### ĐAI HOC QUOC GIA THÀNH PHO HO CHÍ MINH TRƯèNG ĐAI HOC BÁCH KHOA



Thiet ke luan ly

# Báo cáo bài lab 3

NHÓM 14

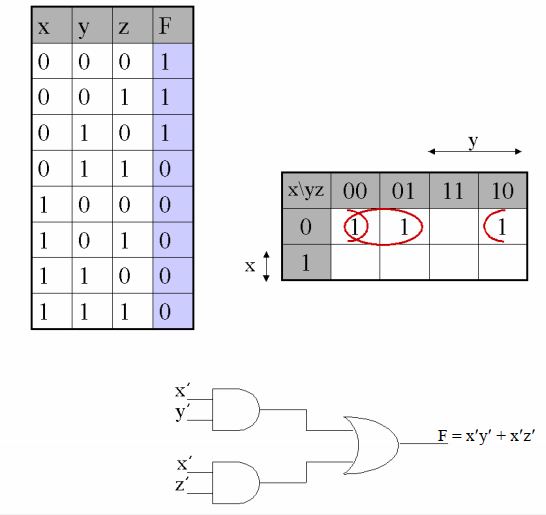
### GVHD: Huỳnh Phúc Nghi

SV: Tran Quoc To - 1814385 Nguyen Quoc Manh - 1813043

TP. HO CHÍ MINH, THÁNG 4/2021

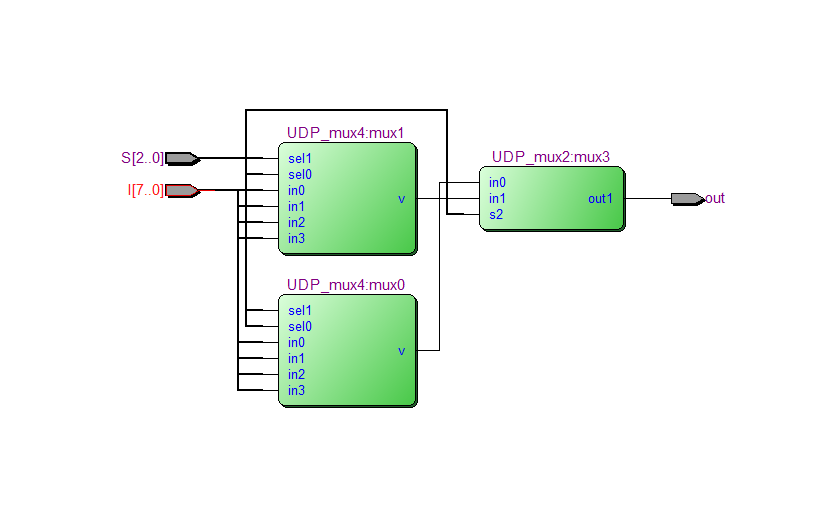
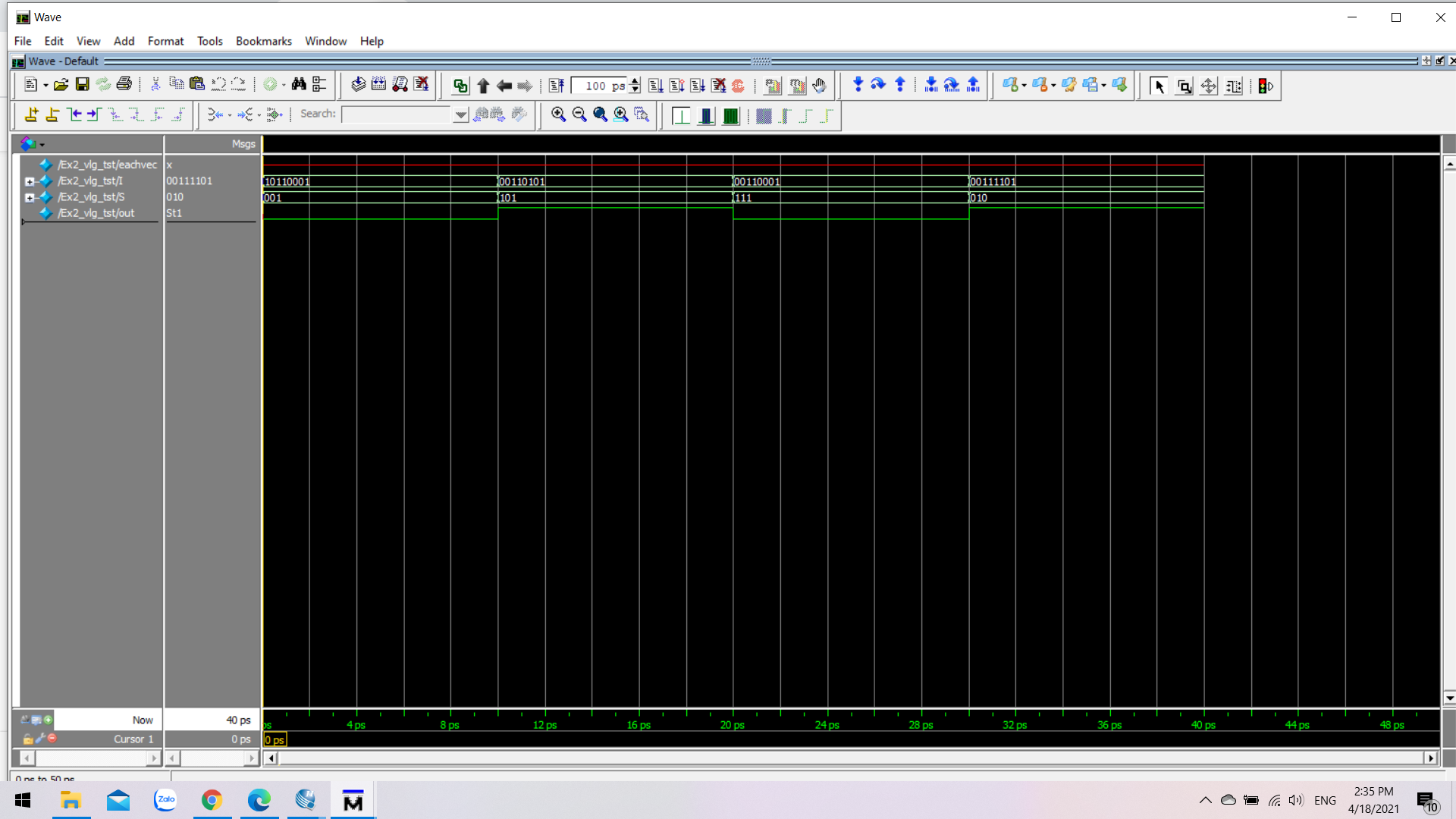
Ex1\_lap3

Use block diagram/schematic file to design an arbitrary logic circuit that has at least 3 inputs and 1 output.



|  |
| --- |
| module Ex1(f,x,y,z); |
|  | input x,y,z; |
|  | output f; |
|  | not X1(x1,x); |
|  | not Y1(y1,y); |
|  | not Z1(z1,z); |
|  | and XY(n1,x1,y1); |
|  | and XZ(n2,x1,z1); |
|  | or F(f,n1,n2); |
|  | endmodule |

EX2



|  |
| --- |
| `timescale 1ns / 1ps |
|  |  |
|  |  |
|  |  |
|  | module mux8\_1( |
|  | output out, |
|  | input [7:0] I, |
|  | input [2:0] S |
|  | ); |
|  |  |
|  | wire w1, w2; |
|  | UDP\_mux4 mux0 (w1, S[1], S[0], I[0], I[1], I[2], I[3]); |
|  | UDP\_mux4 mux1 (w2, S[1], S[0], I[4], I[5], I[6], I[7]); |
|  | UDP\_mux2 mux3 (out, w1, w2, S[2]); |
|  | endmodule |
|  |  |
|  | primitive UDP\_mux2 (out1, in0, in1, s2); |
|  | output out1; |
|  | input in0, in1, s2; |
|  |  |
|  | table |
|  | // in0 in1 s2 : out1 |
|  | 0 ? 0 : 0; |
|  | 1 ? 0 : 1; |
|  | ? 0 1 : 0; |
|  | ? 1 1 : 1; |
|  | //0 0 ? : 0; |
|  | //1 1 ? : 1; |
|  | endtable |
|  | endprimitive |
|  |  |
|  |  |
|  | primitive UDP\_mux4 (y, sel1, sel0, in0, in1, in2, in3); |
|  | output y; |
|  | input in0, in1, in2, in3, sel0, sel1; |
|  | table |
|  | //sel1 sel0 in0 in1 in2 in3 : out |
|  | 0 0 0 ? ? ? : 0; |
|  | 0 0 1 ? ? ? : 1; |
|  | 0 1 ? 0 ? ? : 0; |
|  | 0 1 ? 1 ? ? : 1; |
|  | 1 0 ? ? 0 ? : 0; |
|  | 1 0 ? ? 1 ? : 1; |
|  | 1 1 ? ? ? 0 : 0; |
|  | 1 1 ? ? ? 1 : 1; |
|  | endtable |
|  | endprimitive |
|  |  |
|  | /////////////////////////////////////////////////////////////////////////// |
|  |  |
|  | //Question 1: 8 to 1 mux test bench |
|  |  |
|  | `timescale 1ns / 1ps |
|  |  |
|  |  |
|  | module mux8\_1\_tb(); |
|  |  |
|  | reg [7:0] I; |
|  | reg [2:0] S; |
|  |  |
|  | wire out; |
|  |  |
|  | mux8\_1 uut ( |
|  | .out(out), |
|  | .I(I), |
|  | .S(S) |
|  | ); |
|  |  |
|  | initial |
|  | $monitor ("I=%b, S=%b, out%b", I, S, out); |
|  |  |
|  | initial |
|  | begin |
|  |  |
|  | #0 I = 8'b10110001; |
|  | S = 3'b001; |
|  |  |
|  | #10 I = 8'b00110101; |
|  | S = 3'b101; |
|  |  |
|  | #10 I = 8'b00110001; |
|  | S = 3'b111; |
|  |  |
|  | #10 I = 8'b00111101; |
|  | S = 3'b010; |
|  |  |
|  | #10 $stop; |
|  | end |
|  | endmodule |
|  |  |
|  | /////////////////////////////////////////////////////////////////////////// |
|  |  |
|  | //Question 2: Majority Gate |
|  |  |
|  | `timescale 1ns / 1ps |
|  |  |
|  |  |
|  | primitive majorityGate( |
|  | output z1, |
|  | input x1, |
|  | input x2, |
|  | input x3, |
|  | input x4, |
|  | input x5 |
|  | ); |
|  |  |
|  | table |
|  | //x1 x2 x3 x4 x5 : z1 |
|  | 0 0 0 0 0 : 0; |
|  | 0 0 0 0 1 : 0; |
|  | 0 0 0 1 0 : 0; |
|  | 0 0 0 1 1 : 0; |
|  | 0 0 1 0 0 : 0; |
|  | 0 0 1 0 1 : 0; |
|  | 0 0 1 1 0 : 0; |
|  | 0 0 1 1 1 : 1; |
|  | 0 1 0 0 0 : 0; |
|  | 0 1 0 0 1 : 0; |
|  | 0 1 0 1 0 : 0; |
|  | 0 1 0 1 1 : 1; |
|  | 0 1 1 0 0 : 0; |
|  | 0 1 1 0 1 : 1; |
|  | 0 1 1 1 0 : 1; |
|  | 0 1 1 1 1 : 1; |
|  | 1 0 0 0 0 : 0; |
|  | 1 0 0 0 1 : 0; |
|  | 1 0 0 1 0 : 0; |
|  | 1 0 0 1 1 : 1; |
|  | 1 0 1 0 0 : 0; |
|  | 1 0 1 0 1 : 1; |
|  | 1 0 1 1 0 : 1; |
|  | 1 0 1 1 1 : 1; |
|  | 1 1 0 0 0 : 0; |
|  | 1 1 0 0 1 : 1; |
|  | 1 1 0 1 0 : 1; |
|  | 1 1 0 1 1 : 1; |
|  | 1 1 1 0 0 : 1; |
|  | 1 1 1 0 1 : 1; |
|  | 1 1 1 1 0 : 1; |
|  | 1 1 1 1 1 : 1; |
|  | endtable |
|  |  |
|  | endprimitive |
|  |  |
|  | ////////////////////////////////////////////////////////////////////////////// |
|  |  |
|  | //Question 2: Majority Gate test bench |
|  |  |
|  | `timescale 1ns / 1ps |
|  |  |
|  |  |
|  | module majorityGate\_tb(); |
|  |  |
|  | reg x1, x2, x3, x4, x5; |
|  | wire z1; |
|  |  |
|  | majorityGate uut( |
|  | .z1(z1), |
|  | .x1(x1), |
|  | .x2(x2), |
|  | .x3(x3), |
|  | .x4(x4), |
|  | .x5(x5) |
|  | ); |
|  |  |
|  |  |
|  | initial |
|  | $monitor ("x1=%b, x2=%b, x3=%b, x4=%b, x5=%b, z1=%b", x1, x2, x3, x4, x5, z1); |
|  |  |
|  | initial |
|  | begin |
|  |  |
|  | #0 x1 = 1'b0; |
|  | x2 = 1'b0; |
|  | x3 = 1'b1; |
|  | x4 = 1'b0; |
|  | x5 = 1'b1; |
|  |  |
|  | #10 x1 = 1'b1; |
|  | x2 = 1'b1; |
|  | x3 = 1'b1; |
|  | x4 = 1'b0; |
|  | x5 = 1'b1; |
|  |  |
|  | #10 x1 = 1'b0; |
|  | x2 = 1'b0; |
|  | x3 = 1'b1; |
|  | x4 = 1'b1; |
|  | x5 = 1'b0; |
|  |  |
|  | #10 x1 = 1'b1; |
|  | x2 = 1'b0; |
|  | x3 = 1'b1; |
|  | x4 = 1'b1; |
|  | x5 = 1'b0; |
|  |  |
|  | #10 $stop; |
|  | end |
|  | endmodule |

EX3

|  |  |  |  |
| --- | --- | --- | --- |
| module Ex3(in, out, clk,set,en,on,rst); | | | |
|  | | | | input [3:0]in; | |
|  | | | | input clk,set,en,on,rst; | |
|  | | | | output reg[7:0] out; | |
|  | | | |  | |
|  | | | | always@(\*) begin | |
|  | | | | case(in) | |
|  | | | | 4'b0000: out = 8'b00000000;//0 | |
|  | | | | 4'b0001: out = 8'b00000001;//1 | |
|  | | | | 4'b0010: out = 8'b00000010;//2 | |
|  | | | | 4'b0011: out = 8'b00000011;//3 | |
|  | | | | 4'b0100: out = 8'b00000100;//4 | |
|  | | | | 4'b0101: out = 8'b00000101;//5 | |
|  | | | | 4'b0110: out = 8'b00000110;//6 | |
|  | | | | 4'b0111: out = 8'b00000111;//7 | |
|  | | | | 4'b1000: out = 8'b00001000;//8 | |
|  | | | | 4'b1001: out = 8'b00001001;//9 | |
|  | | | | 4'b1010: out = 8'b00010000;//10 | |
|  | | | | 4'b1011: out = 8'b00010001;//11 | |
|  | | | | 4'b1100: out = 8'b00010010;//12 | |
|  | | | | 4'b1101: out = 8'b00010011;//13 | |
|  | | | | 4'b1110: out = 8'b00010100;//14 | |
|  | | | | 4'b1111: out = 8'b00010101;//15 | |
|  | | | | endcase | |
|  | | | | end | |
|  | | | | LED2 A( | |
|  | | | | .in(out) | |
|  | | | | ,.clk(clk) | |
|  | | | | ,.set(set) | |
|  | | | | ,.rst(rst) | |
|  | | | | ,.en(en) | |
|  | | | | ,.on(on) | |
|  | | | | ); | |
|  | | | | endmodule | |
|  |
| `timescale 1 ps/ 1 ps |
|  | module Ex3\_vlg\_tst(); | | | | |
|  | // constants | | | | |
|  | // general purpose registers | | | | |
|  | reg eachvec; | | | | |
|  | // test vector input registers | | | | |
|  | reg clk; | | | | |
|  | reg en; | | | | |
|  | reg [3:0] in; | | | | |
|  | reg on; | | | | |
|  | reg rst; | | | | |
|  | reg set; | | | | |
|  | // wires | | | | |
|  | wire [7:0] out; | | | | |
|  |  | | | | |
|  | // assign statements (if any) | | | | |
|  | Ex3 i1 ( | | | | |
|  | // port map - connection between master ports and signals/registers | | | | |
|  | .clk(clk), | | | | |
|  | .en(en), | | | | |
|  | .in(in), | | | | |
|  | .on(on), | | | | |
|  | .out(out), | | | | |
|  | .rst(rst), | | | | |
|  | .set(set) | | | | |
|  | ); | | | | |
|  | integer i; | | | | |
|  | initial | | | | |
|  | begin | | | | |
|  | clk = 0; | | | | |
|  | forever #5 clk = ~clk; | | | | |
|  | end | | | | |
|  | initial | | | | |
|  | begin | | | | |
|  | en = 0;rst= 1;on = 0;set = 0; | | | | |
|  | #10; | | | | |
|  | en = 1;rst = 0;set = 0;on = 1; | | | | |
|  | #10; | | | | |
|  | // code that executes only once | | | | |
|  | // insert code here --> begin | | | | |
|  | in = -1; | | | | |
|  | for(i = 0;i < 16;i = i +1) | | | | |
|  | begin | | | | |
|  | in = in + 1; #10; | | | | |
|  | end | | | | |
|  | // --> end | | | | |
|  | $display("Running testbench"); | | | | |
|  | end | | | | |
|  | always | | | | |
|  | // optional sensitivity list | | | | |
|  | // @(event1 or event2 or .... eventn) | | | | |
|  | begin | | | | |
|  | // code executes for every event on sensitivity list | | | | |
|  | // insert code here --> begin | | | | |
|  |  | | | | |
|  | @eachvec; | | | | |
|  | // --> end | | | | |
|  | end | | | | |
|  | endmodule | | | | |
|  |  | | | | |
| module LED(in,out,clk,set,rst,en,on); | |
|  | |  | | | |
|  | | input [3:0] in; | | | |
|  | | input rst,clk,set,en,on; | | | |
|  | | output reg [6:0] out; | | | |
|  | |  | | | |
|  | | parameter led0 = 7'b0111111,led1 = 7'b0000110,led2 = 7'b1011011,led3 = 7'b1001111,led4 = 7'b1100110,led5 = 7'b1101101,led6 = 7'b1111101 ,led7 = 7'b0000111,led8 = 7'b1111111,led9 = 7'b1101111; | | | |
|  | | parameter wrong = 7'b1000000; | | | |
|  | | parameter l0 = 4'b0000,l1 = 4'b0001,l2 = 4'b0010,l3 = 4'b0011,l4 = 4'b0100,l5 = 4'b0101,l6 = 4'b0110,l7 = 4'b0111,l8 = 4'b1000,l9 = 4'b1001; | | | |
|  | | always @(negedge clk,negedge rst) begin | | | |
|  | | if(!rst) out <= ~led0; | | | |
|  | | else if(on) begin | | | |
|  | | if(en) begin | | | |
|  | | if(set) out <= ~led8; | | | |
|  | | else begin | | | |
|  | | case (in) | | | |
|  | | l0 : out = ~led0 ; | | | |
|  | | l1 : out = ~led1 ; | | | |
|  | | l2 : out = ~led2 ; | | | |
|  | | l3 : out = ~led3 ; | | | |
|  | | l4 : out = ~led4 ; | | | |
|  | | l5 : out = ~led5 ; | | | |
|  | | l6 : out = ~led6 ; | | | |
|  | | l7 : out = ~led7 ; | | | |
|  | | l8 : out = ~led8 ; | | | |
|  | | l9 : out = ~led9 ; | | | |
|  | | endcase | | | |
|  | | if(in > 4'b1001) out <= ~wrong; | | | |
|  | | end | | | |
|  | | end | | | |
|  | | else out <= out; | | | |
|  | | end | | | |
|  | | else out <= 7'b1111111; | | | |
|  | | end | | | |
|  | |  | | | |
|  | |  | | | |
|  | | endmodule | | | |
| module LED2(in,out1,out2,clk,set,rst,en,on); | | |
|  | | |  | |
|  | | | input [7:0] in; | |
|  | | | input clk,set,rst,en,on; | |
|  | | | output [6:0] out1; | |
|  | | | output [6:0] out2; | |
|  | | | reg [3:0] in1,in0; | |
|  | | | always @(\*) begin | |
|  | | | in1 = in[7:4]; | |
|  | | | in0 = in[3:0]; | |
|  | | | end | |
|  | | |  | |
|  | | |  | |
|  | | | LED E( | |
|  | | | .in(in0) | |
|  | | | ,.out(out1[6:0]) | |
|  | | | ,.clk(clk) | |
|  | | | ,.set(set) | |
|  | | | ,.rst(rst) | |
|  | | | ,.en(en) | |
|  | | | ,.on(on) | |
|  | | | ); | |
|  | | | LED F( | |
|  | | | .in(in1) | |
|  | | | ,.out(out2[6:0]) | |
|  | | | ,.clk(clk) | |
|  | | | ,.set(set) | |
|  | | | ,.rst(rst) | |
|  | | | ,.en(en) | |
|  | | | ,.on(on) | |
|  | | | ); | |
|  | | |  | |
|  | | | endmodule | |

