DDCO ASSIGNMENT

WEEK 5

GAUTAM VIJAY HANCHINAL

PES1UG22CS215

SECTION D

1) Library File:

```
2)    module invert (input wire i, output wire o);
3)       assign o = !i;
4)    endmodule
5)
6)    module and2 (input wire i0, i1, output wire o);
7)      assign o = i0 & i1;
8)    endmodule
9)
10)   module or2 (input wire i0, i1, output wire o);
11)     assign o = i0 | i1;
12)   endmodule
13)
14)   module xor2 (input wire i0, i1, output wire o);
15)     assign o = i0 ^ i1;
16)   endmodule
17)
18)   module nand2 (input wire i0, i1, output wire o);
19)      wire t;
20)      and2 and2_0 (i0, i1, t);
21)      invert invert_0 (t, o);
22)   endmodule
23)
24)   module nor2 (input wire i0, i1, output wire o);
25)      wire t;
26)      or2 or2_0 (i0, i1, t);
27)      invert invert_0 (t, o);
28)   endmodule
29)
30)   module xnor2 (input wire i0, i1, output wire o);
31)      wire t;
32)      xor2 xor2_0 (i0, i1, t);
33)      invert invert_0 (t, o);
34)   endmodule
35)
36)   module and3 (input wire i0, i1, i2, output wire o);
37)      wire t;
38)      and2 and2_0 (i0, i1, t);
39)      and2 and2_1 (i2, t, o);
40)   endmodule
41)
42)   module or3 (input wire i0, i1, i2, output wire o);
43)      wire t;
44)      or2 or2_0 (i0, i1, t);
45)      or2 or2_1 (i2, t, o);
46)   endmodule
47)
```

```
48)   module nor3 (input wire i0, i1, i2, output wire o);
49)      wire t;
50)      or2 or2_0 (i0, i1, t);
51)      nor2 nor2_0 (i2, t, o);
52)   endmodule
53)
54)   module nand3 (input wire i0, i1, i2, output wire o);
55)      wire t;
56)      and2 and2_0 (i0, i1, t);
57)      nand2 nand2_1 (i2, t, o);
58)   endmodule
59)
60)   module xor3 (input wire i0, i1, i2, output wire o);
61)      wire t;
62)      xor2 xor2_0 (i0, i1, t);
63)      xor2 xor2_1 (i2, t, o);
64)   endmodule
65)
66)   module xnor3 (input wire i0, i1, i2, output wire o);
67)      wire t;
68)      xor2 xor2_0 (i0, i1, t);
69)      xnor2 xnor2_0 (i2, t, o);
70)   endmodule
71)
72)   module mux2 (input wire i0, i1, j, output wire o);
73)     assign o = (j==0)?i0:i1;
74)   endmodule
75)
76)   module mux4 (input wire [0:3] i, input wire j1, j0, output wire o);
77)     wire  t0, t1;
78)     mux2 mux2_0 (i[0], i[1], j1, t0);
79)     mux2 mux2_1 (i[2], i[3], j1, t1);
80)     mux2 mux2_2 (t0, t1, j0, o);
81)   endmodule
82)
83)   module mux8 (input wire [0:7] i, input wire j2, j1, j0, output wire
      o);
84)     wire  t0, t1;
85)     mux4 mux4_0 (i[0:3], j2, j1, t0);
86)     mux4 mux4_1 (i[4:7], j2, j1, t1);
87)     mux2 mux2_0 (t0, t1, j0, o);
88)   endmodule
89)
90)   module demux2 (input wire i, j, output wire o0, o1);
91)     assign o0 = (j==0)?i:1'b0;
92)     assign o1 = (j==1)?i:1'b0;
93)   endmodule
94)
```

```
95)   module demux4 (input wire i, j1, j0, output wire [0:3] o);
96)     wire  t0, t1;
97)     demux2 demux2_0 (i, j1, t0, t1);
98)     demux2 demux2_1 (t0, j0, o[0], o[1]);
99)     demux2 demux2_2 (t1, j0, o[2], o[3]);
100)  endmodule
101)
102)  module demux8 (input wire i, j2, j1, j0, output wire [0:7] o);
103)    wire  t0, t1;
104)    demux2 demux2_0 (i, j2, t0, t1);
105)    demux4 demux4_0 (t0, j1, j0, o[0:3]);
106)    demux4 demux4_1 (t1, j1, j0, o[4:7]);
107)  endmodule
108)
109)  module df (input wire clk, in, output wire out);
110)    reg df_out;
111)    always@(posedge clk) df_out <= in;
112)    assign out = df_out;
113)  endmodule
114)
115)  module dfr (input wire clk, reset, in, output wire out);
116)    wire reset_, df_in;
117)    invert invert_0 (reset, reset_);
118)    and2 and2_0 (in, reset_, df_in);
119)    df df_0 (clk, df_in, out);
120)  endmodule
121)
122)  module dfrl (input wire clk, reset, load, in, output wire out);
123)    wire _in;
124)    mux2 mux2_0(out, in, load, _in);
125)    dfr dfr_1(clk, reset, _in, out);
126)  endmodule
```

2) Register File:

```
3)    module dfrl_16 (input wire clk, reset, load, input wire [15:0] in,
      output wire [15:0] out);
4)    dfrl _f0(clk,reset,load,in[0],out[0]);
5)    dfrl _f1(clk,reset,load,in[1],out[1]);
6)    dfrl _f2(clk,reset,load,in[2],out[2]);
7)    dfrl _f3(clk,reset,load,in[3],out[3]);
8)    dfrl _f4(clk,reset,load,in[4],out[4]);
9)    dfrl _f5(clk,reset,load,in[5],out[5]);
10)   dfrl _f6(clk,reset,load,in[6],out[6]);
11)   dfrl _f7(clk,reset,load,in[7],out[7]);
12)   dfrl _f8(clk,reset,load,in[8],out[8]);
```

```
13)   dfrl _f9(clk,reset,load,in[9],out[9]);
14)   dfrl _f10(clk,reset,load,in[10],out[10]);
15)   dfrl _f11(clk,reset,load,in[11],out[11]);
16)   dfrl _f12(clk,reset,load,in[12],out[12]);
17)   dfrl _f13(clk,reset,load,in[13],out[13]);
18)   dfrl _f14(clk,reset,load,in[14],out[14]);
19)   dfrl _f15(clk,reset,load,in[15],out[15]);
20)   endmodule
21)   module mux8_16 (input wire [0:15] i0, i1, i2, i3, i4, i5, i6, i7,
      input wire [0:2] j, output wire [0:15] o);
22)     mux8 mux8_0({i0[0], i1[0], i2[0], i3[0], i4[0], i5[0], i6[0],
      i7[0]}, j[0], j[1], j[2], o[0]);
23)     mux8 mux8_1({i0[1], i1[1], i2[1], i3[1], i4[1], i5[1], i6[1],
      i7[1]}, j[0], j[1], j[2], o[1]);
24)     mux8 mux8_2({i0[2], i1[2], i2[2], i3[2], i4[2], i5[2], i6[2],
      i7[2]}, j[0], j[1], j[2], o[2]);
25)     mux8 mux8_3({i0[3], i1[3], i2[3], i3[3], i4[3], i5[3], i6[3],
      i7[3]}, j[0], j[1], j[2], o[3]);
26)     mux8 mux8_4({i0[4], i1[4], i2[4], i3[4], i4[4], i5[4], i6[4],
      i7[4]}, j[0], j[1], j[2], o[4]);
27)     mux8 mux8_5({i0[5], i1[5], i2[5], i3[5], i4[5], i5[5], i6[5],
      i7[5]}, j[0], j[1], j[2], o[5]);
28)     mux8 mux8_6({i0[6], i1[6], i2[6], i3[6], i4[6], i5[6], i6[6],
      i7[6]}, j[0], j[1], j[2], o[6]);
29)     mux8 mux8_7({i0[7], i1[7], i2[7], i3[7], i4[7], i5[7], i6[7],
      i7[7]}, j[0], j[1], j[2], o[7]);
30)     mux8 mux8_8({i0[8], i1[8], i2[8], i3[8], i4[8], i5[8], i6[8],
      i7[8]}, j[0], j[1], j[2], o[8]);
31)     mux8 mux8_9({i0[9], i1[9], i2[9], i3[9], i4[9], i5[9], i6[9],
      i7[9]}, j[0], j[1], j[2], o[9]);
32)     mux8 mux8_10({i0[10], i1[10], i2[10], i3[10], i4[10], i5[10],
      i6[10], i7[10]}, j[0], j[1], j[2], o[10]);
33)     mux8 mux8_11({i0[11], i1[11], i2[11], i3[11], i4[11], i5[11],
      i6[11], i7[11]}, j[0], j[1], j[2], o[11]);
34)     mux8 mux8_12({i0[12], i1[12], i2[12], i3[12], i4[12], i5[12],
      i6[12], i7[12]}, j[0], j[1], j[2], o[12]);
35)     mux8 mux8_13({i0[13], i1[13], i2[13], i3[13], i4[13], i5[13],
      i6[13], i7[13]}, j[0], j[1], j[2], o[13]);
36)     mux8 mux8_14({i0[14], i1[14], i2[14], i3[14], i4[14], i5[14],
      i6[14], i7[14]}, j[0], j[1], j[2], o[14]);
37)     mux8 mux8_15({i0[15], i1[15], i2[15], i3[15], i4[15], i5[15],
      i6[15], i7[15]}, j[0], j[1], j[2], o[15]);
38)   endmodule
39)   module reg_file (input wire  clk, reset, wr, input wire [0:2]
      rd_addr_a, rd_addr_b, wr_addr, input wire [0:15] d_in, output wire
      [0:15] d_out_a, d_out_b);
40)     wire [0:7] load;
```

```
41)     wire [0:15] dout_0, dout_1, dout_2, dout_3, dout_4, dout_5,
      dout_6, dout_7;
42)     dfrl_16 dfrl_16_0(clk, reset, load[0], d_in, dout_0);
43)     dfrl_16 dfrl_16_1(clk, reset, load[1], d_in, dout_1);
44)     dfrl_16 dfrl_16_2(clk, reset, load[2], d_in, dout_2);
45)     dfrl_16 dfrl_16_3(clk, reset, load[3], d_in, dout_3);
46)     dfrl_16 dfrl_16_4(clk, reset, load[4], d_in, dout_4);
47)     dfrl_16 dfrl_16_5(clk, reset, load[5], d_in, dout_5);
48)     dfrl_16 dfrl_16_6(clk, reset, load[6], d_in, dout_6);
49)     dfrl_16 dfrl_16_7(clk, reset, load[7], d_in, dout_7);
50)     demux8 demux8_0(wr, wr_addr[2], wr_addr[1], wr_addr[0], load);
51)     mux8_16 mux8_16_9(dout_0, dout_1, dout_2, dout_3, dout_4, dout_5,
      dout_6, dout_7, rd_addr_a, d_out_a);
52)     mux8_16 mux8_16_10(dout_0, dout_1, dout_2, dout_3, dout_4, dout_5,
      dout_6, dout_7, rd_addr_b, d_out_b);
53)   endmodule
54)
55)
```

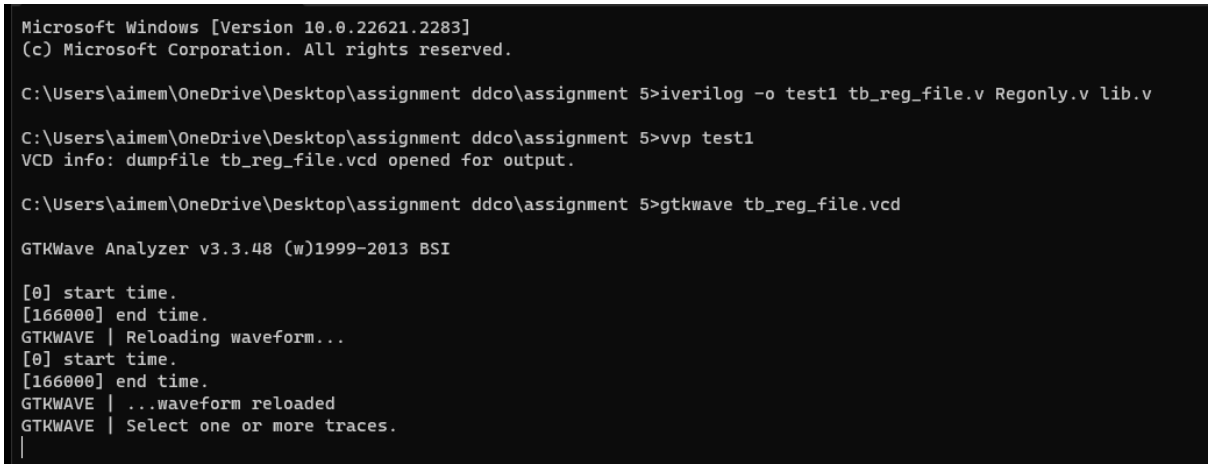
3) Test Bench Code:

```
`timescale 1 ns / 100 ps
`define TESTVECS 6

module tb;
  reg clk, reset, wr;
  reg [2:0] rd_addr_a, rd_addr_b, wr_addr; reg [15:0] d_in;
  wire [15:0] d_out_a, d_out_b;
  reg [25:0] test_vecs [0:(`TESTVECS-1)];
  integer i;
  initial begin $dumpfile("tb_reg_file.vcd"); $dumpvars(0,tb); end
  initial begin reset = 1'b1; #12.5 reset = 1'b0; end
  initial clk = 1'b0; always #5 clk =~ clk;
  initial begin
    test_vecs[0][25] = 1'b1; test_vecs[0][24:22] = 3'ox; test_vecs[0][21:19] =
3'ox;
    test_vecs[0][18:16] = 3'h3; test_vecs[0][15:0] = 16'hcdef;
    test_vecs[1][25] = 1'b1; test_vecs[1][24:22] = 3'ox; test_vecs[1][21:19] =
3'ox;
    test_vecs[1][18:16] = 3'o7; test_vecs[1][15:0] = 16'h3210;
    test_vecs[2][25] = 1'b1; test_vecs[2][24:22] = 3'o3; test_vecs[2][21:19] =
3'o7;
    test_vecs[2][18:16] = 3'o5; test_vecs[2][15:0] = 16'h4567;
    test_vecs[3][25] = 1'b1; test_vecs[3][24:22] = 3'o1; test_vecs[3][21:19] =
3'o5;
    test_vecs[3][18:16] = 3'o0; test_vecs[3][15:0] = 16'hba98;
```

```
    test_vecs[4][25] = 1'b0; test_vecs[4][24:22] = 3'o1; test_vecs[4][21:19] =
3'o5;
    test_vecs[4][18:16] = 3'o1; test_vecs[4][15:0] = 16'hxxxx;
    test_vecs[5][25] = 1'b0; test_vecs[5][24:22] = 3'o0; test_vecs[5][21:19] =
3'o0;
    test_vecs[5][18:16] = 3'ox; test_vecs[5][15:0] = 16'hxxxx;
  end
  initial {wr, rd_addr_a, rd_addr_b, wr_addr, d_in} = 0;
  reg_file reg_file_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in,
d_out_a, d_out_b);
  initial begin
    #6 for(i=0;i<`TESTVECS;i=i+1)
      begin #10 {wr, rd_addr_a, rd_addr_b, wr_addr, d_in}=test_vecs[i]; end
    #100 $finish;
  end
endmodule
```

CMD screenshots:



GTKWAVE screenshots confirming the test cases from the truth table: