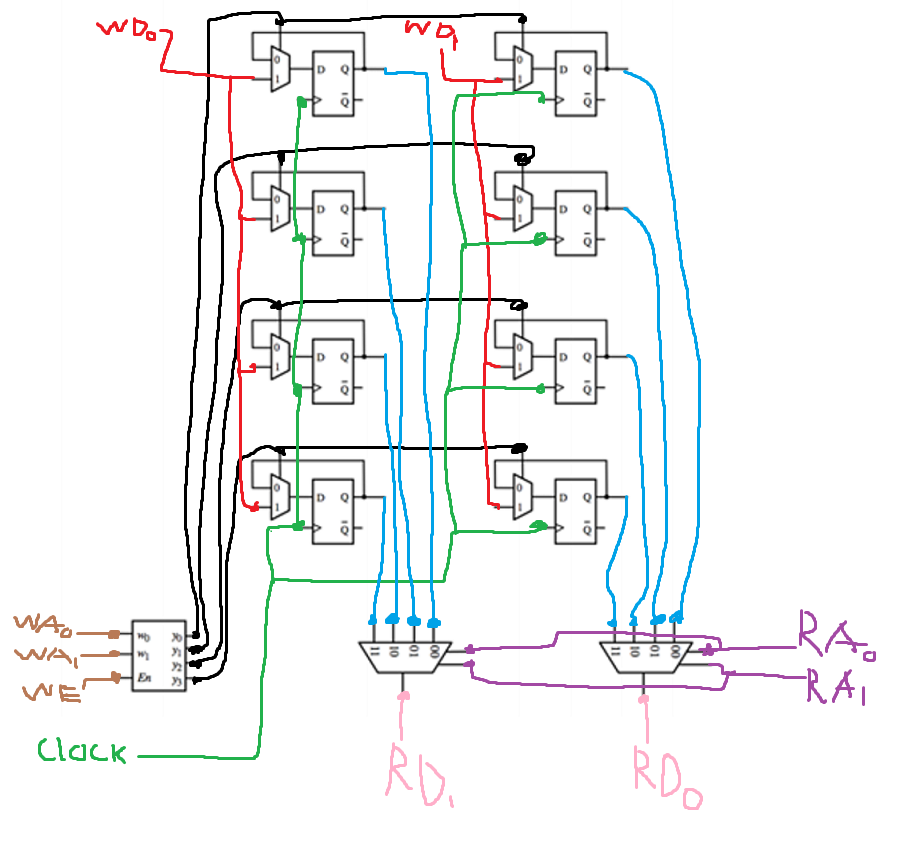
Name and Student ID: Wyatt Duberstein 629635057 Lab Section: 19

Date: 11/20/2020

**PRELAB:**

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

Please complete the following circuit diagram to implement a register file with four 2-bit registers, one write port, one read port, and one write enable line. Label all inputs and outputs of your circuit. Please use different colors for the different types of wires.



**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 @(S2, S1, S0, W0, W1, W2, W3, W4, W5, W6, W7, F)

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

**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;

reg Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;

always @(EN, W2, W1, W0)

begin

if (EN)

begin

Y0=1'b0;

Y1=1'b0;

y2=1'b0;

Y3=1'b0;

Y4=1'b0;

Y5=1'b0;

Y6=1'b0;

Y7=1'b0;

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

else

Y0=1'b0;

Y1=1'b0;

y2=1'b0;

Y3=1'b0;

Y4=1'b0;

Y5=1'b0;

Y6=1'b0;

Y7=1'b0;

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 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;

input wire RP2;

input wire RP1;

input wire RP0;

input wire RQ2;

input wire RQ1;

input wire RQ0;

input wire WA2;

input wire WA1;

input wire WA0;

input wire WR;

input wire CLRN;

input wire CLK;

input wire [3:0] LD\_DATA;

output wire DATAP3;

output wire DATAP2;

output wire DATAP1;

output wire DATAP0;

output wire DATAQ3;

output wire DATAQ2;

output wire DATAQ1;

output wire DATAQ0;

wire SYNTHESIZED\_WIRE\_0;

wire [3:0] SYNTHESIZED\_WIRE\_24;

wire [3:0] SYNTHESIZED\_WIRE\_25;

wire [3:0] SYNTHESIZED\_WIRE\_26;

wire [3:0] SYNTHESIZED\_WIRE\_27;

wire [3:0] SYNTHESIZED\_WIRE\_28;

wire [3:0] SYNTHESIZED\_WIRE\_29;

wire [3:0] SYNTHESIZED\_WIRE\_30;

wire [3:0] SYNTHESIZED\_WIRE\_31;

wire SYNTHESIZED\_WIRE\_9;

wire SYNTHESIZED\_WIRE\_10;

wire SYNTHESIZED\_WIRE\_11;

wire SYNTHESIZED\_WIRE\_12;

wire SYNTHESIZED\_WIRE\_13;

wire SYNTHESIZED\_WIRE\_14;

wire SYNTHESIZED\_WIRE\_15;

Decoder3to8 b2v\_inst(

.EN(WR),

.W2(WA2),

.W1(WA1),

.W0(WA0),

.Y0(SYNTHESIZED\_WIRE\_15),

.Y1(SYNTHESIZED\_WIRE\_0),

.Y2(SYNTHESIZED\_WIRE\_9),

.Y3(SYNTHESIZED\_WIRE\_14),

.Y4(SYNTHESIZED\_WIRE\_10),

.Y5(SYNTHESIZED\_WIRE\_11),

.Y6(SYNTHESIZED\_WIRE\_12),

.Y7(SYNTHESIZED\_WIRE\_13));

reg4b b2v\_inst1(

.LOAD(SYNTHESIZED\_WIRE\_0),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_25));

Mux8\_4b b2v\_inst10(.S2(RP2), .S1(RP1), .S0(RP0),

.W0(SYNTHESIZED\_WIRE\_24),

.W1(SYNTHESIZED\_WIRE\_25),

.W2(SYNTHESIZED\_WIRE\_26),

.W3(SYNTHESIZED\_WIRE\_27),

.W4(SYNTHESIZED\_WIRE\_28),

.W5(SYNTHESIZED\_WIRE\_29),

.W6(SYNTHESIZED\_WIRE\_30),

.W7(SYNTHESIZED\_WIRE\_31)

);

reg4b b2v\_inst2(

.LOAD(SYNTHESIZED\_WIRE\_9),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_26));

reg4b b2v\_inst3(

.LOAD(SYNTHESIZED\_WIRE\_10),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_28));

reg4b b2v\_inst4(

.LOAD(SYNTHESIZED\_WIRE\_11),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_29));

reg4b b2v\_inst5(

.LOAD(SYNTHESIZED\_WIRE\_12),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_30));

reg4b b2v\_inst6(

.LOAD(SYNTHESIZED\_WIRE\_13),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_31));

reg4b b2v\_inst7(

.LOAD(SYNTHESIZED\_WIRE\_14),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_27));

reg4b b2v\_inst8(

.LOAD(SYNTHESIZED\_WIRE\_15),

.CLK(CLK),

.CLRN(CLRN),

.IN(LD\_DATA),

.Out(SYNTHESIZED\_WIRE\_24));

Mux8\_4b b2v\_inst9(

.S2(RQ2),

.S1(RQ1),

.S0(RQ0),

.W0(SYNTHESIZED\_WIRE\_24),

.W1(SYNTHESIZED\_WIRE\_25),

.W2(SYNTHESIZED\_WIRE\_26),

.W3(SYNTHESIZED\_WIRE\_27),

.W4(SYNTHESIZED\_WIRE\_28),

.W5(SYNTHESIZED\_WIRE\_29),

.W6(SYNTHESIZED\_WIRE\_30),

.W7(SYNTHESIZED\_WIRE\_31)

);

Endmodule

Prelab TA Initials: PS

**LAB:**

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

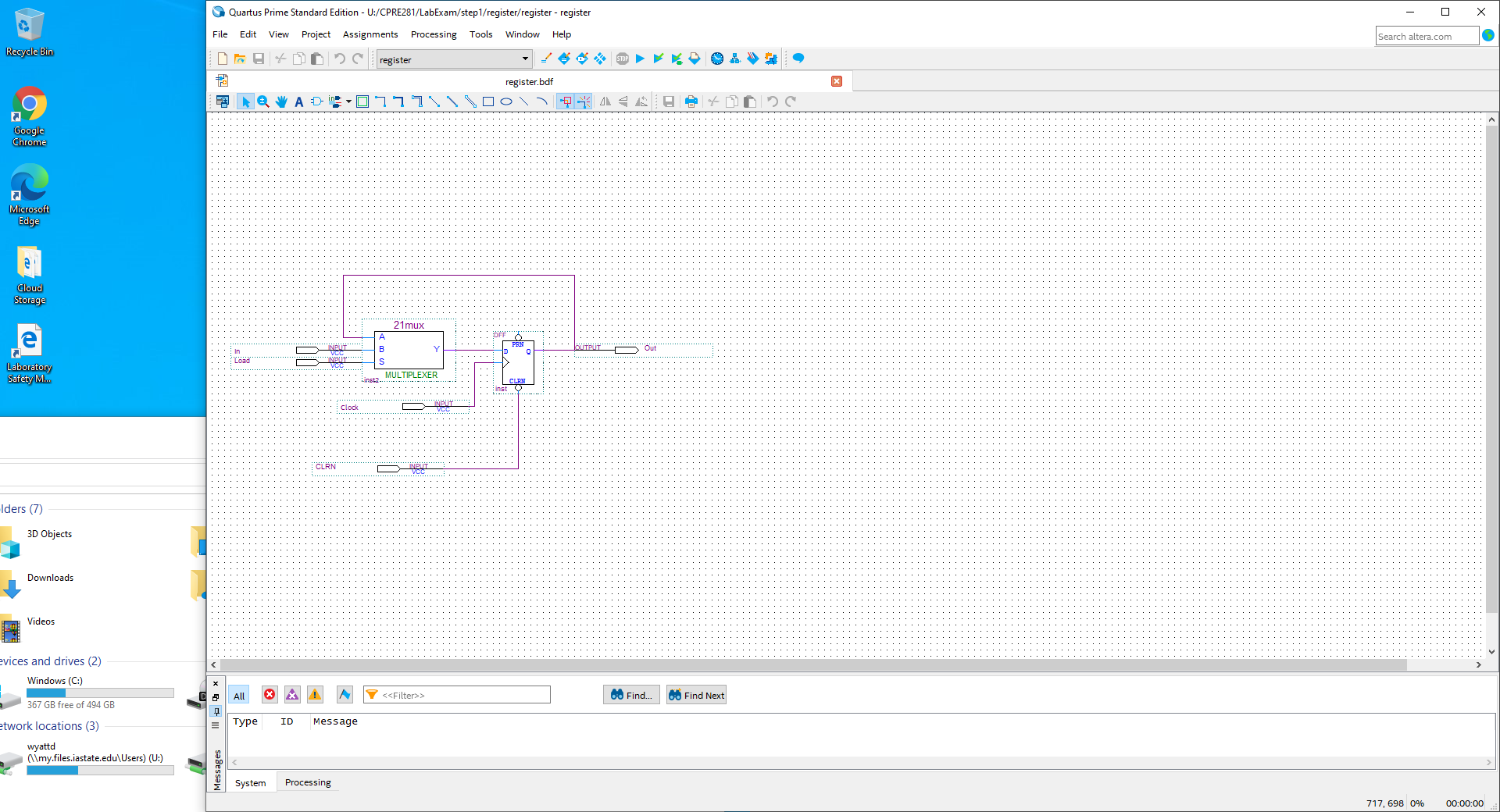
|  |  |  |
| --- | --- | --- |
| In | Load | Out |
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

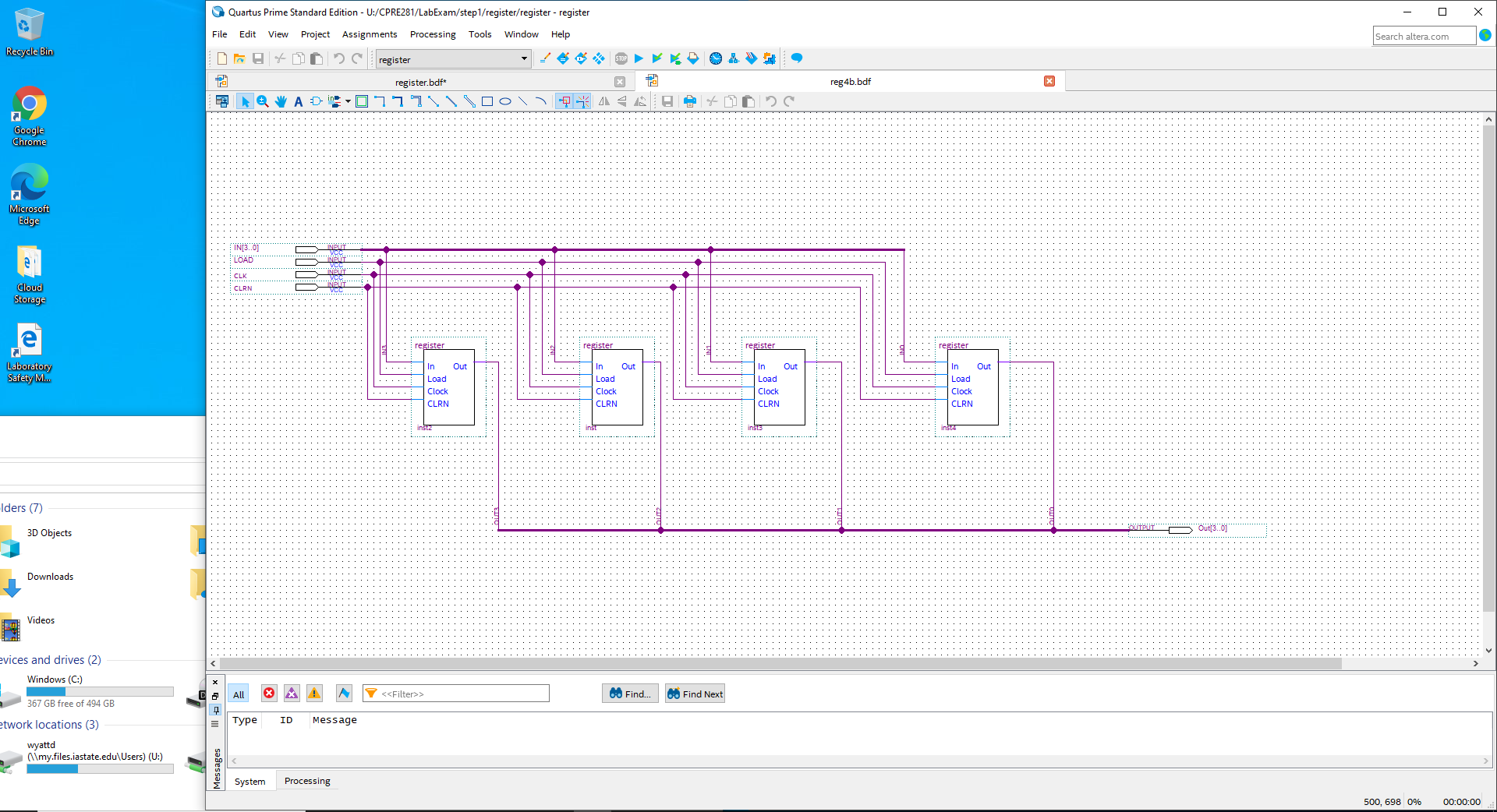
Fill in the table with steps that will load the registers as follows: Reg[0]=F, Reg[1]=A. Reg{2]=C. Reg[3]=E, Reg[4]=2, Reg[5]=7, Reg[6]=6, and Reg[7]=1.

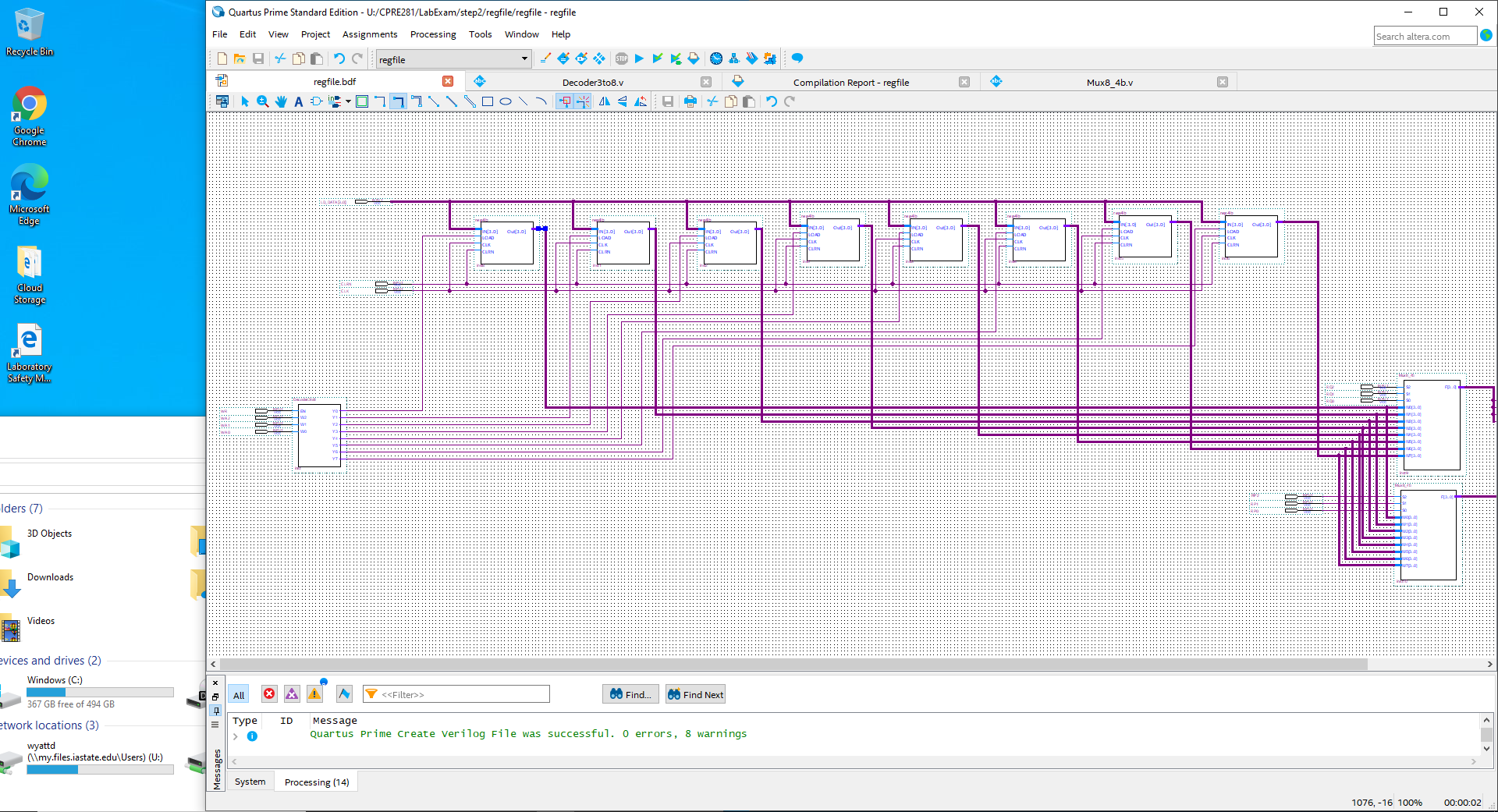
|  |  |
| --- | --- |
| LD\_DATA | WA |
| Reg[0] | F |
| Reg[1] | A |
| Reg[2] | C |
| Reg[3] | E |
| Reg[4] | 2 |
| Reg[5] | 7 |
| Reg[6] | 6 |
| Reg[7] | 1 |

**4.0 Register File with Eight 4-bit Registers**

Screenshots:







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] 🡨 5 |
| 0010 | 0 | 101 | 101 | 110 | 1 | 1 | Reg[5] 🡨 4 |
| 0100 | 0 | 100 | 100 | 101 | 1 | 1 | Reg[4] 🡨 3 |
| 0101 | 0 | 011 | 011 | 100 | 0 | 1 | Reg[3] 🡨 2 |
| 0001 | 0 | 010 | 010 | 011 | 0 | 1 | Reg[2] 🡨 1 |
| 0111 | 0 | 001 | 001 | 010 | 1 | 1 | Reg[1] 🡨 0 |
| 1000 | 0 | 000 | 000 | 001 | 1 | 1 | Register File Contents:  FACE2761 |
| 0000 | 1 | 001 | 000 | 001 | 0 | 1 | Reg[4] 🡨 3 |
| 0001 | 1 | 000 | 010 | 011 | 0 | 1 | Reg[3] 🡨 2 |
| 1111 | 1 | 010 | 100 | 101 | 1 | 1 | Reg[2] 🡨 1 |
| 1001 | 1 | 101 | 110 | 111 | 1 | 1 | Reg[1] 🡨 0 |
| 0100 | 1 | 010 | 010 | 101 | 1 | 0 | Register File Contents:  FACE2761 |

**5.0 Register File with Adder and 7-Seg Displays**

