**Logo

Description automatically generated San Francisco Bay University**

**EE461 Verilog-HDL**

**2023 Spring Midterm Exam**

**Student Name: Simon Gezae Student ID:19830**

1. Identify the bugs in the following snippet of source code and fix them.

*module module;*

*time a;*

*reg b, c;*

*assign begin*

*a = (b)? b : c;*

*end*

*endmodule*

*Ans: module is a keyword so it cannot be used as module name. the left side of the assignment (a) should be wire data type here it is time. Can be fixed like this*

*module module1;*

*wire a;*

*reg b, c;*

*assign a = (b)? b : c;*

*endmodule*

1. What is delay time for the following program?

*`timescale 100ps / 100ps*

*module sampleDesign (z,x1,x2);*

*input x1, x2;*

*output z;*

*nor #3.1415 (z, x1, x2);*

*endmodule*

*Ans:*

*#3.1415*

*Delay = 3.1415\*100 %100 ps*

*Delay = 314.15 %100ps*

*Delay = 300ps*

1. What are the values of (a = = = b) and (a ! = = b) if a = 4’b01xz, b = 4’b xz10?

Ans: (a = = = b) Ans: 0

(a ! = = b) Ans : 1

1. When a is 4’bxz01, what are values of ~a and !a ?

Ans: ~a = xx10, !a = 0

1. Please calculate b and c’s values.

*module test;*

*reg[3:0] a;*

*wire[3:0] b, c;*

*assign b = a << 2;*

*assign c = 2 << a;*

*initial begin*

*a=4’b0010;*

*#10 $finish;*

*end*

*endmodule*

*Ans: B = 1000*

*C = 1000*

1. Assuming that there are three students to vote whether they eat in the restaurant ABC or not. If the minority of them agrees to go, the rest of them will follow. Write UDP to implement it.

*primitive agree(going, std1,std2,std3 );*

*output going;*

*input std1;*

*input std2;*

*input std3;*

*// Here is your code*

*Endprimitive*

*Ans:*

*primitive agree(going, std1,std2,std3);*

*output going;*

*input std1;*

*input std2;*

*input std3;*

*// reg going;*

*table*

*//std1 std2 std3 : A*

*0 0 0:0; //if none agree they are not going*

*0 0 1:1;*

*0 1 0:1;*

*1 0 0:1;*

*1 1 ?:0;*

*1 ? 1:0;*

*? 1 1:0;*

*endtable*

*endprimitive*

*module aggretb();*

*reg std1, std2, std3;*

*wire going;*

*agree u1(going,*

*std1,*

*std2,*

*std3*

*);*

*initial begin*

*$display("std1\tstd2\tstd3 |\tgoing");*

*$monitor("%b\t%b\t%b |\t%b",std1,std2,std3,going);*

*std1 = 0; std2 = 0; std3 = 0;*

*#2 std1 = 1;*

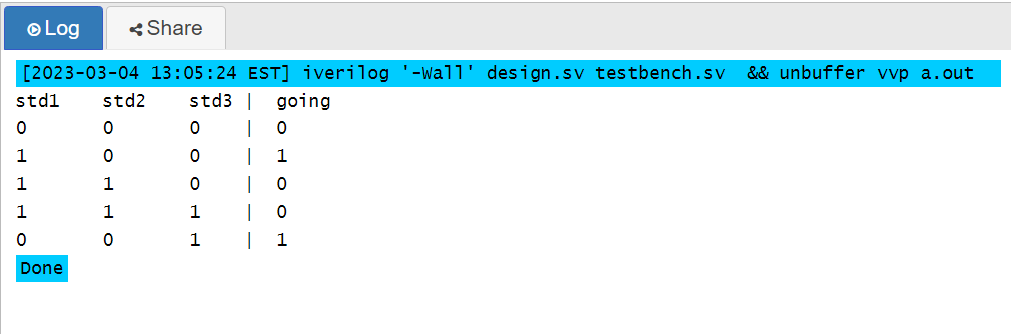
*#2 std2 = 1;*

*#2 std3 = 1;*

*#2 std1 = 0; std2 = 0;*

*end*

*endmodule*



Link: <https://edaplayground.com/x/vECQ>

1. Fix the bugs in the following RTL module

*module test;*

*wire a;*

*wire[1:0] b;*

*wire[2:0] c;*

*wire[3:0] d;*

*always@(a or b or posedge c)begin*

*c <= a+b;*

*d <= b+ c;*

*assign a <= a+d;*

*end*

*endmodule*

Ans: we are not supposed to use combinational and sequential trigger in the same sensitivity c is used here as clk but clk should be reg data type. We shouldn’t use assign inside always block.

module test;

reg a;

reg[1:0] b;

reg[2:0] c;

reg[3:0] d;

always@(a or b or c)begin

c <= a+b;

d <= b+ c;

a <= a+d;

end

endmodule

This program runs without any error

1. Design a module in RTL (not gate level) to calculate output for two inputs x and y in 32 bits with using only adders and inverters, no more else. And ignore overflow issue.

out = 14x + 16y

Ans:

module question8();

reg [31:0] x;

reg [31:0] y;

wire [31:0] out;

wire [31:0] a1;

wire [31:0] a2; wire [31:0] a3; wire [31:0] a4; wire [31:0] a5;

wire [31:0] a6; wire [31:0] a7; wire [31:0] a8; wire [31:0] a9;

assign a1 = x+x; //a=2x

assign a2 = a1+a1; //4x

assign a3 = a2+a2; //8x

assign a4 = a3+a3; //16x

assign a5 = a4-a1; //14x

assign a6 = y+y;

assign a7 = a6+a6; //4y

assign a8 = a7+a7; //8y

assign a9 = a8+a8; //16y

assign out = a5+a9; //14x+16y

initial begin

x = 32'b0;

y = 32'b1;

#2 $display("X: %b Y: %b out: %b", x, y, out);

end

endmodule

<https://edaplayground.com/x/dcge>

1. What is different between continuous assignment and always block in the following code?

*module test;*

*wire a;*

*reg b, c, d;*

*assign a = (b)? c: d;*

*endmodule*

*module test;*

*reg a;*

*reg b, c, d;*

*always@(\*)begin*

*if(b) a = c;*

*else a = d;*

*endmodule*

*Ans:*

*The first module is continuous assignment and it is executed when ever b c or d changes. The second module is always block the \* makes it execute when c or d or b change when b is true a is c else a is d. second module need end for the always block*

1. Design Gray counter to count 6 in RTL with clock signal.

Gray Bin

0 0000 0000

1 0001 0001

2 0011 0010

3 0010 0011

4 0110 0100

5 0111 0101

Ans: in github link <https://github.com/sami-dotcom/midterm_class.git>

1. Correct the following bad design module.

*module test;*

*reg a, b, c, en;*

*always@(a or en)begin*

*if(en) c = a | b;*

*end*

*endmodule*

*Ans:*

*module test;*

*reg a, b, c, en;*

*always@(a or en or b)begin*

*if(en) c = a | b;*

*end*

*endmodule*