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:

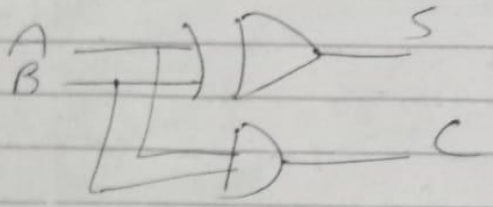
Design a full adder using two half adders using hierarchical modeling and other primitive logic gate (if required). Simulate and verify the following truth table in Xilinx ISE simulator.

Full Adder –Truth Table				
Input			Output	
A	B	Carry in	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

.....

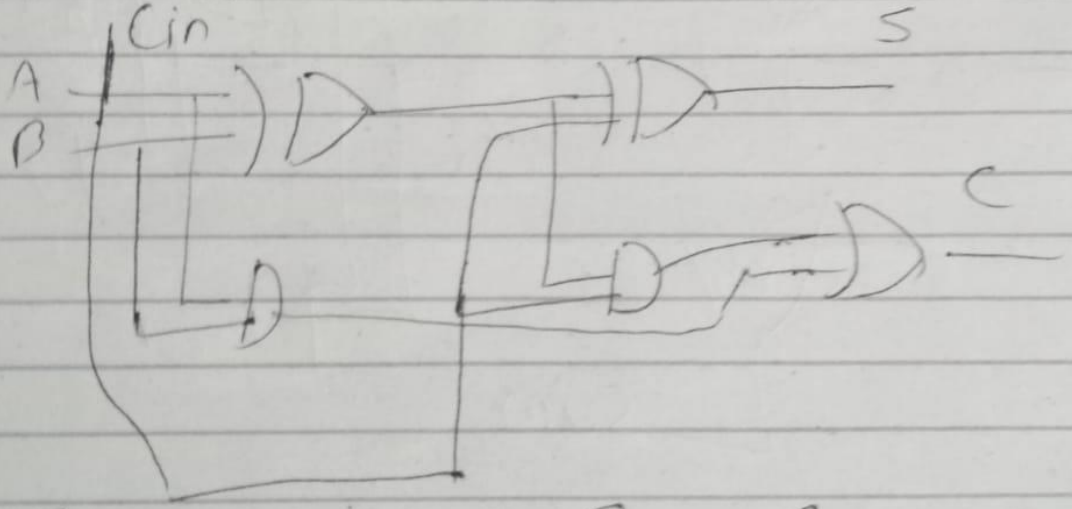
A	B	C	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Half adder



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full adder



A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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
```


//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:

```
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 ns for 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
```

//21-CP-26

module Task1tb;

    // Inputs

    reg A;

    reg B;

    reg Cin;

    // Outputs

    wire S;

    wire Cout;

    // Instantiate the Unit Under Test (UUT)

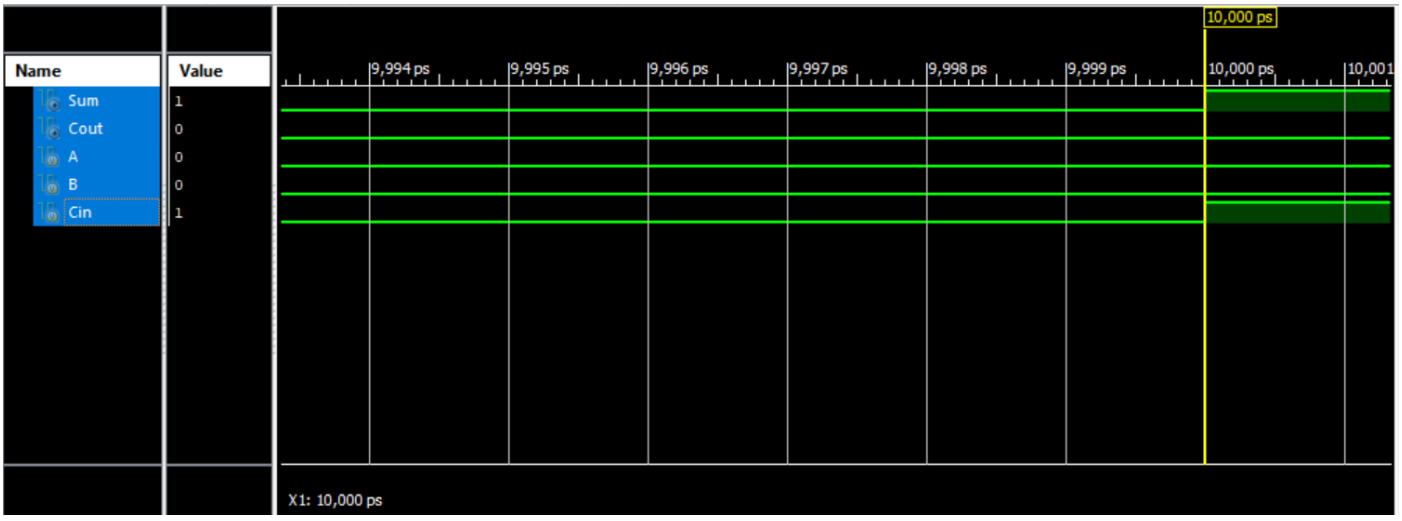
    Task1 uut (

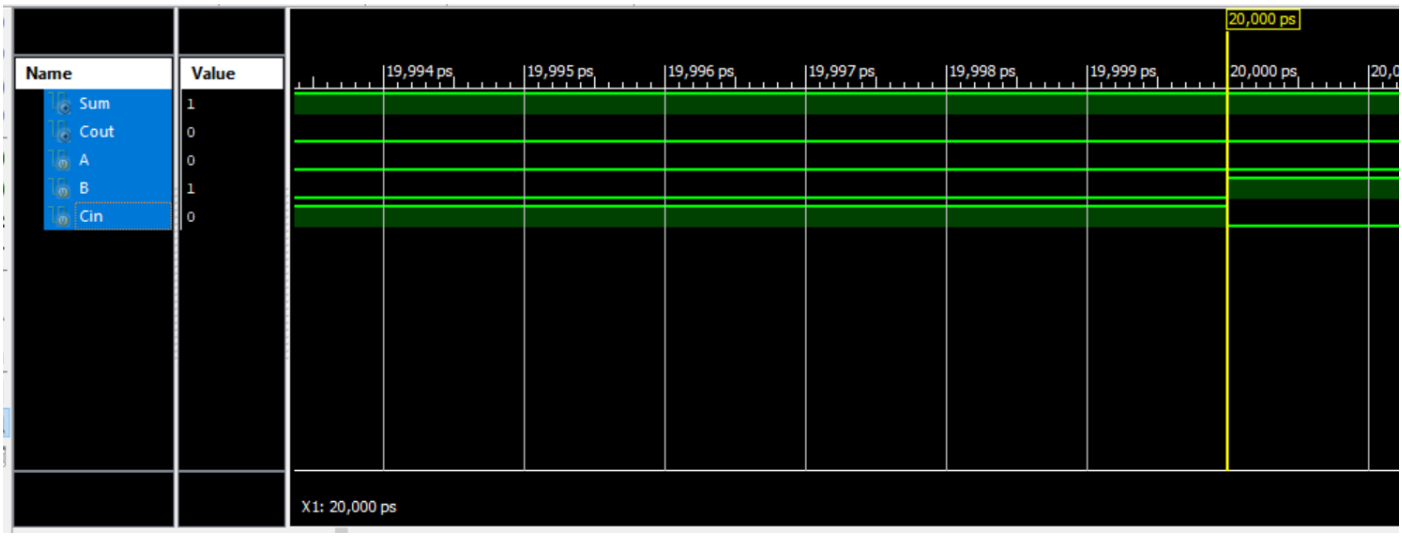
```
        .S(S),
        .Cout(Cout),
        .A(A),
        .B(B),
        .Cin(Cin)
    );

    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
```

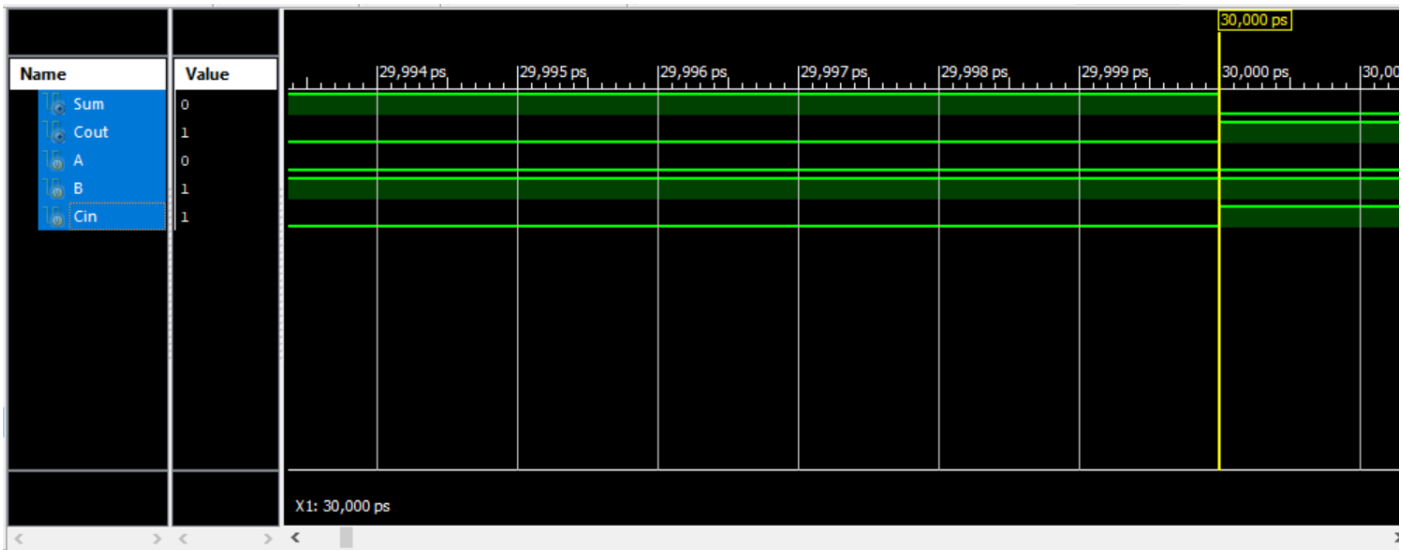
Simulations :

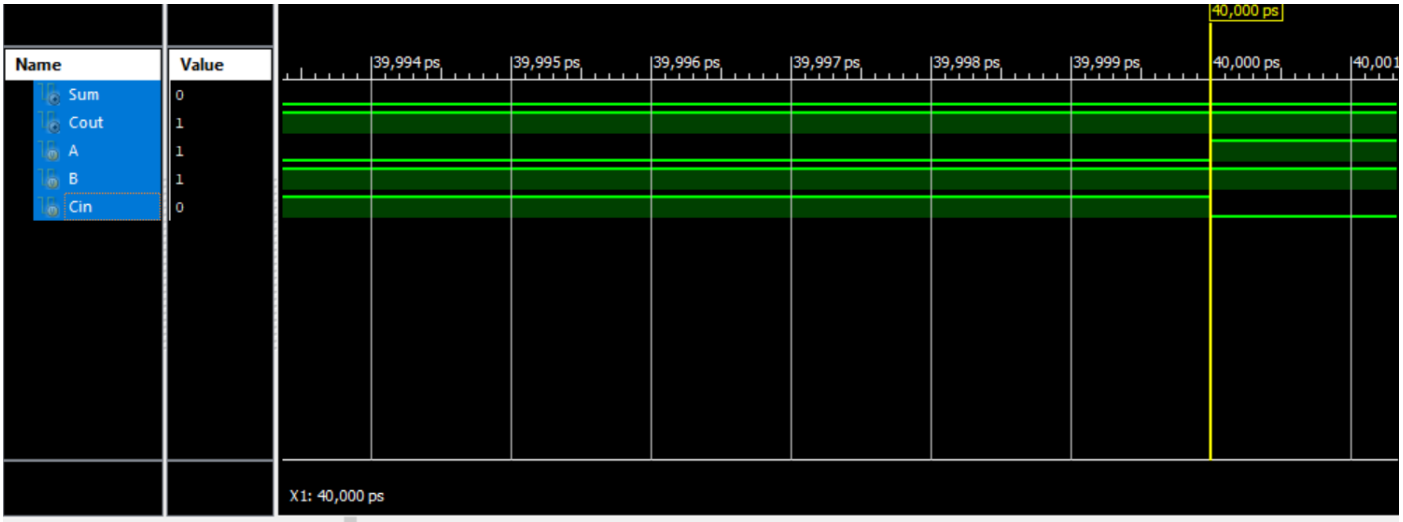
As you can see the truth table verifies

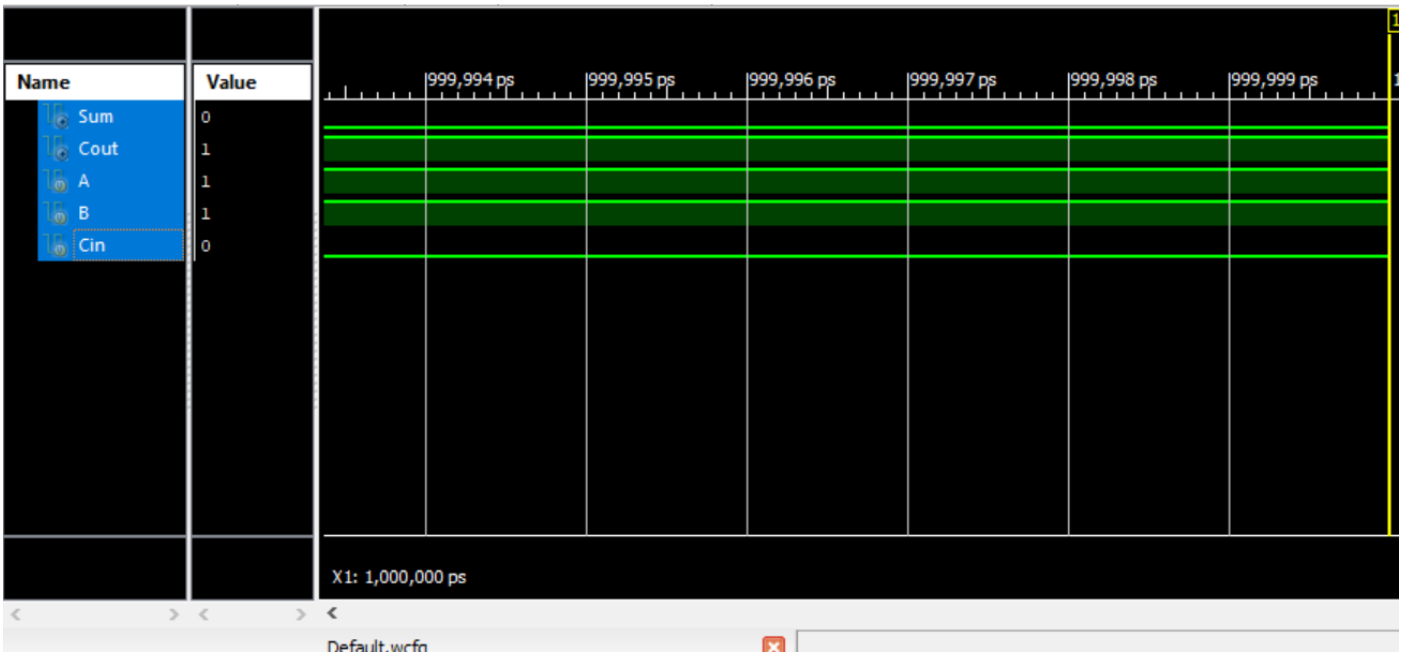
Full Adder –Truth Table				
Input			Output	
A	B	Carry in	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

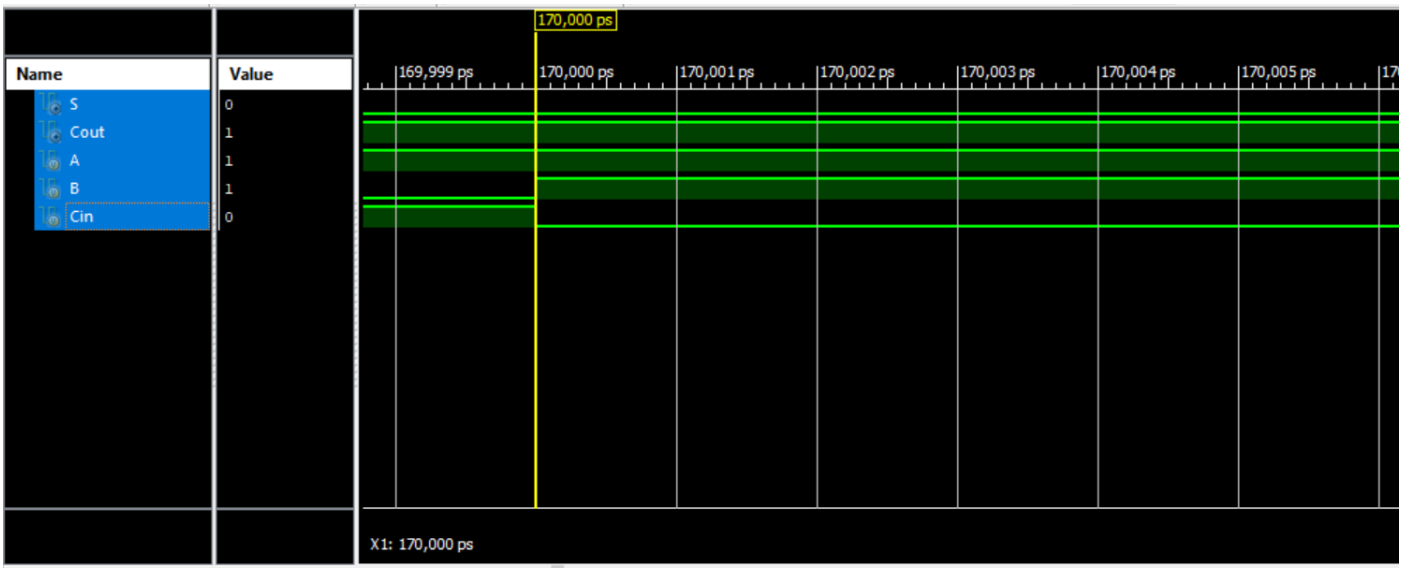





Name
Value
Sum
Cout
A
B
Cin
0
1
0
1
1
29,994 ps
29,995 ps
29,996 ps
29,997 ps
29,998 ps
29,999 ps
30,000 ps
X1: 30,000 ps


Name
Value
Sum
Cout
A
B
Cin
0
1
1
1
0
39,994 ps
39,995 ps
39,996 ps
39,997 ps
39,998 ps
39,999 ps
40,000 ps
X1: 40,000 ps


Name
Value
Sum
Cout
A
B
Cin
0
1
1
1
0
999,994 ps
999,995 ps
999,996 ps
999,997 ps
999,998 ps
999,999 ps
X1: 1,000,000 ps
Default.wcfg



Question:

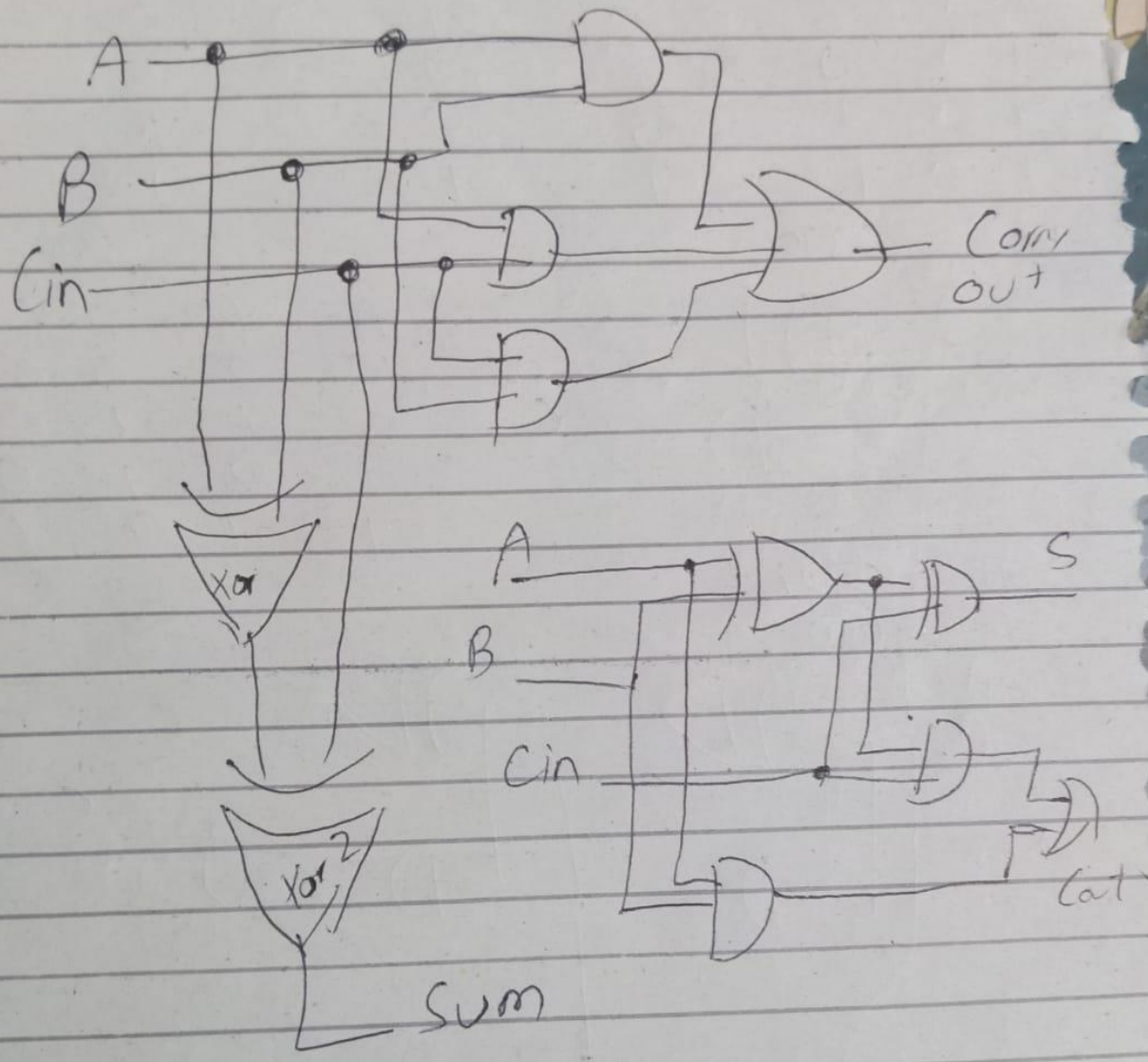
Task 2:

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:

Task 2 32 bit adder

One bit



A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

```
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
```


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:

```
module Task2tb;

    // Inputs
    reg [31:0] A;
    reg [31:0] B;
    reg Cin;

    // Outputs
    wire [31:0] S;
    wire Cout;

    // Instantiate the U
    Task2 uut (
        .S(S),
        .Cout(Cout),
        .A(A),
        .B(B),
        .Cin(Cin)
    );

    initial begin
        // Initialize Inputs
        A = 0;
        B = 0;
        Cin = 0;

        // Wait 100 ns for global reset to finish
        #100;
        A= 32'b11111111111111111111111111111111;
        B = 32'b00000000000000000000000000000010;
        #10
        A= 32'b11111111111111111111111111111111;
        B = 32'b11111000000000000000000000000010;


    end

endmodule
```

```
1   //21-cp-26
2   module t2tb;
3   // Inputs
4       reg [31:0] A;
5       reg [31:0] B;
6       reg Cin;
7   // Outputs
8       wire [31:0] Sum;
9       wire [31:0] Cout;
10      // Instantiate the Unit Under Test (UUT)
11      t2 uut (
12          .A(A),
13          .B(B),
14          .Cin(Cin),
15          .Sum(Sum),
16          .Cout(Cout)
17      );
18  initial begin
19  // Initialize Inputs
20          A = 0;
21          B = 0;
22          Cin = 0;
23  // Wait 100 ns for global reset to finish
24  #10; A = 32'b11001100110011001100110011001100;
25     B = 32'b00110011001100110011001100110011;
26     Cin = 1;
27  #10         A = 32'b11001100110011001100110011001100;
28          B = 32'b00110011001100110011001100110011;
29     #10 A = 32'b11111111111111111111111111111111;
30    B = 32'b00000000000000000000000000000001;
31
32    A = 32'b11111111111111111111111111111111;
33    B = 32'b00000000000000000000000000000001;
34
35    A = 32'b11001100110011001100110011001100;
36    B = 32'b00110011001100110011001100110011;
37        A = 32'b11001100110011001100110011001100;
38          B = 32'b00110011001100110011001100110011;
39          Cin = 1;
40  #10
41          A = 32'b11001100110011001100110011001100;
42          B = 32'b00110011001100110011001100110011;
43     #10 A = 32'b11111111111111111111111111111111;
44    B = 32'b00000000000000000000000000000001;
45    Cin=0;
46        end
47
48  endmodule
49
```

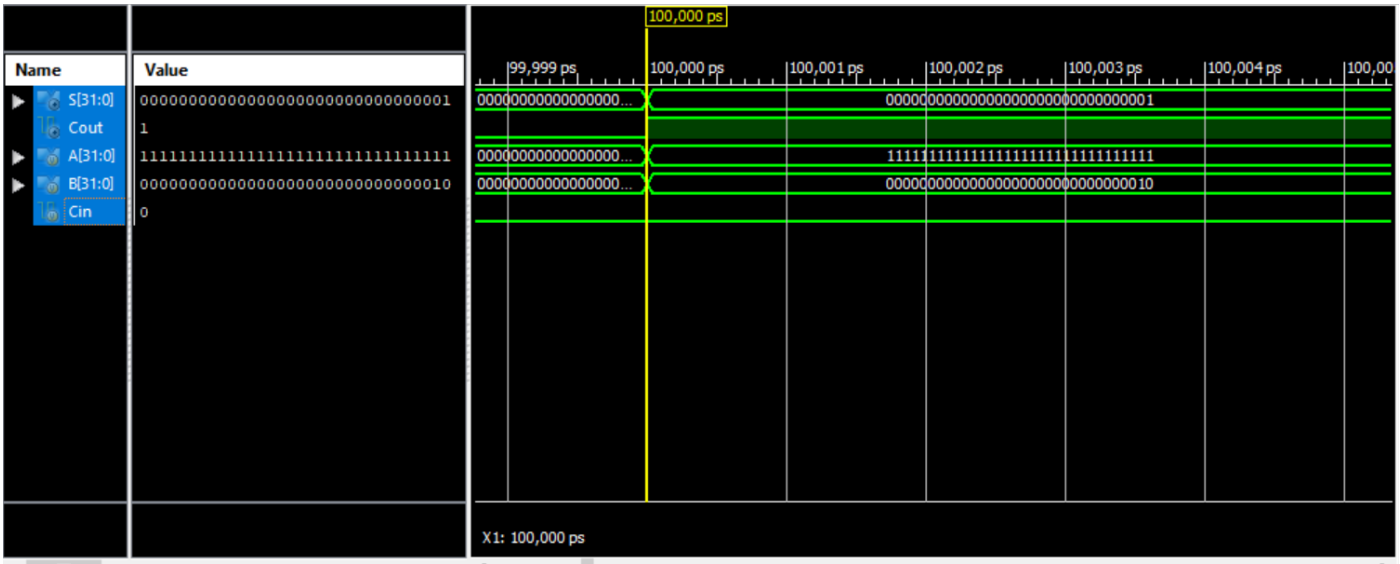
//

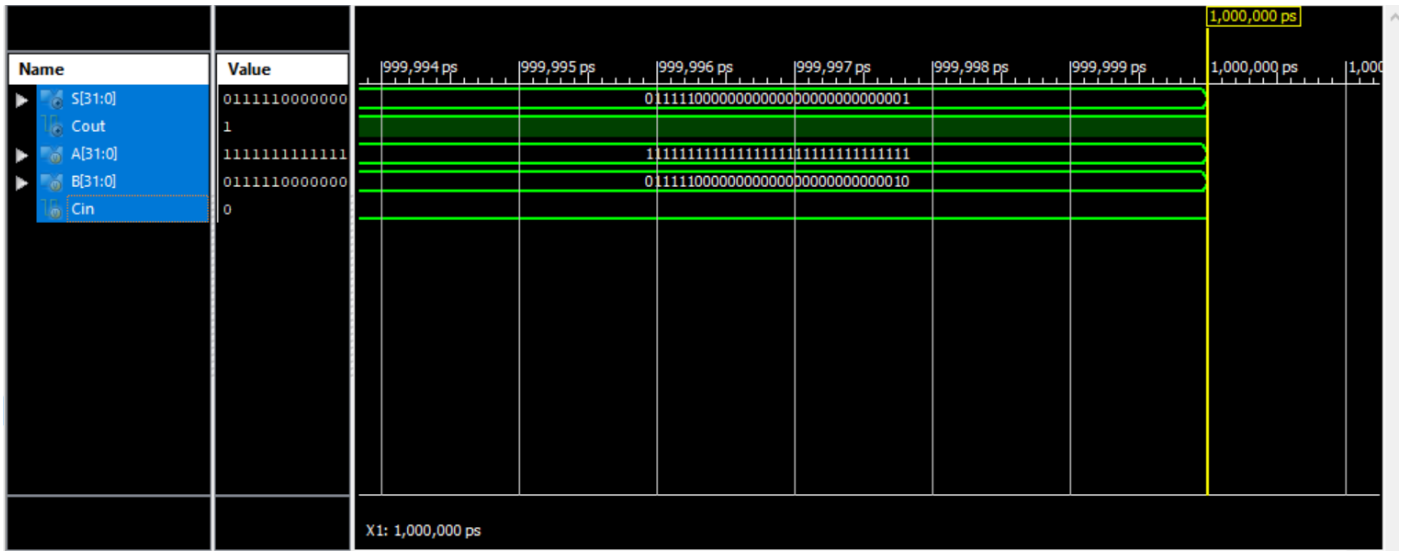
//21-cp-26

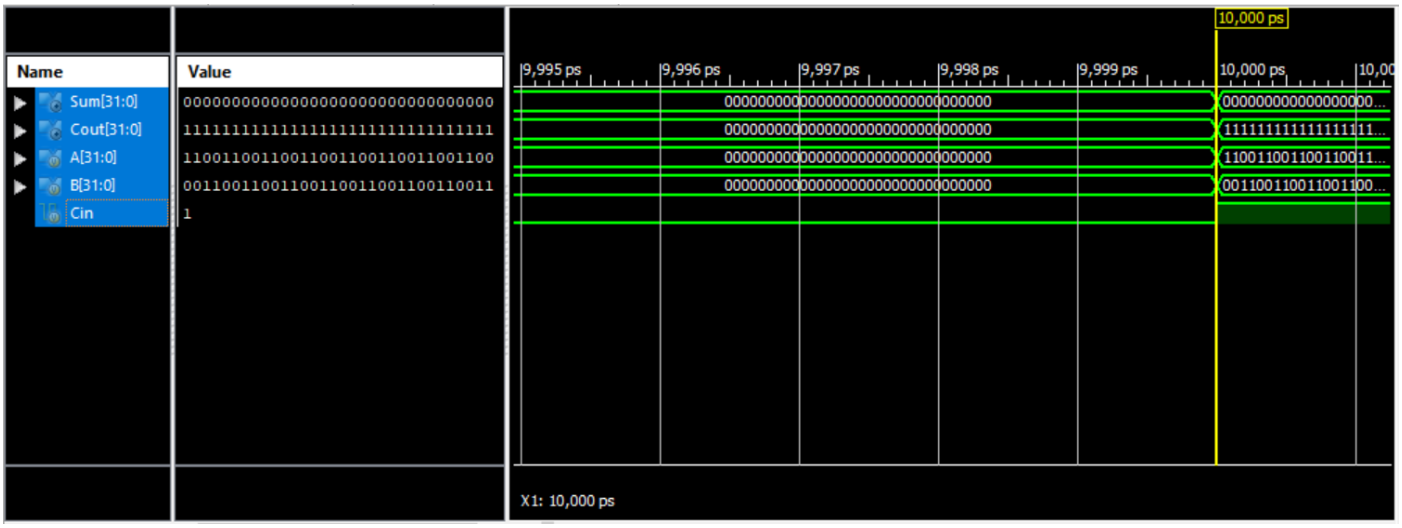
```
module t2tb;
// Inputs
	reg [31:0] A;
	reg [31:0] B;
	reg Cin;
// Outputs
	wire [31:0] Sum;
	wire [31:0] Cout;
	// Instantiate the Unit Under Test (UUT)
	t2 uut (
		.A(A),
		.B(B),
		.Cin(Cin),
		.Sum(Sum),
		.Cout(Cout)
	);
initial begin
// Initialize Inputs
		A = 0;
		B = 0;
		Cin = 0;
// Wait 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;
```

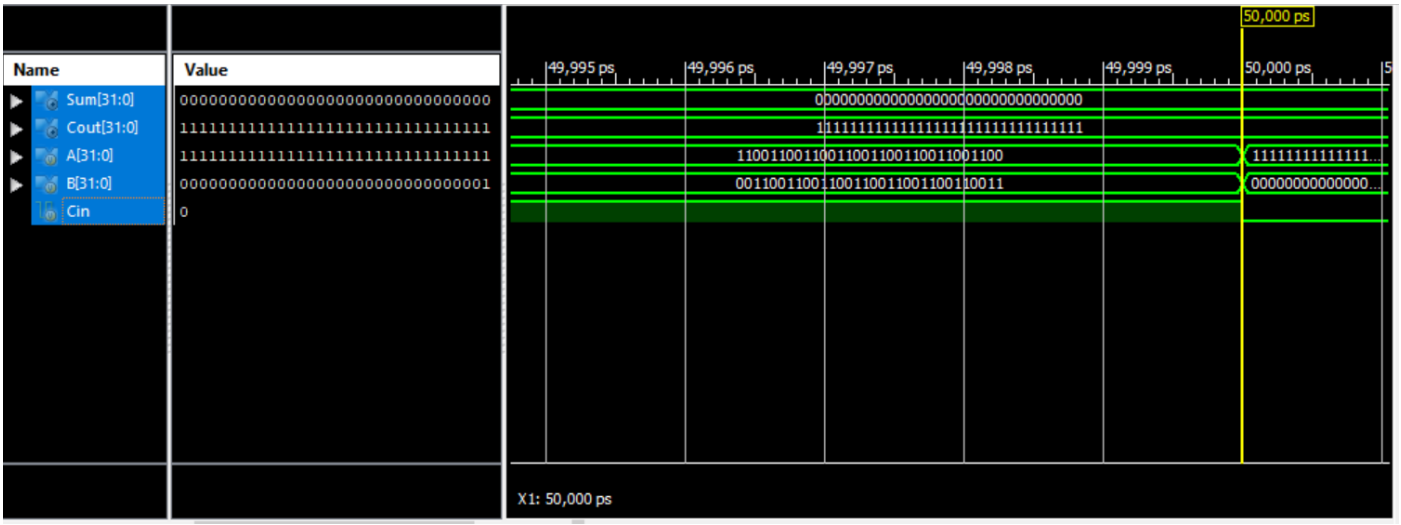
```
    A = 32'b11001100110011001100110011001100;
     B = 32'b00110011001100110011001100110011;
     Cin = 1;
#10
                A = 32'b11001100110011001100110011001100;
    B = 32'b00110011001100110011001100110011;
  #10 A = 32'b11111111111111111111111111111111;
  B = 32'b00000000000000000000000000000001;
  Cin=0;
                end

endmodule
```

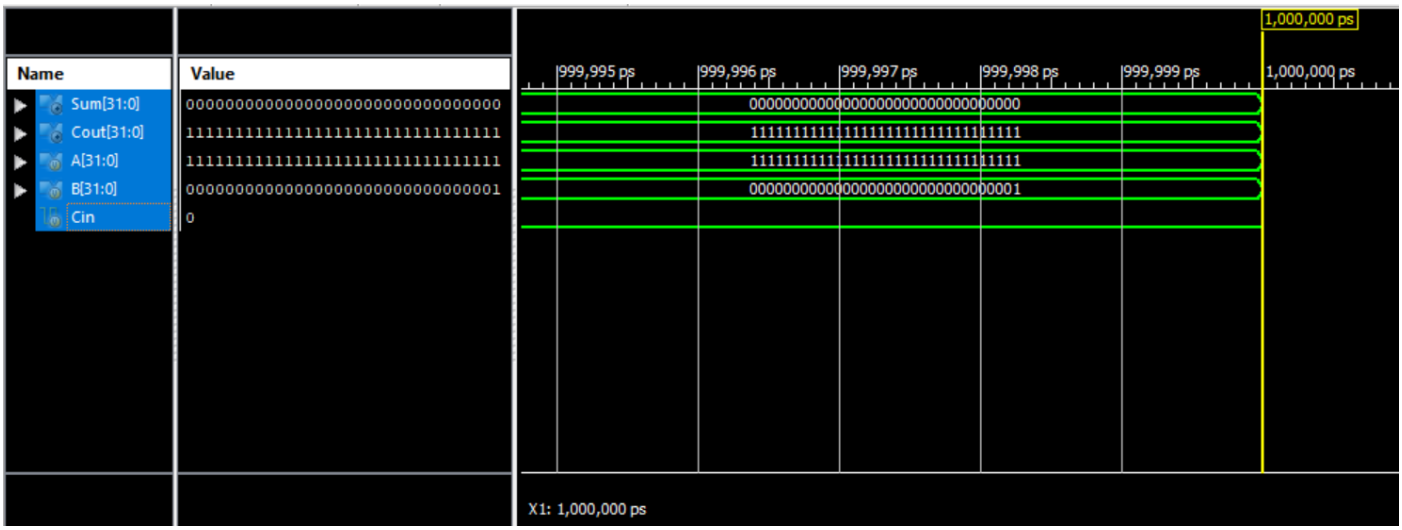
Simulations :






Name
Value
Sum[31:0]
Cout[31:0]
A[31:0]
B[31:0]
Cin
00000000000000000000000000000000
11111111111111111111111111111111
11001100110011001100110011001100
00110011001100110011001100110011
1
10,000 ps
9,995 ps
9,996 ps
9,997 ps
9,998 ps
9,999 ps
10,000 ps
X1: 10,000 ps


Name
Value
Sum[31:0]
Cout[31:0]
A[31:0]
B[31:0]
Cin
00000000000000000000000000000000
11111111111111111111111111111111
11111111111111111111111111111111
00000000000000000000000000000001
0
50,000 ps
49,995 ps
49,996 ps
49,997 ps
49,998 ps
49,999 ps
50,000 ps
X1: 50,000 ps


Name
Value
Sum[31:0]
Cout[31:0]
A[31:0]
B[31:0]
Cin
00000000000000000000000000000000
11111111111111111111111111111111
11111111111111111111111111111111
00000000000000000000000000000001
0
1,000,000 ps
999,995 ps
999,996 ps
999,997 ps
999,998 ps
999,999 ps
1,000,000 ps
X1: 1,000,000 ps

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
```

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;
```

module Task3tb;

reg [3:0] x;

// Outputs

wire a;

// Instantiate the Unit Under Test (UUT)

Task3 uut (

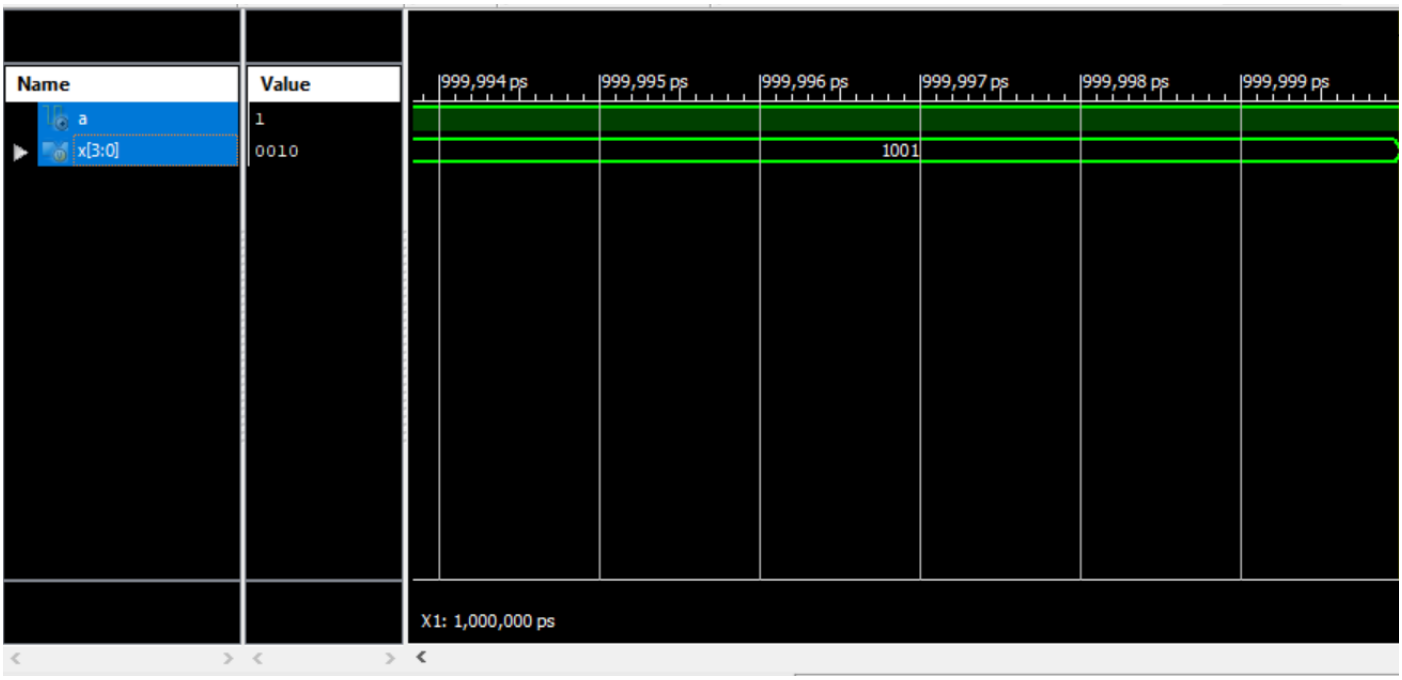
.x(x),

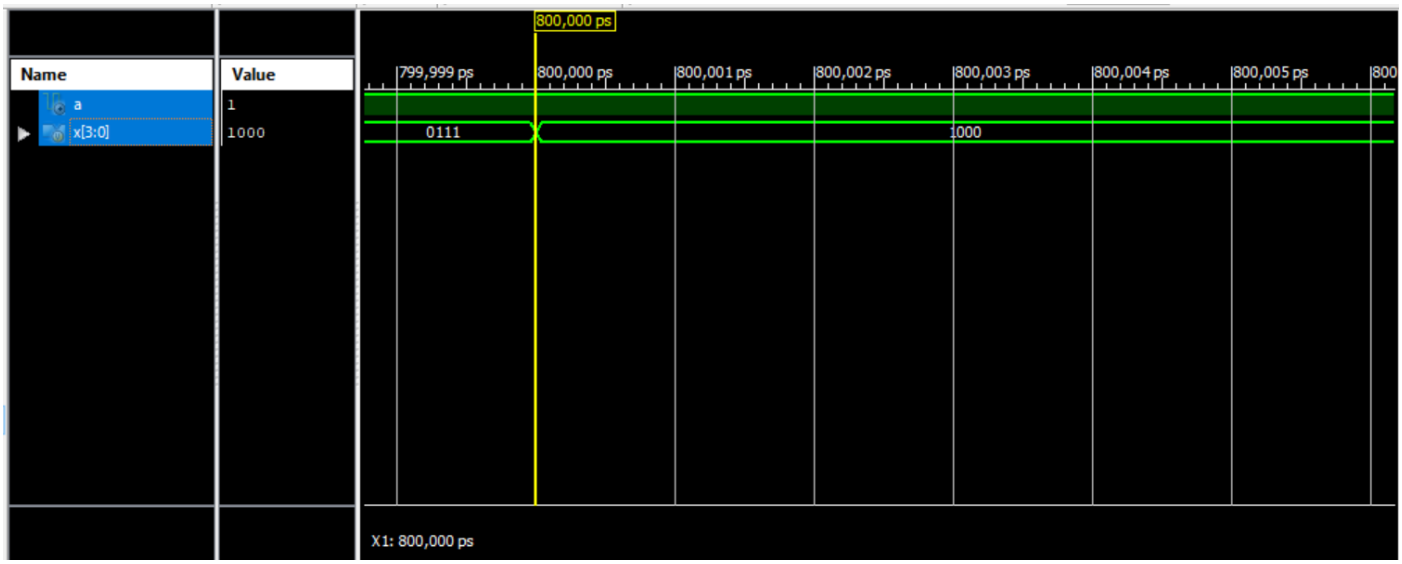
.a(a)

```
);

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 :


Name
Value
a
1
x[3:0]
0010
999,994 ps
999,995 ps
999,996 ps
999,997 ps
999,998 ps
999,999 ps
1001
X1: 1,000,000 ps


Name
Value
a
1
x[3:0]
1000
800,000 ps
799,999 ps
800,000 ps
800,001 ps
800,002 ps
800,003 ps
800,004 ps
800,005 ps
0111
1000
X1: 800,000 ps