

**LAB#04**

**OBJECT:** Introduction to Data Modeling

**TASK1:** To design AND,OR,NAND,NOT&XOR gate in verilog by using Data Follow Modeling

**AND GATE**

**CODING:**

//Design Module

module dand(c,a,b);

input a,b;

output c;

assign c=a&b;

endmodule

//Stimullux Module

module aazib;

reg a,b;

wire c;

dand f1(c,a,b);

initial

begin

a=1'b0; b=1'b0;

#20

a=1'b0; b=1'b1;

#20

a=1'b1; b=1'b0;

#20

a=1'b1; b=1'b1;

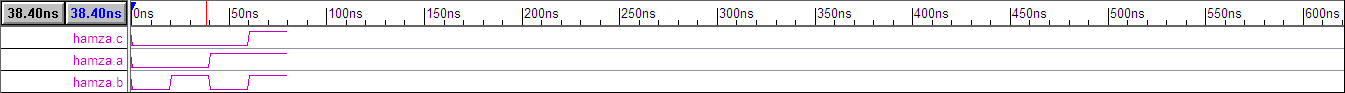
#20

$finish;

end

endmodule

**RESULT:**



**OR CODING**

//Design Module

module dor(c,a,b);

input a,b;

output c;

assign c=a|b;

endmodule

//Stimullux Module

module aazib;

reg a,b;

wire c;

dor f1(c,a,b);

initial

begin

a=1'b0; b=1'b0;

#20

a=1'b0; b=1'b1;

#20

a=1'b1; b=1'b0;

#20

a=1'b1; b=1'b1;

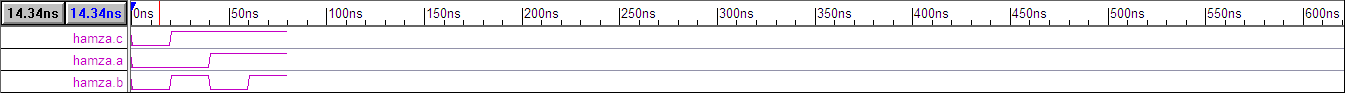
#20

$finish;

end

endmodule

**RESULT:**



**NOT CODING**

//Design Module

module dnot(out,a);

input a;

output out;

assign out=!a;

endmodule

//Stimullux Module

module aazib;

reg a;

wire out;

dnot f1(out,a);

initial

begin

a=1'b0;

#20

a=1'b1;

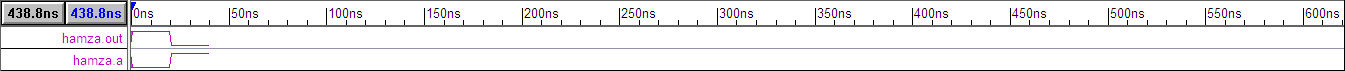
#20

$finish;

end

endmodule

**RESULT:**

****

**NAND GATE**

//Design Module

module dnand(c,a,b);

input a,b;

output c;

assign c=~(a&b);

endmodule

//Stimullux Module

module aazib;

reg a,b;

wire c;

dnand f1(c,a,b);

initial

begin

a=1'b0; b=1'b0;

#20

a=1'b0; b=1'b1;

#20

a=1'b1; b=1'b0;

#20

a=1'b1; b=1'b1;

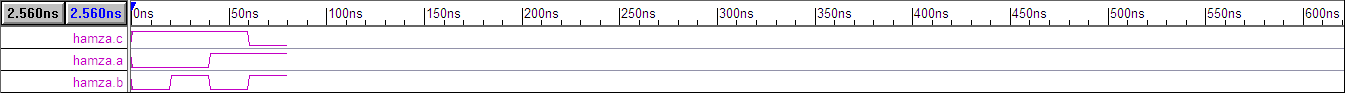
#20

$finish;

end

endmodule

**RESULT:**



**TASK2: D-Flip FLOP**

**CODING:**

//Design Module

module dff(q,qb,d,clk);

input d,clk;

output q,qb;

wire a,b,db;

assign a=~(d&clk);

assign q=~(a&qb);

assign b=~(db&clk);

assign qb=~(b&q);

assign db=!d;

endmodule

//Stimullux Module

module aazib;

reg d,clk;

wire q,qb;

dff f1(q,qb,d,clk);

initial

begin

d=1'b1; clk=1'b1;

#20

d=1'b1; clk=1'b1;

#20

d=1'b1; clk=1'b0;

#20

d=1'b0; clk=1'b1;

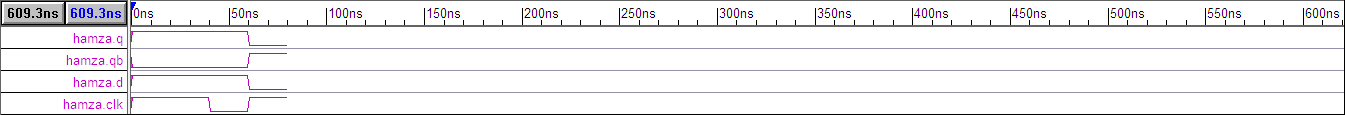
#20

$finish;

end

endmodule

**RESULT:**



**TASK3: One bit full adder**

**CODING:**

//Design Module

module obfa(sum,cout,a,b,cin);

input a,b,cin;

output sum,cout;

assign sum=(a&b&cin)|(!a&b&!cin)|(!a&!b&cin)|(a&!b&!cin);

assign cout=(a&b)|(b&cin)|(a&cin);

endmodule

//Stimullux Module

module aazib;

reg a,b,cin;

wire sum,cout;

obfa f1(sum,cout,a,b,cin);

initial

begin

a=1'b0; b=1'b0; cin=1'b0;

#20

a=1'b0; b=1'b1; cin=1'b1;

#20

a=1'b1; b=1'b0; cin=1'b1;

#20

a=1'b1; b=1'b1; cin=1'b1;

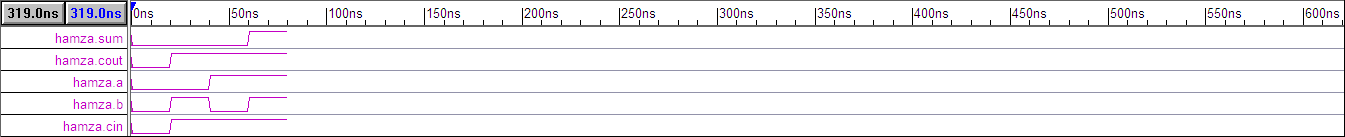
#20

$finish;

end

endmodule

**RESULT:**



**TASK4: Master-Slave j-k Flip Flop**

**CODING:**

//Design Module

module msjk(q,qb,j,k,clr,clk);

input j,k,clr,clk;

output q,qb;

wire a,b,c,x,y,z,clkb;

assign a=~(j&clk&clr&qb);

assign b=~(a&y);

assign c=~(b&clkb);

assign q=~(c&qb);

assign x=~(clk&k&q);

assign y=~(x&b&clr);

assign z=~(y&clkb);

assign qb=~(z&q&clr);

assign clkb=!clk;

endmodule

//Stimullux Module

module aazib;

reg j,k,clr,clk;

wire q,qb;

msjk f1(q,qb,j,k,clr,clk);

initial

clk=1'b0;

always #5

clk=~clk;

initial

begin

j=1'b0; k=1'b0; clr=1'b0;

#30

j=1'b0; k=1'b1; clr=1'b1;

#30

j=1'b1; k=1'b0; clr=1'b1;

#30

j=1'b1; k=1'b1; clr=1'b1;

#30

$finish;

end

endmodule

**RESULT:**

