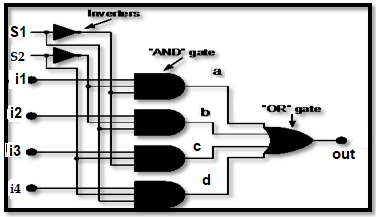
**LAB # 06**

**Objective:** INTRODUCTION TO BEHAVIOUR LEVEL MODELING.

**Task1:** To design 4 x 1 multiplexer in verilog by using case statement in Behaviour level modeling.

**Circuit diagram:**

**Coding:**

//Design Module

module mux(out,i1,i2,i3,i4,s1,s2);

input i1,i2,i3,i4,s1,s2;

output out;

reg out;

always @(i1 or i2 or i3 or i4 or s1 or s2)

begin

case ({s1,s2})

2'b00: out=i1;

2'b01: out=i2;

2'b10: out=i3;

2'b11: out=i4;

endcase

end

endmodule

//Stimullux Module

module aazib;

reg i1,i2,i3,i4,s1,s2;

wire out;

mux a1(out,i1,i2,i3,i4,s1,s2);

initial

begin

i1=1'b0; i2=1'b0; i3=1'b0; i4=1'b1; s1=1'b0; s2=1'b0;

#20

i1=1'b1; i2=1'b0; i3=1'b0; i4=1'b0; s1=1'b0; s2=1'b1;

#20

i1=1'b0; i2=1'b1; i3=1'b0; i4=1'b1; s1=1'b1; s2=1'b0;

#20

i1=1'b1; i2=1'b1; i3=1'b0; i4=1'b0; s1=1'b1; s2=1'b1;

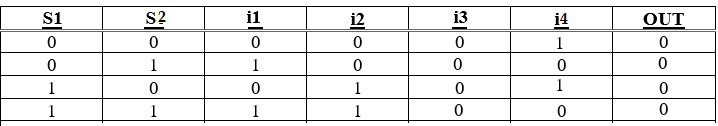
#20

$finish;

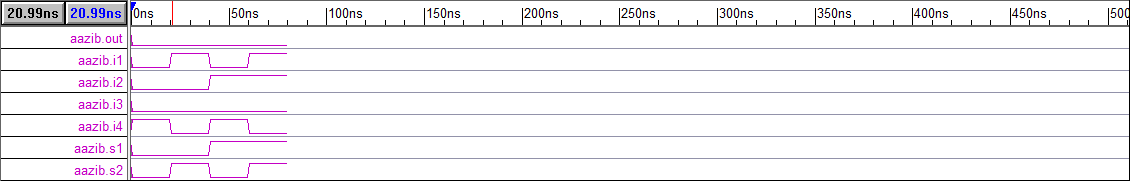
end

endmodule

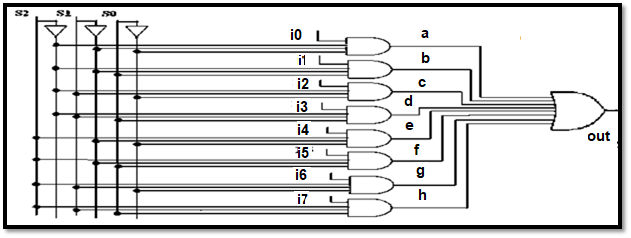
**Truth Table:**

****

**Results (Timing diagram) :**



**Task2:** To design 8x1 mux in verilog by using Behaviour level modeling.

**Circuit diagram:**

**Coding:**

//Design Module

module mux(out,i1,i2,i3,i4,i5,i6,i7,i8,s1,s2,s3);

input i1,i2,i3,i4,i5,i6,i7,i8,s1,s2,s3;

output out;

reg out;

always @(i1 or i2 or i3 or i4 or i5 or i6 or i7 or i8 or s1 or s2 or s3)

begin

case ({s1,s2,s3})

3'b000: out=i1;

3'b001: out=i2;

3'b010: out=i3;

3'b011: out=i4;

3'b100: out=i5;

3'b101: out=i6;

3'b110: out=i7;

3'b111: out=i8;

endcase

end

endmodule

//Stimullux Module

module aazib;

reg i1,i2,i3,i4,i5,i6,i7,i8,s1,s2,s3;

wire out;

mux a1(out,i1,i2,i3,i4,i5,i6,i7,i8,s1,s2,s3);

initial

begin

i1=1'b0; i2=1'b0; i3=1'b0; i4=1'b0; i5=1'b1; i6=1'b1; i7=1'b1; i8=1'b1; s1=1'b0; s2=1'b0; s3=1'b0;

#20

i1=1'b1; i2=1'b0; i3=1'b0; i4=1'b0; i5=1'b1; i6=1'b0; i7=1'b1; i8=1'b1; s1=1'b0; s2=1'b0; s3=1'b1;

#20

i1=1'b0; i2=1'b1; i3=1'b0; i4=1'b0; i5=1'b1; i6=1'b1; i7=1'b0; i8=1'b1; s1=1'b0; s2=1'b1; s3=1'b0;

#20

i1=1'b1; i2=1'b1; i3=1'b0; i4=1'b0; i5=1'b1; i6=1'b0; i7=1'b0; i8=1'b1; s1=1'b0; s2=1'b1; s3=1'b1;

#20

i1=1'b0; i2=1'b0; i3=1'b1; i4=1'b0; i5=1'b1; i6=1'b1; i7=1'b1; i8=1'b0; s1=1'b1; s2=1'b0; s3=1'b0;

#20

i1=1'b1; i2=1'b0; i3=1'b1; i4=1'b0; i5=1'b1; i6=1'b0; i7=1'b1; i8=1'b0; s1=1'b1; s2=1'b0; s3=1'b1;

#20

i1=1'b0; i2=1'b1; i3=1'b1; i4=1'b0; i5=1'b1; i6=1'b1; i7=1'b0; i8=1'b0; s1=1'b1; s2=1'b1; s3=1'b0;

#20

i1=1'b1; i2=1'b1; i3=1'b1; i4=1'b0; i5=1'b1; i6=1'b0; i7=1'b0; i8=1'b0; s1=1'b1; s2=1'b1; s3=1'b1;

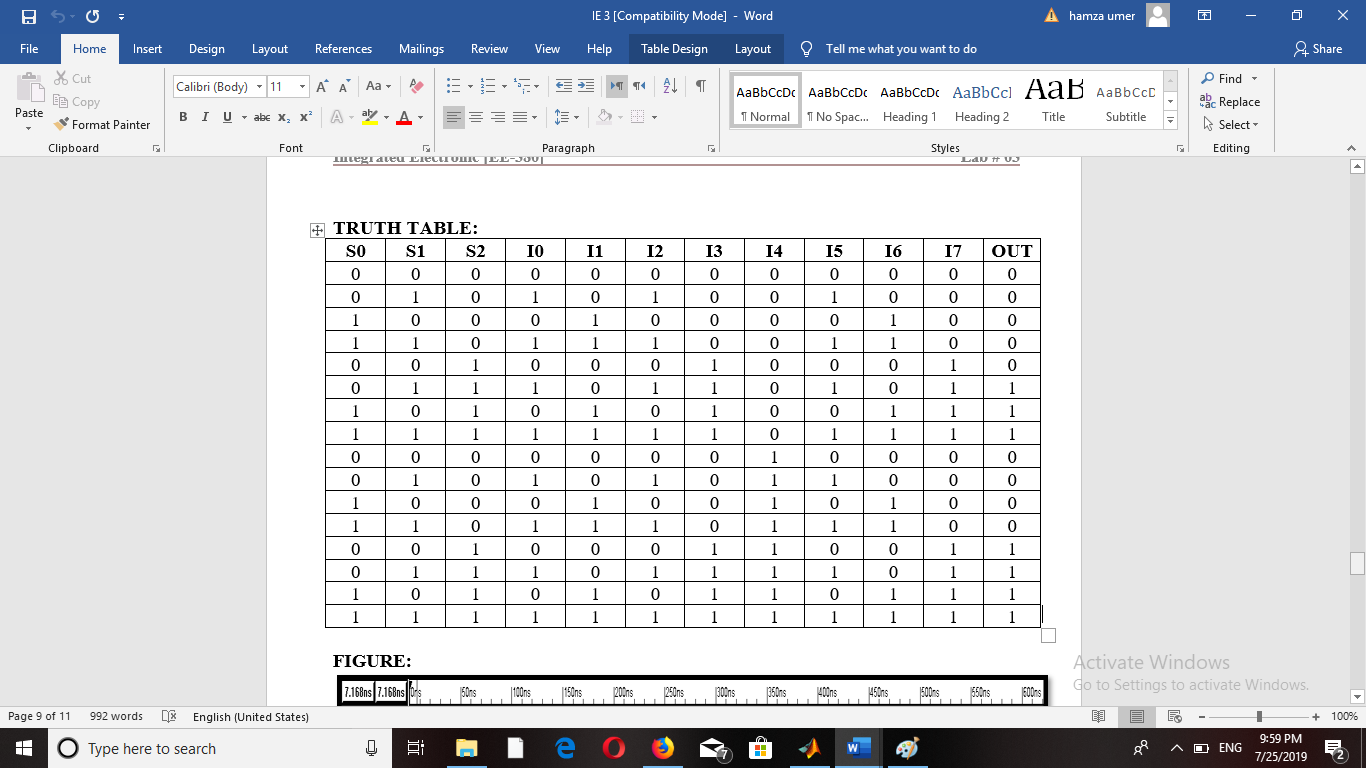
#20

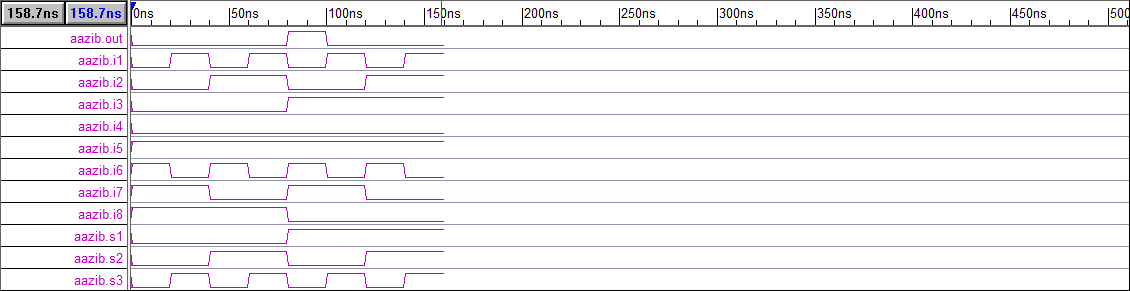
$finish;

end

endmodule

**Truth Table:**



**Results (Timing diagram) :**

**Task3:** Design a clock with time period 40ns & duty cycle 25% by using always & initial statement. The value of clk at time = 0 should be initializing to 0.

**Coding:**

//Stimullux Module

module aazib;

reg clk;

initial

clk=1'b0;

always

begin

#30

clk=~clk;

#10

clk=~clk;

end

initial

#200

$finish;

Endmodule

**Results (Timing diagram) :**



**Task4:** Using case statement, Design a function ALU that takes 4-bit i/p “A” & “B” & 3-bit i/p signal “Select” & give 5-bit o/p “OUT” .The ALU implements the following function on 3 -bit i/p select.

**Coding:**

//Design Module

module ALU(out,a,b,select);

input [3:0]a,b;

input [2:0]select;

output [4:0]out;

reg [4:0]out;

always @(a or b or select)

begin

case ({select})

3'b000: out=a;

3'b001: out=a+b;

3'b010: out=a-b;

3'b011: out=a/b;

3'b100: out=a%b;

3'b101: out=a<<1;

3'b110: out=a>>1;

3'b111: out=a>b;

endcase

end

endmodule

//Stimullux Module

module aazib;

reg [3:0]a,b;

reg [2:0]select;

wire out;

ALU a1(out,a,b,select);

initial

begin

a=4'h9; b=4'h3; select=3'b000;

#20

a=4'h9; b=4'h3; select=3'b001;

#20

a=4'h9; b=4'h3; select=3'b010;

#20

a=4'h9; b=4'h3; select=3'b011;

#20

a=4'h9; b=4'h3; select=3'b100;

#20

a=4'h9; b=4'h3; select=3'b101;

#20

a=4'h9; b=4'h3; select=3'b110;

#20

a=4'h9; b=4'h3; select=3'b111;

#20

$finish;

end

endmodule

**Results (Timing diagram) :**

