**University of Engineering and Technology,Taxila**

**Department of Computer Engineering**

# Lab Report 03

**For the Course of DSD lab**

**Submitted By**:Muhammad Ibrahim (21-CP-26)

**Section:** Omega

## **Lab Instructor:**Sir Shahid Ali

**Course Instructor:**Dr.Abdul rehman aslam

**Date**: 26-01-24.

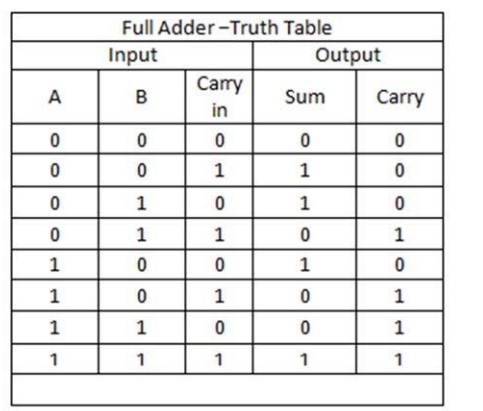
## **Course Title**: DSD Lab

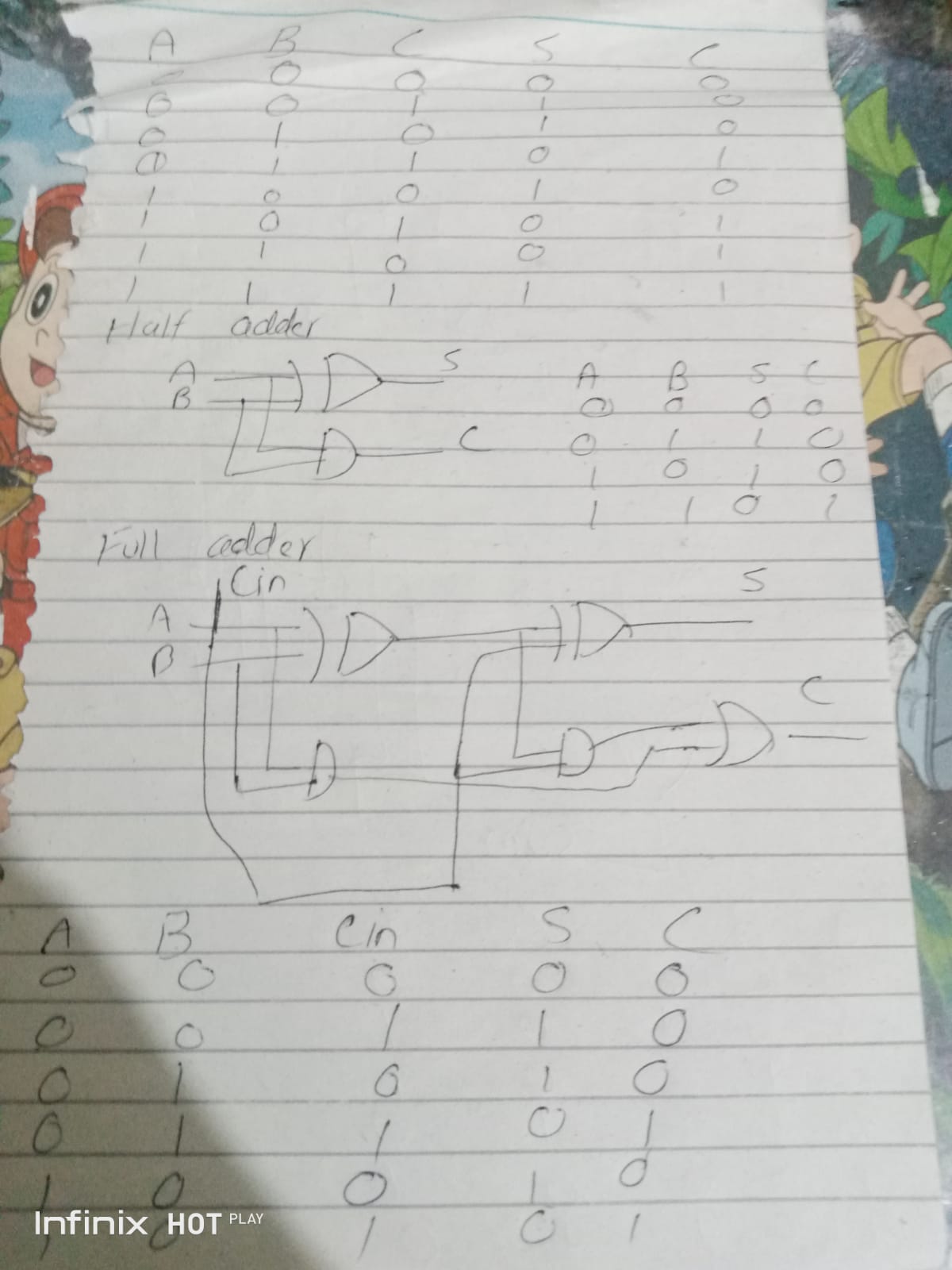
**Lab Tasks**

**Question:**

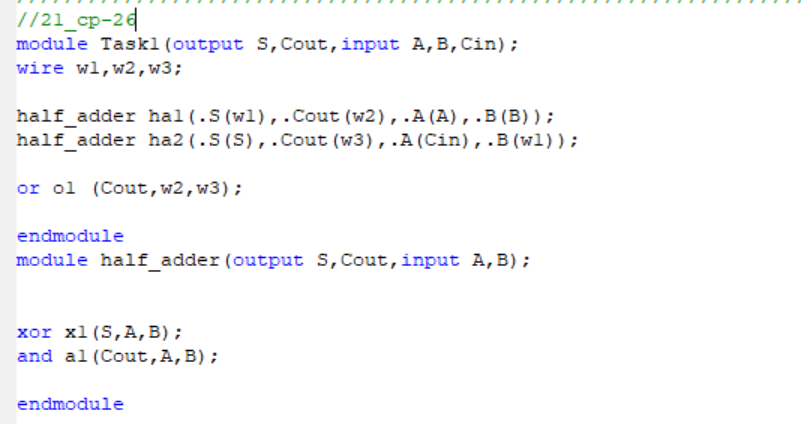
**Task 1:**

Design a full adder using two half adders using hierarchical modeling and other primitive logic gate (if required). Simulate and verify the following truth table in Xilinx ISE simulator.

**```````````````````````````**

****

**Code:**



//21\_cp-26

module Task1(output S,Cout,input A,B,Cin);

wire w1,w2,w3;

half\_adder ha1(.S(w1),.Cout(w2),.A(A),.B(B));

half\_adder ha2(.S(S),.Cout(w3),.A(Cin),.B(w1));

or o1 (Cout,w2,w3);

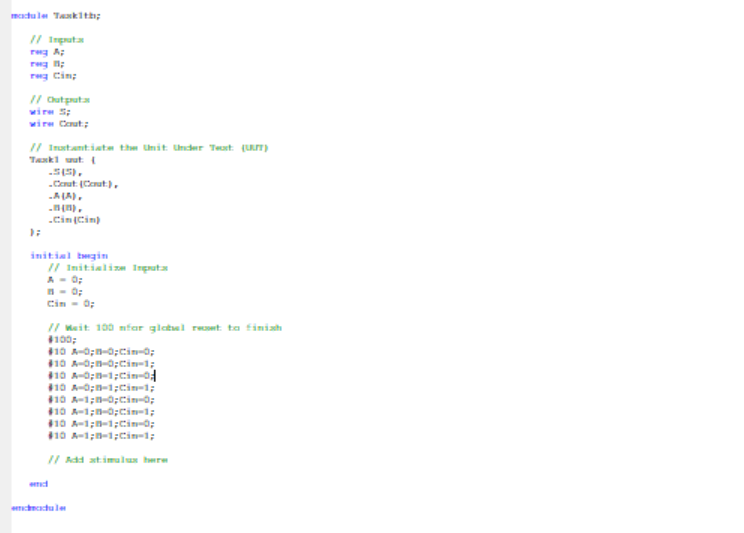
endmodule

module half\_adder(output S,Cout,input A,B);

xor x1(S,A,B);

and a1(Cout,A,B);

endmodule

**TestBench:** 

//21-CP-26

module Task1tb;

// Inputs

reg A;

reg B;

reg Cin;

// Outputs

wire S;

wire Cout;

// Instantiate the Unit Under Test (UUT)

Task1 uut (

.S(S),

.Cout(Cout),

.A(A),

.B(B),

.Cin(Cin)

);

initial begin

// Initialize Inputs

A = 0;

B = 0;

Cin = 0;

// Wait 100 nfor global reset to finish

#100;

#10 A=0;B=0;Cin=0;

#10 A=0;B=0;Cin=1;

#10 A=0;B=1;Cin=0;

#10 A=0;B=1;Cin=1;

#10 A=1;B=0;Cin=0;

#10 A=1;B=0;Cin=1;

#10 A=1;B=1;Cin=0;

#10 A=1;B=1;Cin=1;

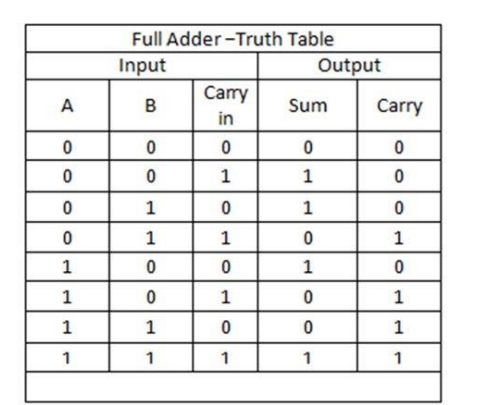
// Add stimulus here

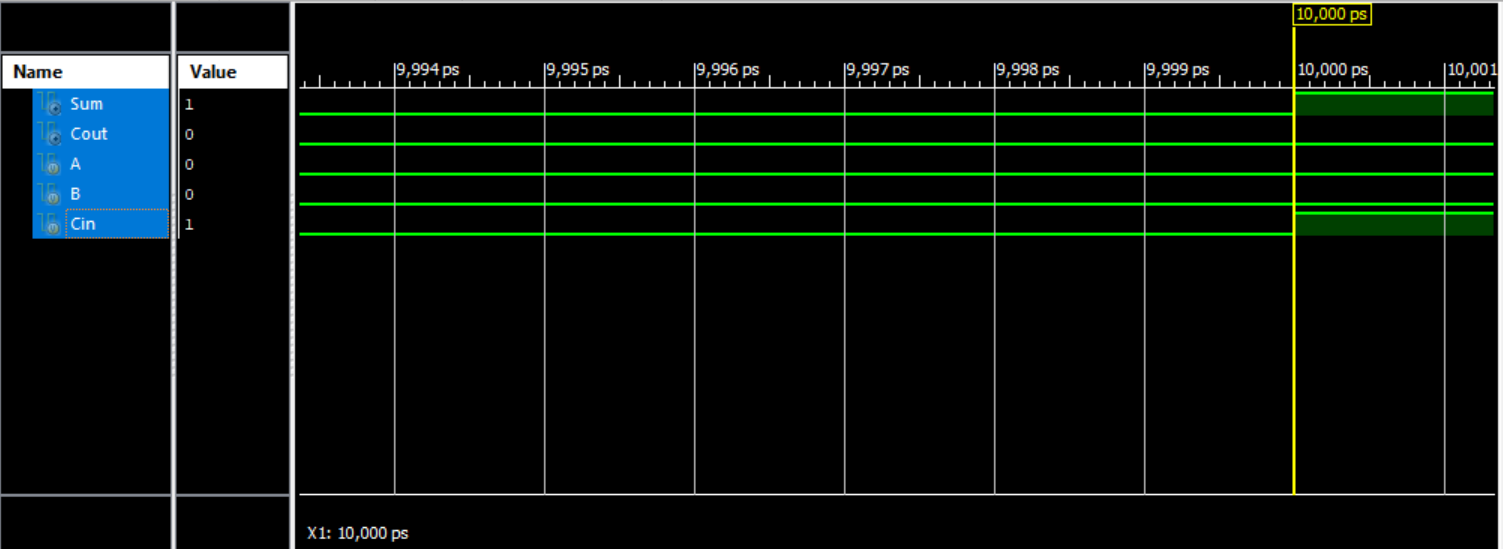
end

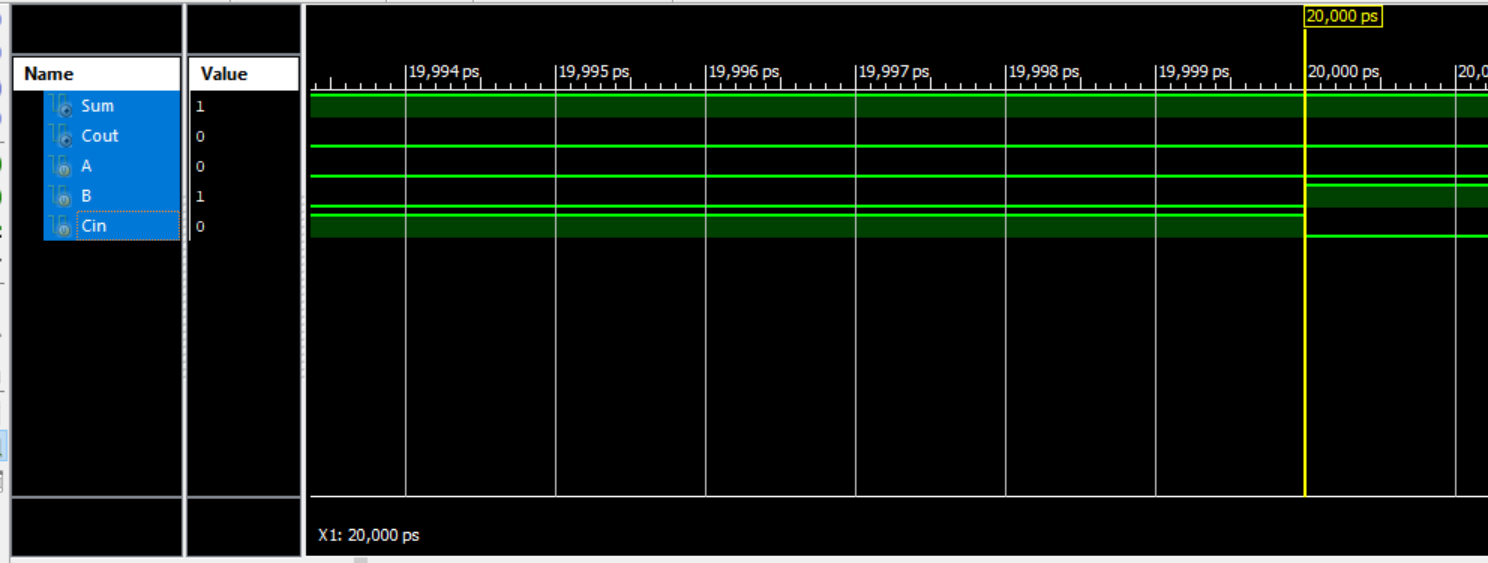
endmodule

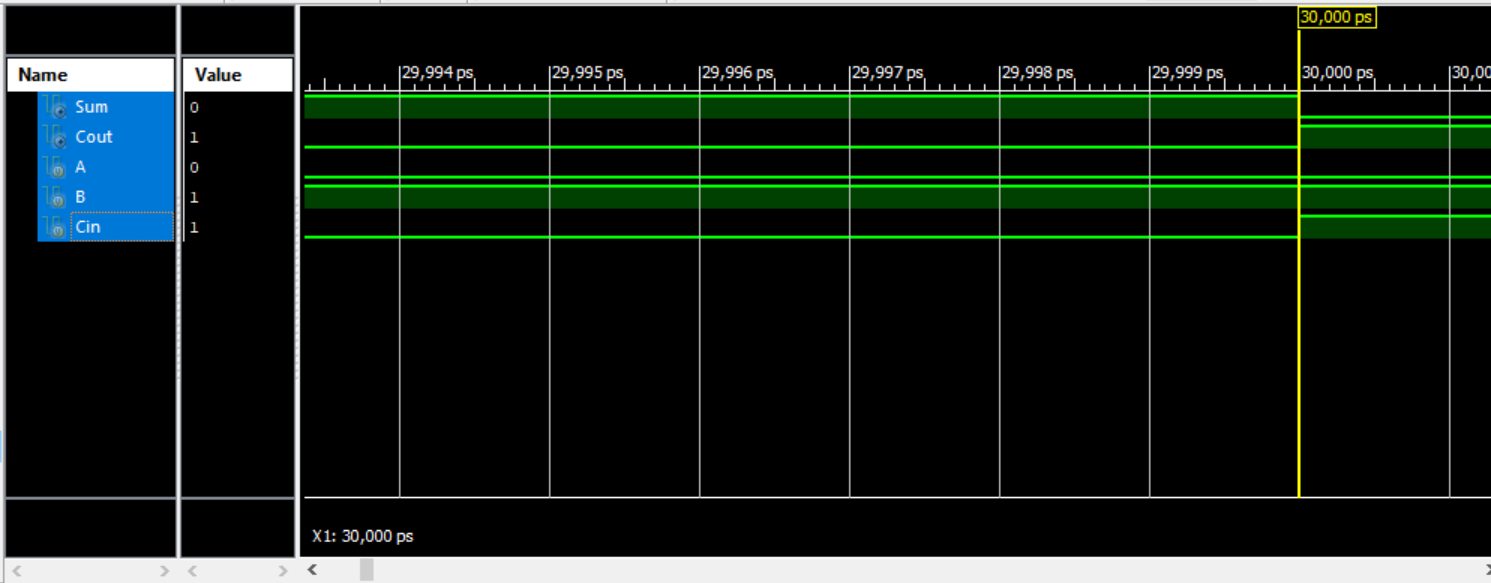
**Simulations :**

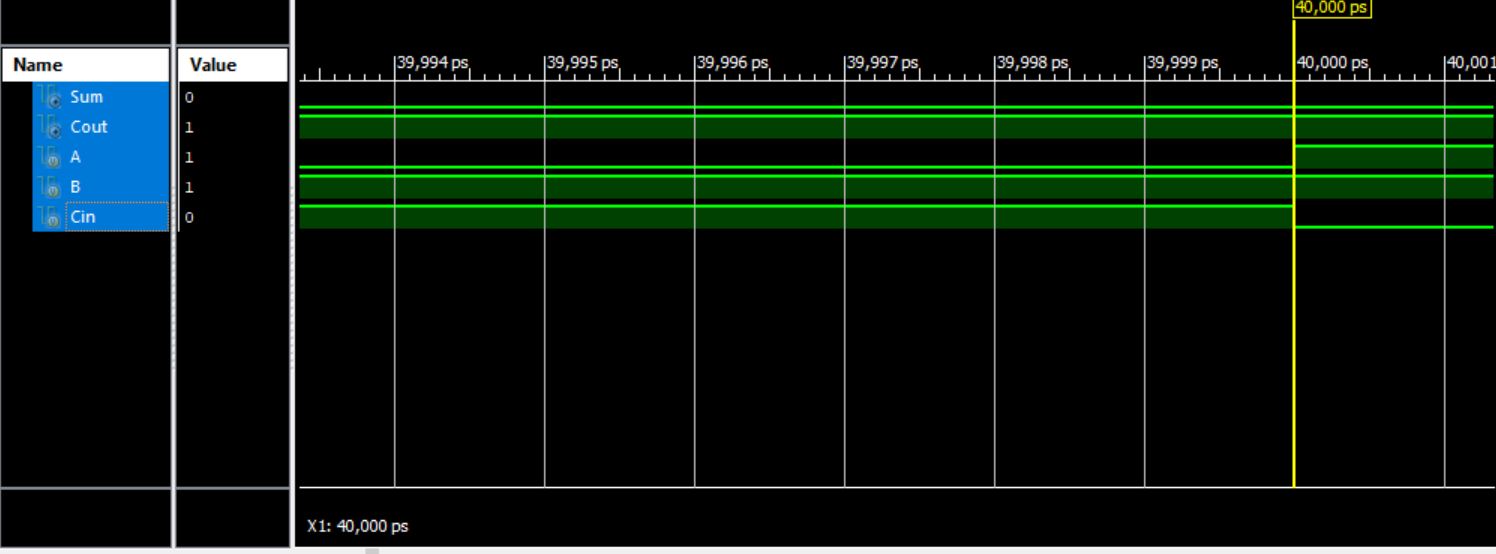
**As you can see the truth table verifies**

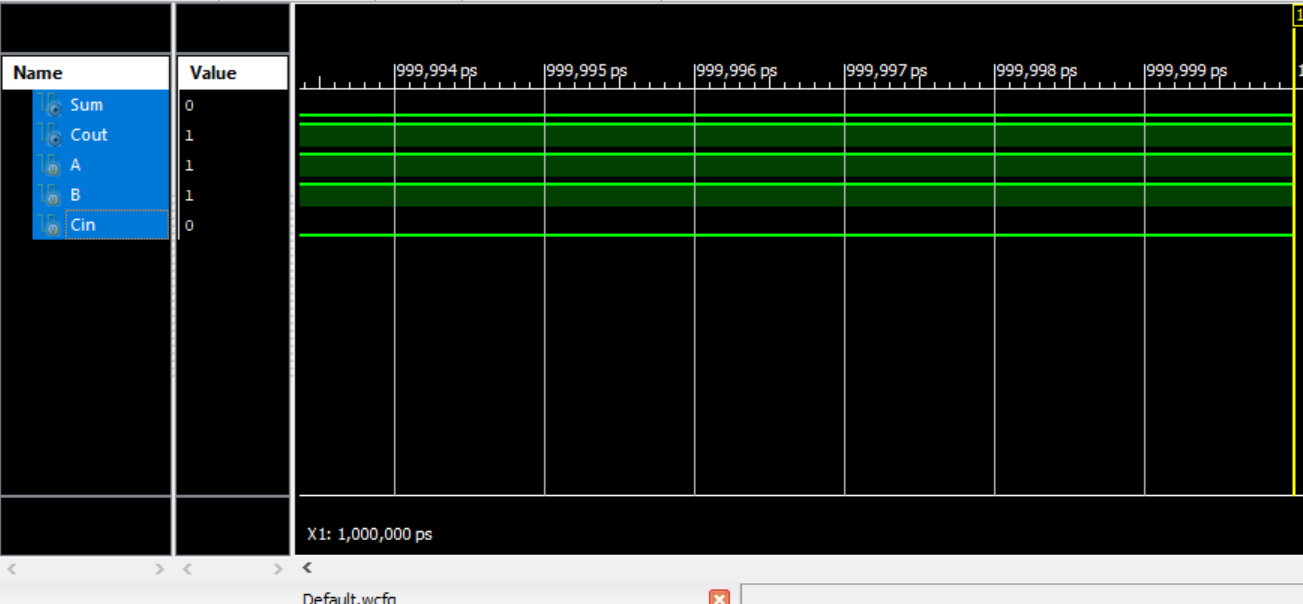


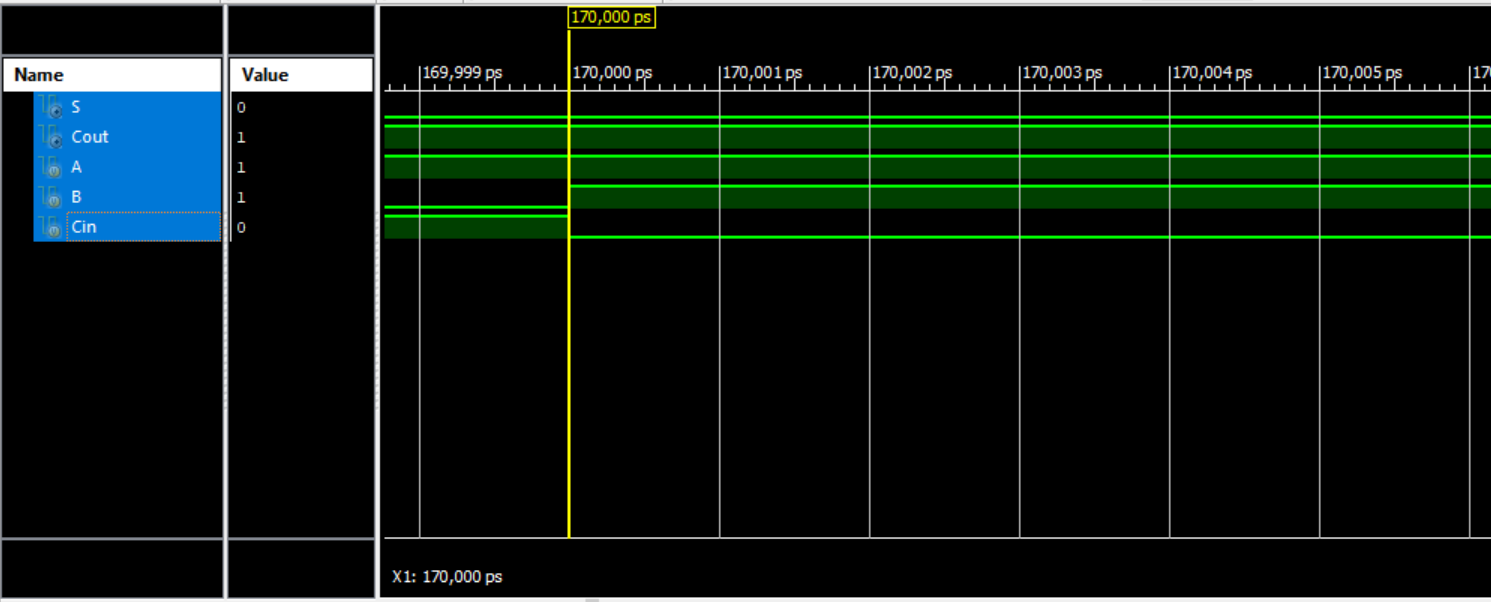










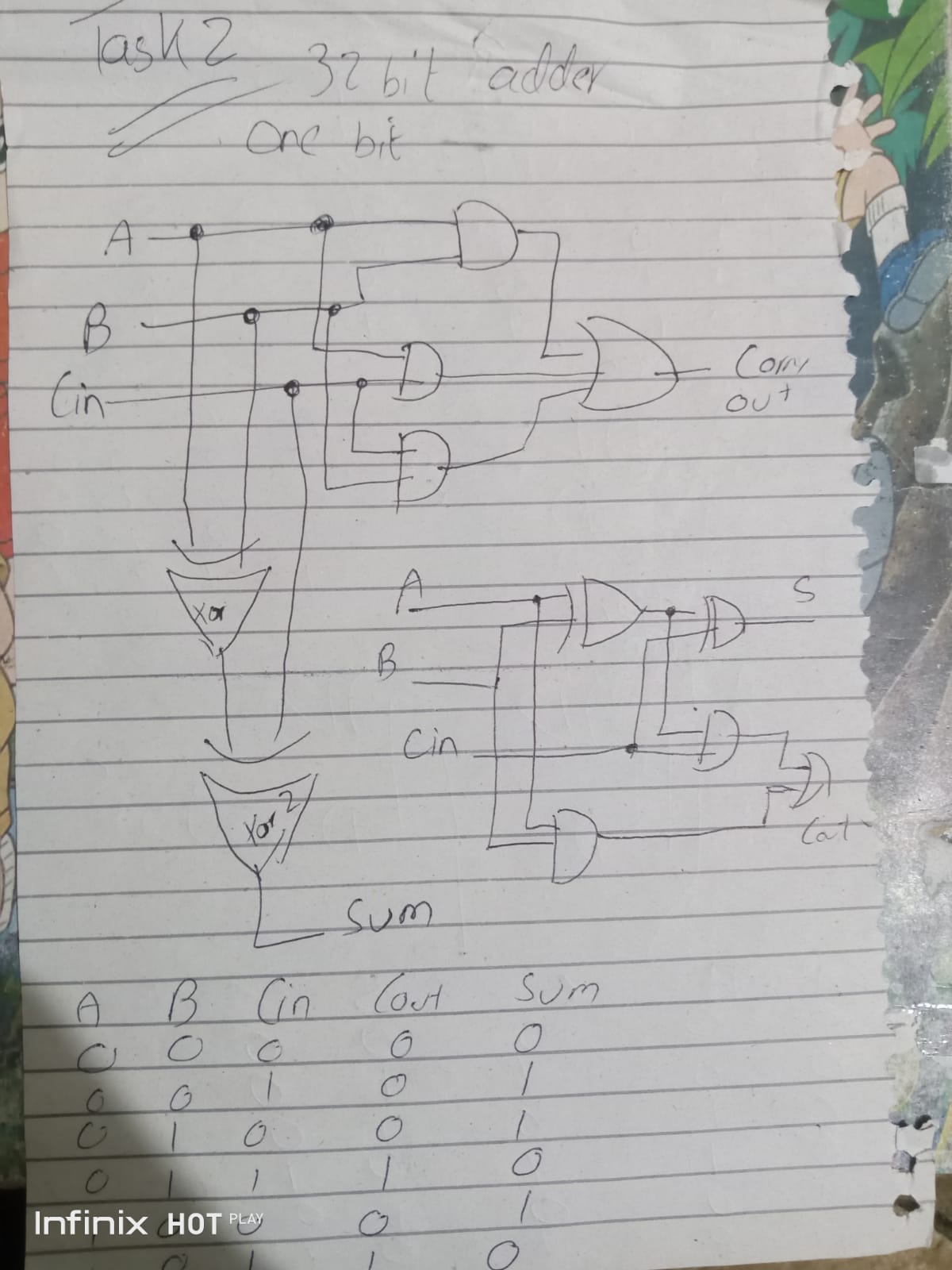


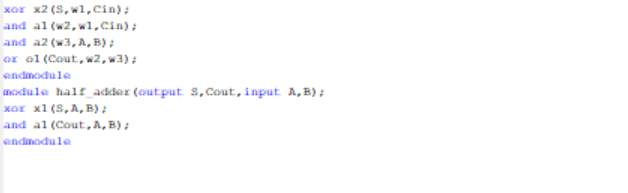
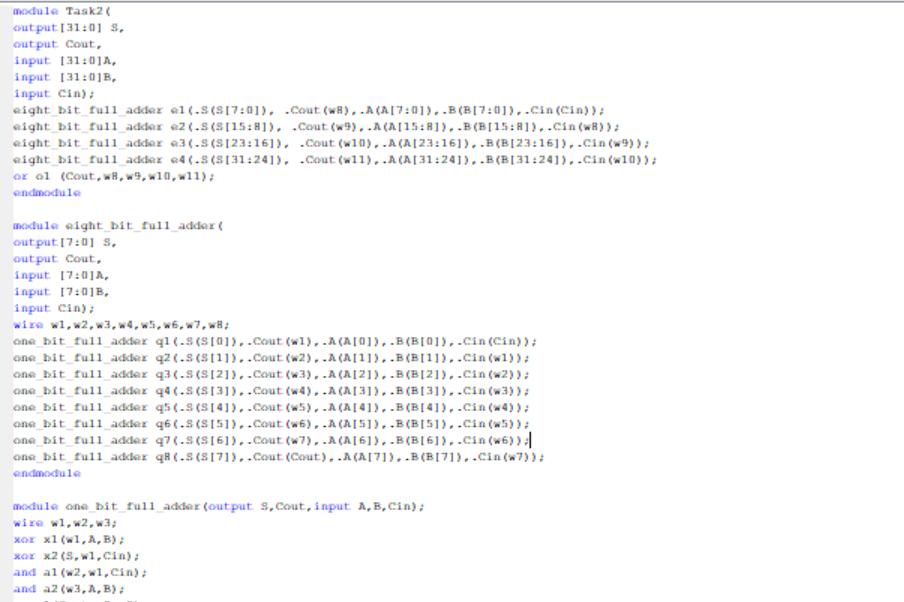
**Question:**

**Task 2:**

Design and stimulate 32-bit adder using a one-bit full adder. Show complete working of your solution. (Hint: Design an 8-bit adder using one-bit full adder, then instantiate these 8-bit adders to design a 32-bit adder).

**Code:**

****



module Task2(

output[31:0] S,

output Cout,

input [31:0]A,

input [31:0]B,

input Cin);

eight\_bit\_full\_adder e1(.S(S[7:0]), .Cout(w8),.A(A[7:0]),.B(B[7:0]),.Cin(Cin));

eight\_bit\_full\_adder e2(.S(S[15:8]), .Cout(w9),.A(A[15:8]),.B(B[15:8]),.Cin(w8));

eight\_bit\_full\_adder e3(.S(S[23:16]), .Cout(w10),.A(A[23:16]),.B(B[23:16]),.Cin(w9));

eight\_bit\_full\_adder e4(.S(S[31:24]), .Cout(w11),.A(A[31:24]),.B(B[31:24]),.Cin(w10));

or o1 (Cout,w8,w9,w10,w11);

endmodule

module eight\_bit\_full\_adder(

output[7:0] S,

output Cout,

input [7:0]A,

input [7:0]B,

input Cin);

wire w1,w2,w3,w4,w5,w6,w7,w8;

one\_bit\_full\_adder q1(.S(S[0]),.Cout(w1),.A(A[0]),.B(B[0]),.Cin(Cin));

one\_bit\_full\_adder q2(.S(S[1]),.Cout(w2),.A(A[1]),.B(B[1]),.Cin(w1));

one\_bit\_full\_adder q3(.S(S[2]),.Cout(w3),.A(A[2]),.B(B[2]),.Cin(w2));

one\_bit\_full\_adder q4(.S(S[3]),.Cout(w4),.A(A[3]),.B(B[3]),.Cin(w3));

one\_bit\_full\_adder q5(.S(S[4]),.Cout(w5),.A(A[4]),.B(B[4]),.Cin(w4));

one\_bit\_full\_adder q6(.S(S[5]),.Cout(w6),.A(A[5]),.B(B[5]),.Cin(w5));

one\_bit\_full\_adder q7(.S(S[6]),.Cout(w7),.A(A[6]),.B(B[6]),.Cin(w6));

one\_bit\_full\_adder q8(.S(S[7]),.Cout(Cout),.A(A[7]),.B(B[7]),.Cin(w7));

endmodule

module one\_bit\_full\_adder(output S,Cout,input A,B,Cin);

wire w1,w2,w3;

xor x1(w1,A,B);

xor x2(S,w1,Cin);

and a1(w2,w1,Cin);

and a2(w3,A,B);

or o1(Cout,w2,w3);

endmodule

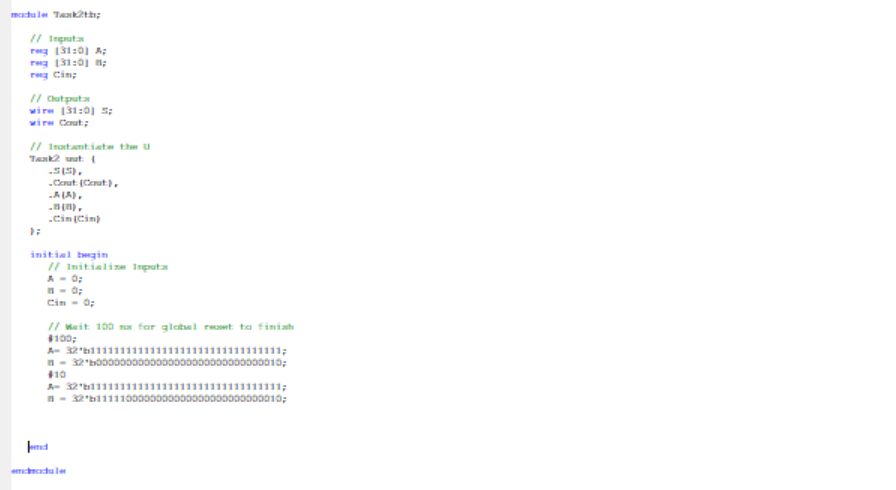
module half\_adder(output S,Cout,input A,B);

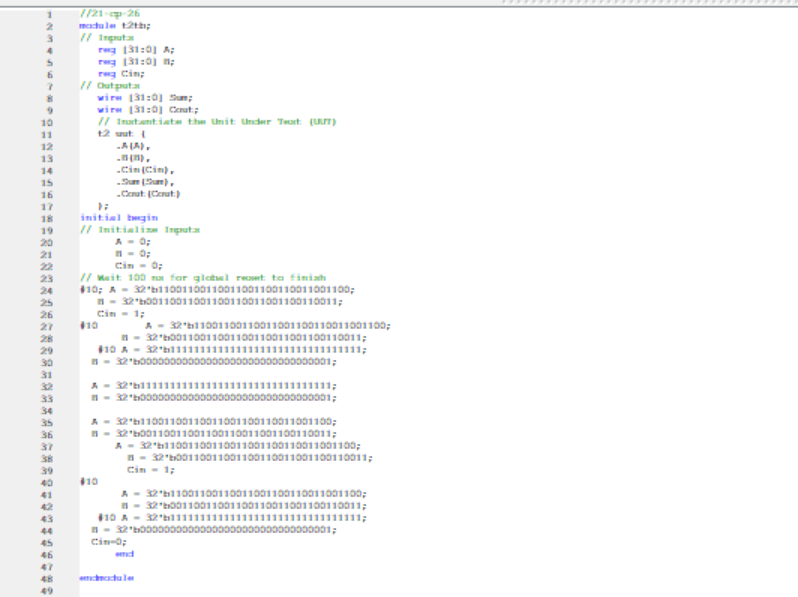
xor x1(S,A,B);

and a1(Cout,A,B);

endmodule

**TestBench:**





//

//21-cp-26

module t2tb;

// Inputs

reg [31:0] A;

reg [31:0] B;

reg Cin;

// Outputs

wire [31:0] Sum;

wire [31:0] Cout;

// Instantiate the Unit Under Test (UUT)

t2 uut (

.A(A),

.B(B),

.Cin(Cin),

.Sum(Sum),

.Cout(Cout)

);

initial begin

// Initialize Inputs

A = 0;

B = 0;

Cin = 0;

// Wait 100 ns for global reset to finish

#10; A = 32'b11001100110011001100110011001100;

B = 32'b00110011001100110011001100110011;

Cin = 1;

#10 A = 32'b11001100110011001100110011001100;

B = 32'b00110011001100110011001100110011;

#10 A = 32'b11111111111111111111111111111111;

B = 32'b00000000000000000000000000000001;

A = 32'b11111111111111111111111111111111;

B = 32'b00000000000000000000000000000001;

A = 32'b11001100110011001100110011001100;

B = 32'b00110011001100110011001100110011;

A = 32'b11001100110011001100110011001100;

B = 32'b00110011001100110011001100110011;

Cin = 1;

#10

A = 32'b11001100110011001100110011001100;

B = 32'b00110011001100110011001100110011;

#10 A = 32'b11111111111111111111111111111111;

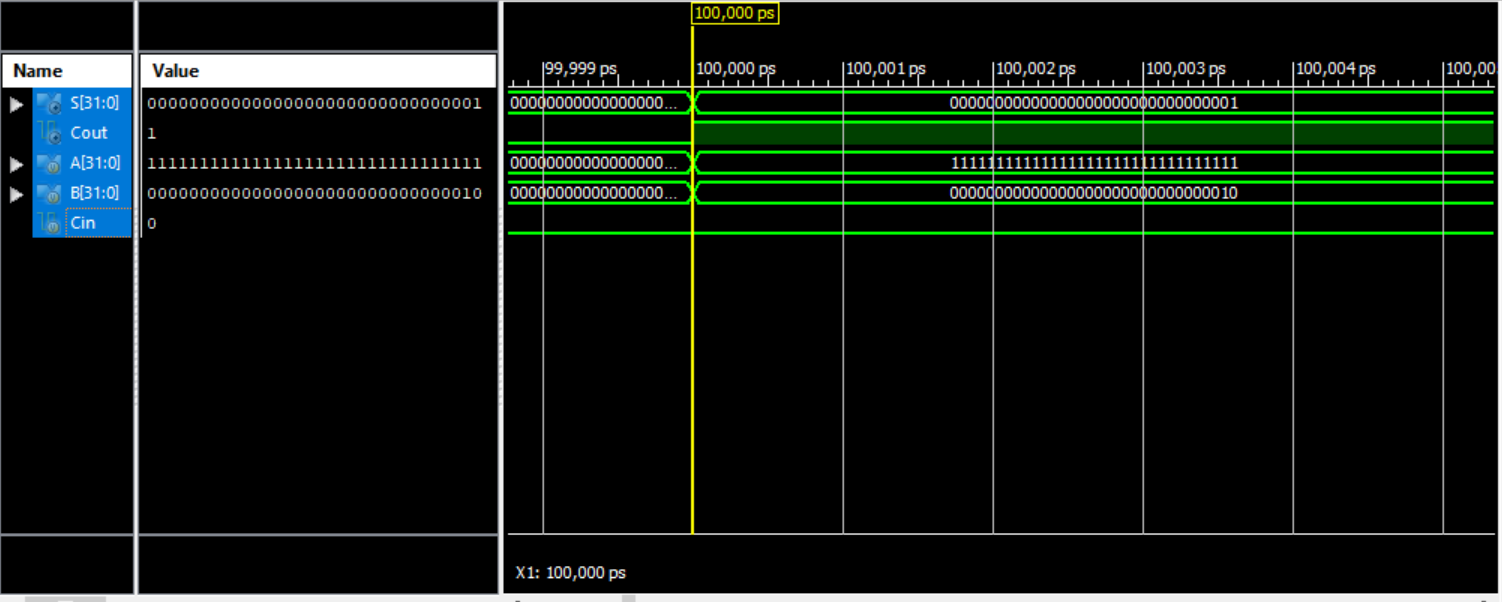
B = 32'b00000000000000000000000000000001;

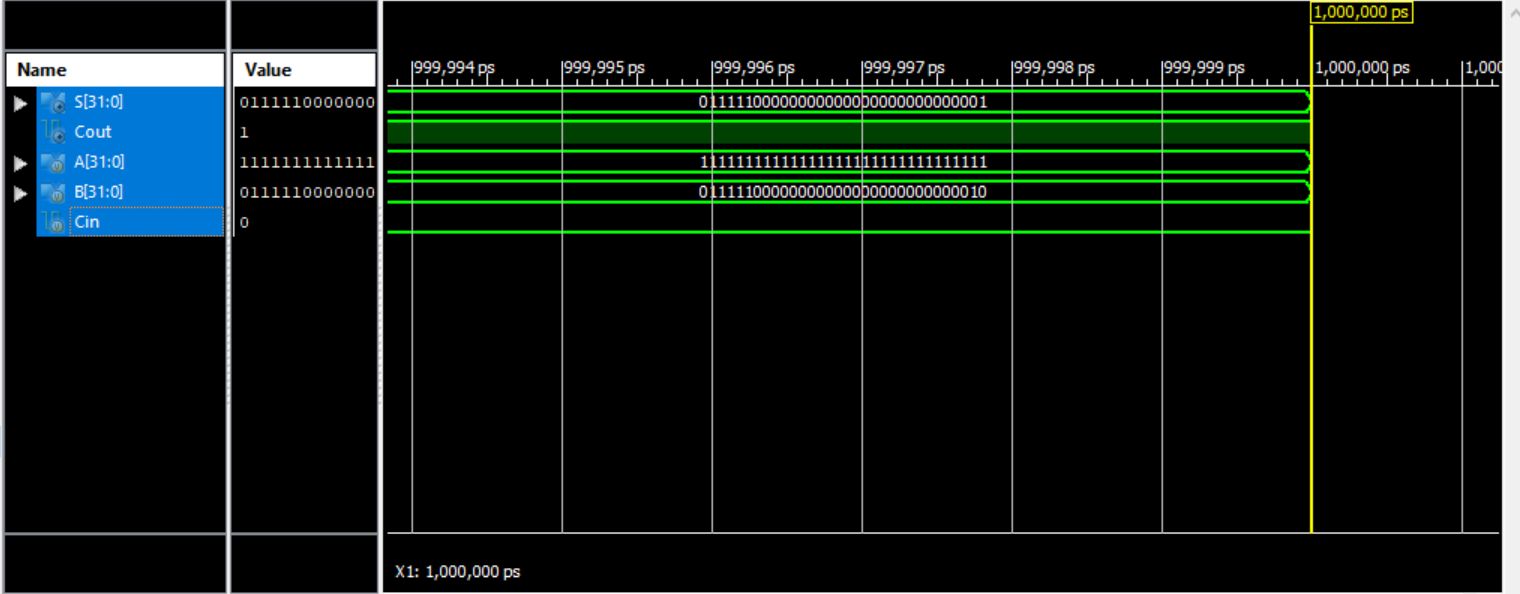
Cin=0;

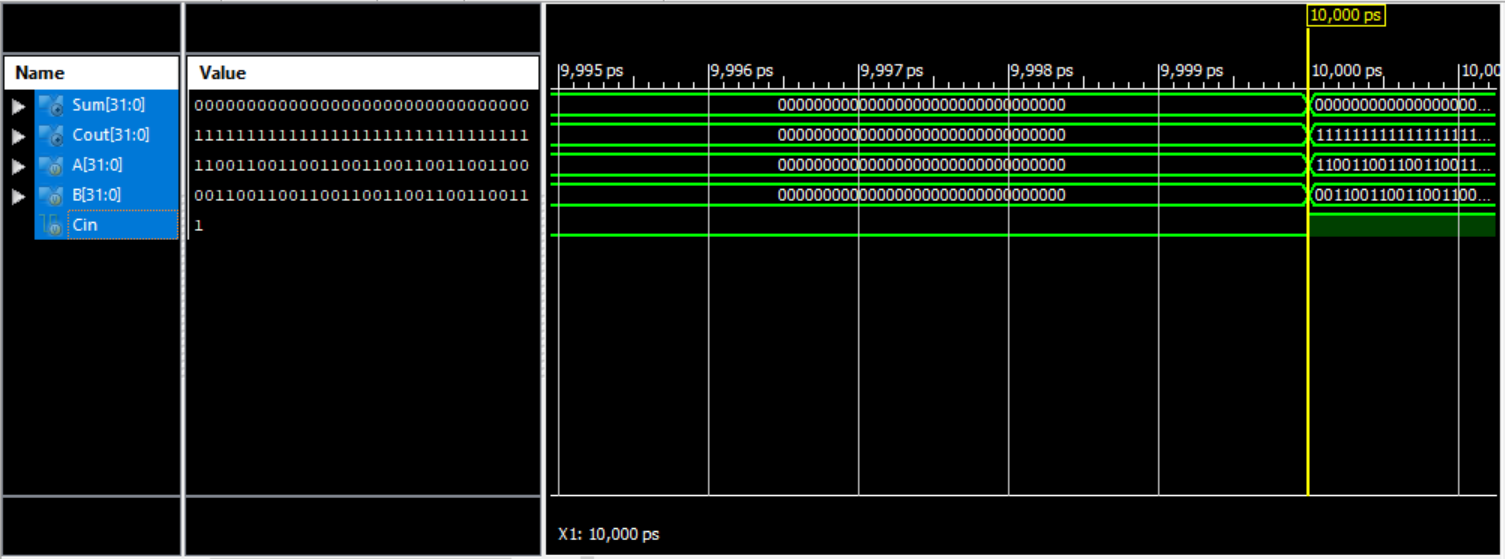
end

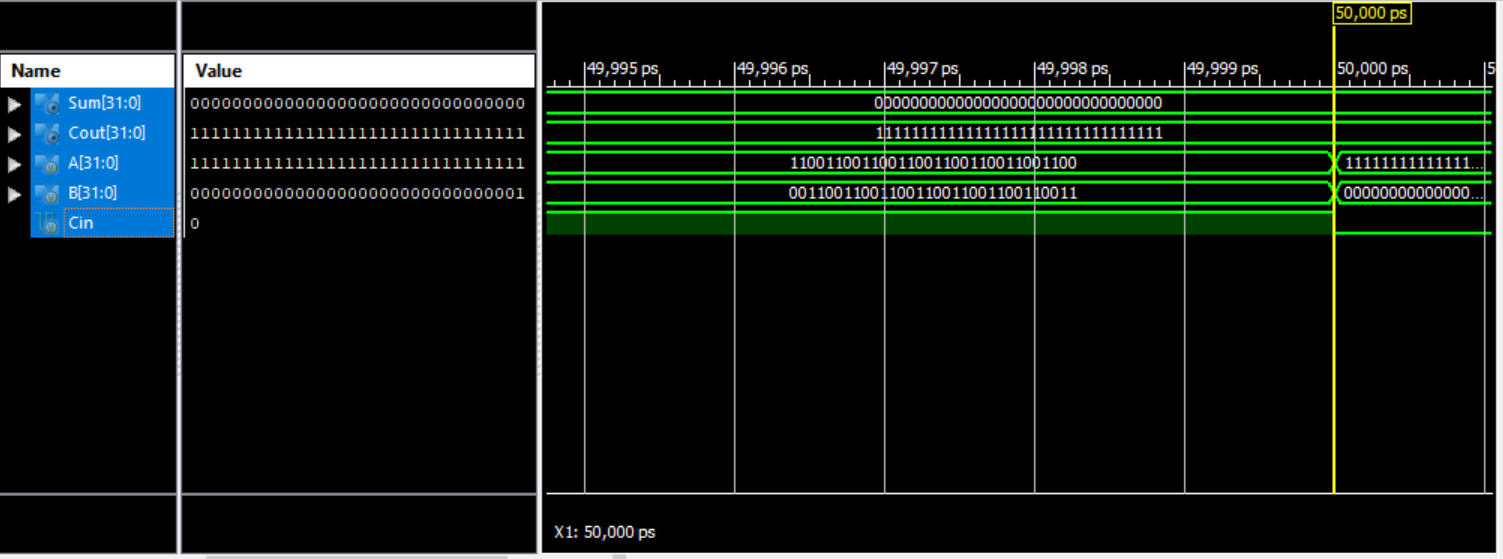
endmodule

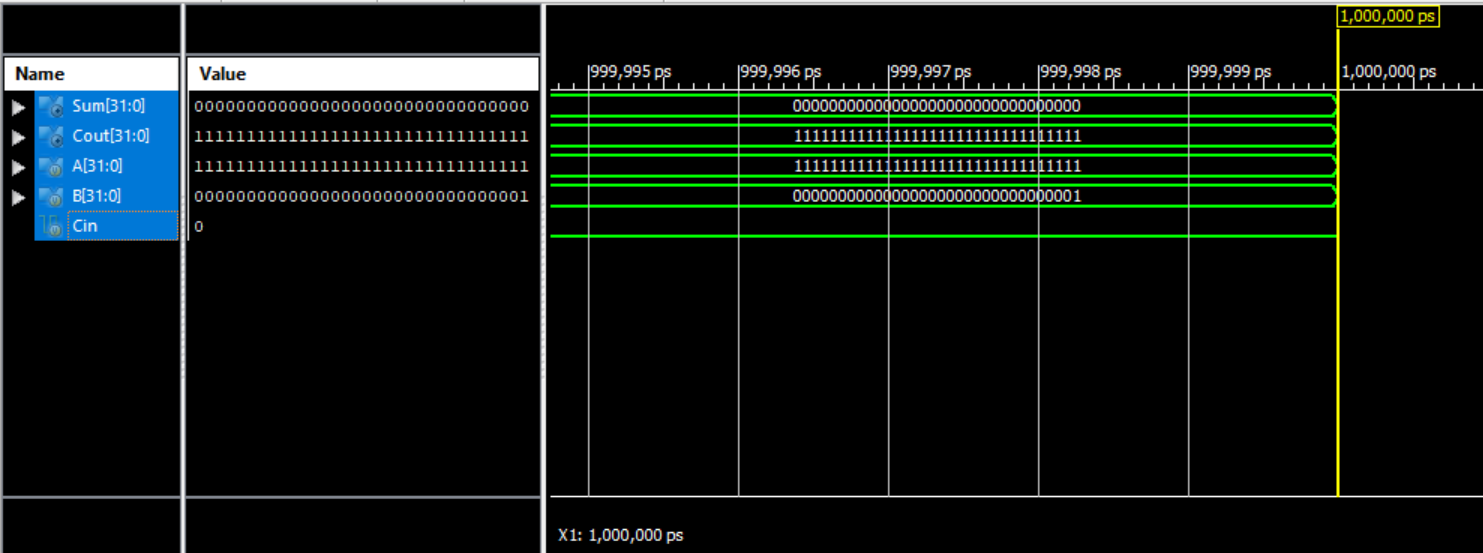
**Simulations :**









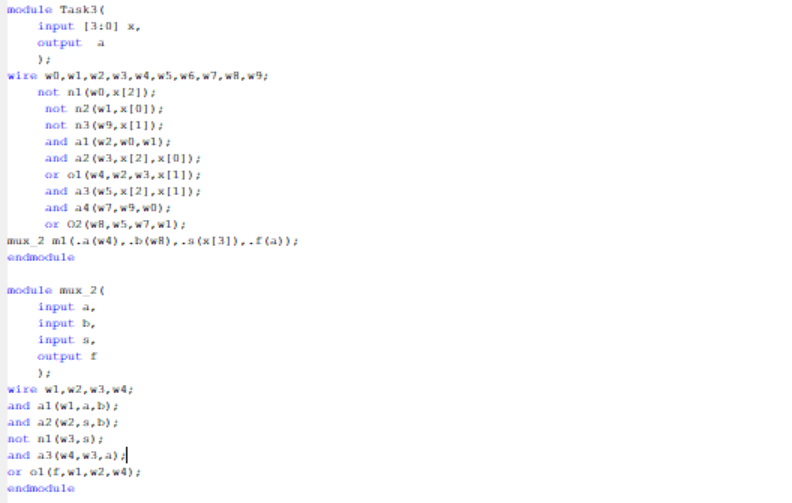


**Question:**

**Task 3:**

Design and simulate a BCD to Seven Segment Converter Using 2-to-1 Multiplexers Only. You cannot use any other component or primitive gateShow complete working of your solution.

**Code:**



module Task3(

input [3:0] x,

output a

);

wire w0,w1,w2,w3,w4,w5,w6,w7,w8,w9;

not n1(w0,x[2]);

not n2(w1,x[0]);

not n3(w9,x[1]);

and a1(w2,w0,w1);

and a2(w3,x[2],x[0]);

or o1(w4,w2,w3,x[1]);

and a3(w5,x[2],x[1]);

and a4(w7,w9,w0);

or O2(w8,w5,w7,w1);

mux\_2 m1(.a(w4),.b(w8),.s(x[3]),.f(a));

endmodule

module mux\_2(

input a,

input b,

input s,

output f

);

wire w1,w2,w3,w4;

and a1(w1,a,b);

and a2(w2,s,b);

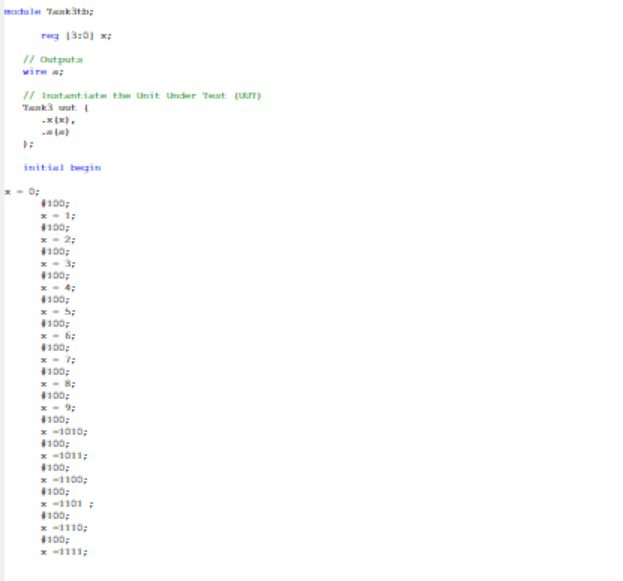
not n1(w3,s);

and a3(w4,w3,a);

or o1(f,w1,w2,w4);

endmodule

**TestBench:**



module Task3tb;

reg [3:0] x;

// Outputs

wire a;

// Instantiate the Unit Under Test (UUT)

Task3 uut (

.x(x),

.a(a)

);

initial begin

x = 0;

#100;

x = 1;

#100;

x = 2;

#100;

x = 3;

#100;

x = 4;

#100;

x = 5;

#100;

x = 6;

#100;

x = 7;

#100;

x = 8;

#100;

x = 9;

#100;

x =1010;

#100;

x =1011;

#100;

x =1100;

#100;

x =1101 ;

#100;

x =1110;

#100;

x =1111;

end

endmodule

**Simulations :**

