**Pre-lab for Lab3**

Xiaoyu Zhou, 1004081147

**Part I**

1. Draw a schematic.

图片包含 文字, 白板

描述已自动生成

1. Write Verilog code for a 7-to-1 multiplexer

module mux(LEDR, SW);

input [9:0] SW;

output [9:0] LEDR;

mux1 u1(

.in1(SW[0]),

.in2(SW[1]),

.in3(SW[2]),

.in4(SW[3]),

.in5(SW[4]),

.in6(SW[5]),

.in7(SW[6]),

.m(SW[9:7]),

.out1(LEDR[0])

);

endmodule

module mux1(in1, in2, in3, in4, in5, in6, in7, m, out1);

input in1;

input in2;

input in3;

input in4;

input in5;

input in6;

input in7;

input m;

output out1; //output

reg out1;

always @(\*)

begin

case(m)

3'b000: out1 = in1;

3'b001: out1 = in2;

3'b010: out1 = in3;

3'b011: out1 = in4;

3'b100: out1 = in5;

3'b101: out1 = in6;

3'b111: out1 = in7;

default: out1 = in1;

endcase

end

endmodule

1. Simulate the circuit.

图片包含 屏幕截图

描述已自动生成

**Part II**

1. Draw a schematic.

图片包含 文字, 白板

描述已自动生成

1. Write a Verilog module for the full adder subcircuit and write another Verilog module that instantiates four instances of this full adder.

module mux(LEDR, SW);

input [9:0] SW;

output [9:0] LEDR;

wire connect1, connect2, connect3;

full\_adder a(

.A(SW[1]),

.B(SW[2]),

.cin(SW[0]),

.S(LEDR[0]),

.cout(connect1)

);

full\_adder b(

.A(SW[3]),

.B(SW[4]),

.cin(connect1),

.S(LEDR[1]),

.cout(connect2)

);

full\_adder c(

.A(SW[5]),

.B(SW[6]),

.cin(connect2),

.S(LEDR[2]),

.cout(connect3)

);

full\_adder d(

.A(SW[7]),

.B(SW[8]),

.cin(connect3),

.S(LEDR[3]),

.cout(LEDR[4])

);

endmodule

module full\_adder(A, B, cin, S, cout);

input A;

input B;

input cin;

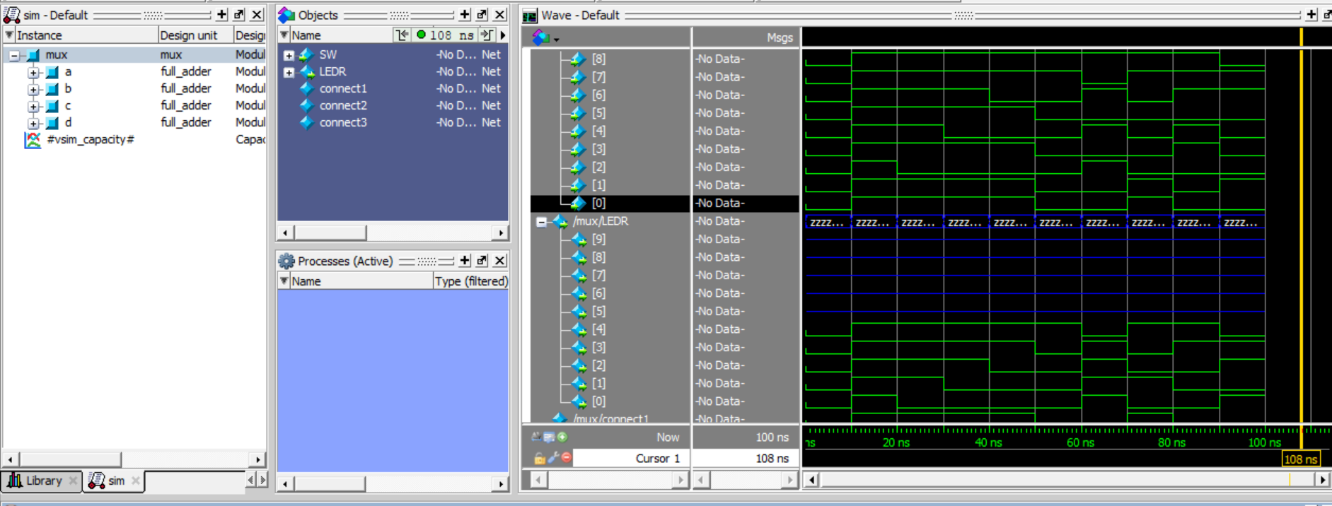
output S;

output cout;

assign cout = A&B | A&cin | B&cin;

assign S = A ^ B ^ cin;

endmodule

1. Simulate the circuit.

**Part III**

1. Draw a schematic.

图片包含 文字, 白板

描述已自动生成

1. Write a Verilog module for the full adder subcircuit and write another Verilog module that instantiates four instances of this full adder.

module mux(LEDR, SW, KEY, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);

input [9:0] SW;

input [2:0] KEY;

output [9:0] LEDR;

output [6:0] HEX0;

output [6:0] HEX1;

output [6:0] HEX2;

output [6:0] HEX3;

output [6:0] HEX4;

output [6:0] HEX5;

trans u1(

.A(SW[7:4]),

.B(SW[3:0]),

.in1(KEY[2:0]),

.out1(LEDR[7:0])

);

hex u2(

.in2(LEDR[3]),

.in3(LEDR[2]),

.in4(LEDR[1]),

.in5(LEDR[0]),

.o2(HEX4[6]),

.o3(HEX4[5]),

.o4(HEX4[4]),

.o5(HEX4[3]),

.o6(HEX4[2]),

.o7(HEX4[1]),

.o8(HEX4[0])

);

hex u3(

.in2(LEDR[7]),

.in3(LEDR[6]),

.in4(LEDR[5]),

.in5(LEDR[4]),

.o2(HEX5[6]),

.o3(HEX5[5]),

.o4(HEX5[4]),

.o5(HEX5[3]),

.o6(HEX5[2]),

.o7(HEX5[1]),

.o8(HEX5[0])

);

hex A(

.in2(SW[7]),

.in3(SW[6]),

.in4(SW[5]),

.in5(SW[4]),

.o2(HEX2[6]),

.o3(HEX2[5]),

.o4(HEX2[4]),

.o5(HEX2[3]),

.o6(HEX2[2]),

.o7(HEX2[1]),

.o8(HEX2[0])

);

hex B(

.in2(SW[3]),

.in3(SW[2]),

.in4(SW[1]),

.in5(SW[0]),

.o2(HEX0[6]),

.o3(HEX0[5]),

.o4(HEX0[4]),

.o5(HEX0[3]),

.o6(HEX0[2]),

.o7(HEX0[1]),

.o8(HEX0[0])

);

assign HEX1[6:0] = 0;

assign HEX3[6:0] = 0;

endmodule

module trans(A, B, in1, out1);

parameter Width = 8;

parameter Narrow = 4;

parameter W = 3;

input [Narrow-1:0] A;

input [Narrow-1:0] B;

input [W-1:0] in1;

output [Width-1:0]out1;

reg [Width-1:0]out1;

always @(\*)

begin

case(in1)

3'b000: out1 = {A, B};

3'b001: out1 = {A | B, A ^ B};

3'b010: out1 = {A&B | A&0 | B&0, A ^ B ^ 0};

3'b011: out1 = A + B;

3'b100: out1 = {A&1 | A&0 | 1&0, A ^ 1 ^ 0};

3'b101: out1 = |A | |B;

default: out1 = 8'b0000\_0000;

endcase

end

endmodule

module hex(in2, in3, in4, in5, o2, o3, o4, o5, o6, o7, o8);

input in2;

input in3;

input in4;

input in5;

output o2;

output o3;

output o4;

output o5;

output o6;

output o7;

output o8;

hex0 u1(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out1(o2)

);

hex1 u2(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out2(o3)

);

hex2 u3(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out3(o4)

);

hex3 u4(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out4(o5)

);

hex4 u5(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out5(o6)

);

hex5 u6(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out6(o7)

);

hex6 u7(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out7(o8)

);

endmodule

module hex0(c3, c2, c1, c0, out1);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out1; //output

assign out1 = ~c3 & c2 & ~c1 & ~c0 | c3 & c2 & ~c1 & c0 | c3 & ~c2 & c1 & c0 | ~c3 & ~c2 & ~c1 & c0;

endmodule

module hex1(c3, c2, c1, c0, out2);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out2; //output

assign out2 = c2 & c1 & ~c0 | c3 & c1 & c0 | ~c3 & c2 & ~c1 & c0 | c3 & c2 & ~c0;

endmodule

module hex2(c3, c2, c1, c0, out3);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out3; //output

assign out3 = ~c3 & ~c2 & c1 & ~c0 | c3 & c2 & c1 | c3 & c2 & ~c0;

endmodule

module hex3(c3, c2, c1, c0, out4);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out4; //output

assign out4 = ~c3 & c2 & ~c1 & ~c0 | ~c2 & ~c1 & c0 | c2 & c1 & c0 | c3 & ~c2 & c1 & ~c0;

endmodule

module hex4(c3, c2, c1, c0, out5);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out5; //output

assign out5 = ~c3 & c2 & ~c1 | ~c2 & ~c1 & c0 | ~c3 & c0;

endmodule

module hex5(c3, c2, c1, c0, out6);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out6; //output

assign out6 = ~c3 & ~c2 & c0 | ~c3 & ~c2 & c1 | ~c3 & c1 & c0 | c3 & c2 & ~c1 & c0;

endmodule

module hex6(c3, c2, c1, c0, out7);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out7; //output

assign out7 = ~c3 & ~c2 & ~c1 | c3 & c2 & ~c1 & ~c0 | ~c3 & c2 & c1 & c0;

endmodule

1. Simulate the circuit.

图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成