Assignment #3

**IV. Exercises**

1. Are there something wrong in the following codes? And fix them.

1. always@(A or B) begin

if(A) C = B ^ A;

else C = D & E;

F = C | A;

end

**Ans:** A, B, D, E need to be inside always bracket as in this case C will only be solved when A and B are changed. To fix it we need to include D and E in parenthesis. We don’t need to include C or F as they can be triggered if one of the rest changed.

always@(A or B or D or E) begin

if(A) C = B ^ A;

else C = D & E;

F = C | A;

end

2. always@(B)

C = |B;

always@(E)

C = ^E;

**Ans:** In this code **we have race condition. Since always blocks run at the same time B and E can be assigned to C. One way to solve it is to use one always. And if both change c will take E value.**

always@(B or E)

C = |B;

C = ^E;

3. always@(posedge clock)

if(A) Q <= D;

always@(Q or E)

case (Q)

0: F = E;

default: F = 1;

endcase

clock : 00001111

D : xxxxxxxx

E : 11110000

Q :

F :

4. module top;

wire B;

bar u1 (A,B);

bar u2 (C,B);

endmodule

module bar (input D; output wire E);

assign E=~D;

endmodule

5. module foo(input A,B; output reg E);

wire C,D;

always@(posedge clock) E=B&D;

assign C=A^D;

assign D=C|B;

endmodule

Ans: c and d are depending in each other d can use c because it is executed first.

1. According to the modules IfMux8 and CaseMux8, draw the corresponding circuits and compare them.

Diagram, schematic

Description automatically generated

Diagram

Description automatically generated

IfMux8 when compared to caseMux it has higher delay length as the critical path is i7 to y whereas in caseMux i7 and i0 has the same delay.