University of Engineering and Technology,Taxila

Department of Computer Engineering

Lab Report 04

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

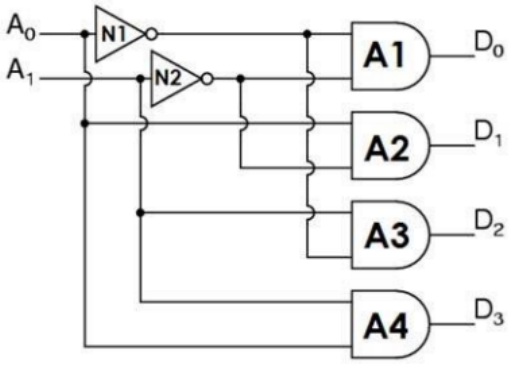
Date: 11-02-24.

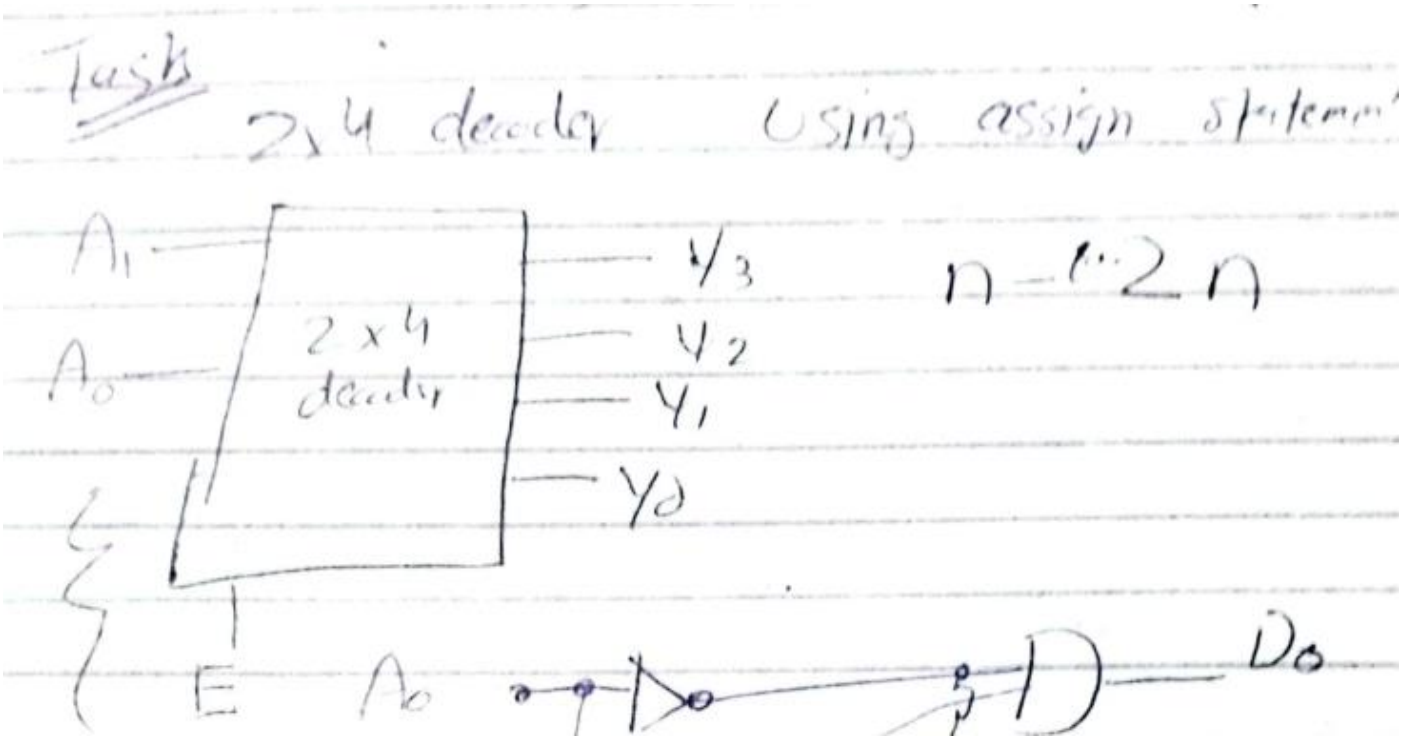
Course Title: DSD Lab

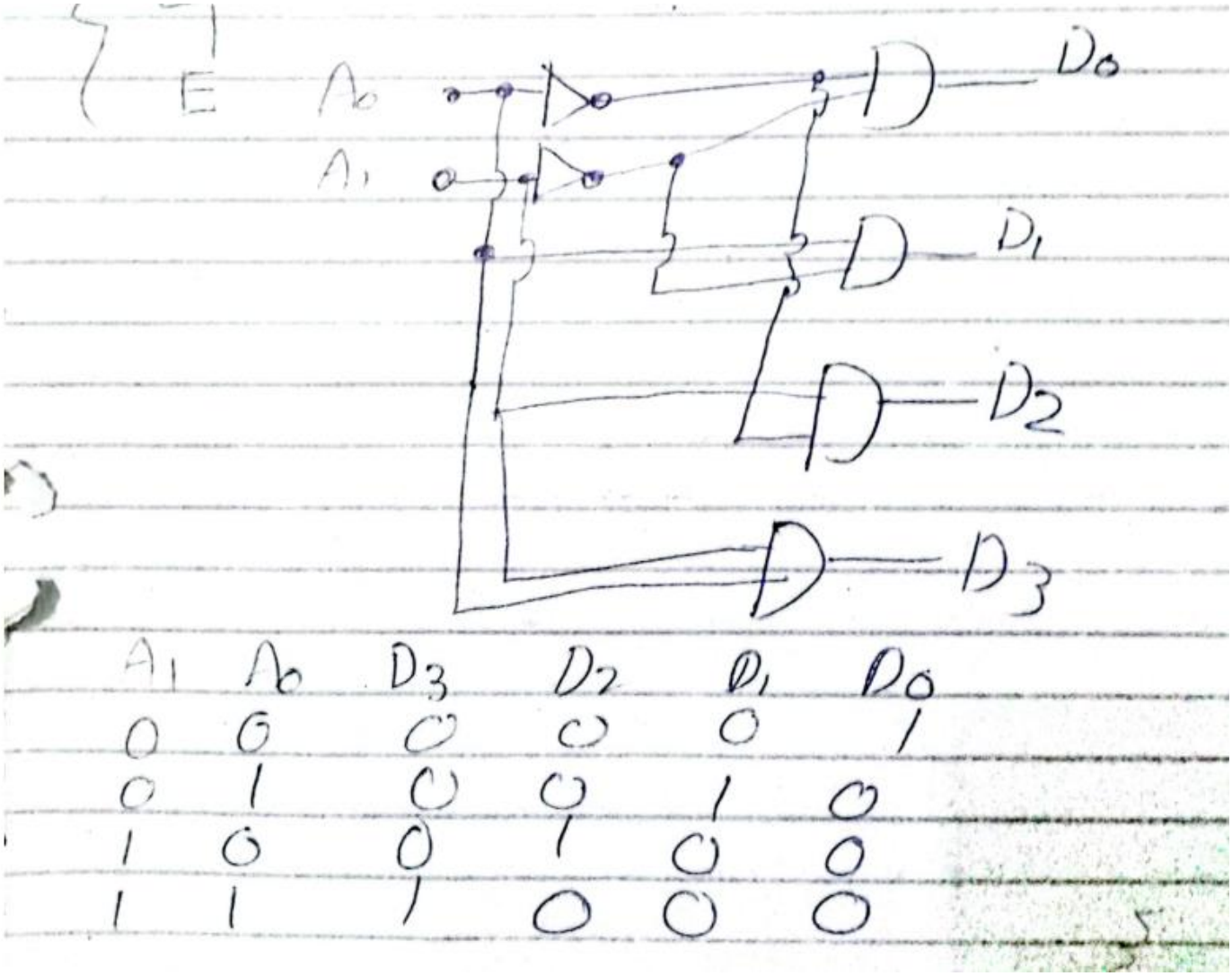
Question 1:

5 Lab Tasks: You can use only Data-Flow Modeling in this Lab.

1. **Simulate 2 × 4 decoder, given in the following diagram, using `assign` statement only.**
   a) Neglect the propagation delays, simulate the design, and analyse the output. Are all the outputs in all three parts the same or different?
   b) Consider propagation delay of inverter `N1` is `5 time-units` and propagation delay of the and gate `A1` is `6 time-units` and change input values in testbench after each `5 timeunits`. Analyze the output.
   c) Now change the propagation delay of `A1` to `5 time-units` as well and keep changing the input values in testbench after each `5 time-units`. Analyze the output again.





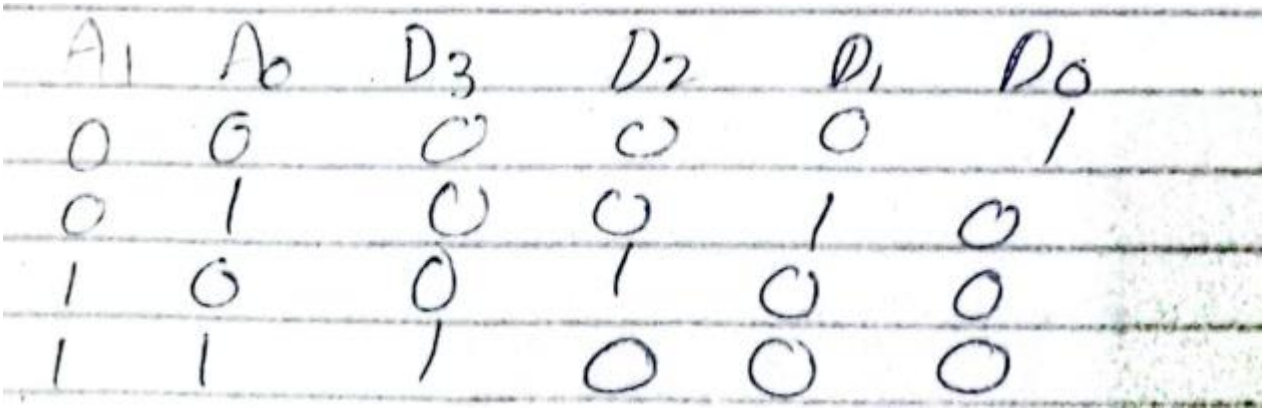


Code:

```
module Task1(output [3:0] G, input A0, A1);
  wire w1, w2, w3, w4;

  not_gate n1 (.A(w1), .B(A0));
  not_gate n2 (.A(w2), .B(A1));
  and_gate a1 (.Y(G[0]), .X(w1), .Z(w2));
  and_gate a2 (.Y(G[1]), .X(A0), .Z(w2));
  and_gate a3 (.Y(G[2]), .X(A1), .Z(w1));
  and_gate a4 (.Y(G[3]), .X(A0), .Z(A1));
endmodule

module and_gate(output Y, input X, Z);
  assign Y = X & Z;
endmodule

module not_gate(output A, input B);
  assign A = !B;
endmodule
```

```
module Task1(output [3:0] G, input A0, A1);
  wire w1, w2, w3, w4;

  not_gate n1 (.A(w1), .B(A0));
  not_gate n2 (.A(w2), .B(A1));
  and_gate a1 (.Y(G[0]), .X(w1), .Z(w2));
  and_gate a2 (.Y(G[1]), .X(A0), .Z(w2));
  and_gate a3 (.Y(G[2]), .X(A1), .Z(w1));
  and_gate a4 (.Y(G[3]), .X(A0), .Z(A1));
endmodule

module and_gate(output Y, input X, Z);
  assign Y = X & Z;
endmodule

module not_gate(output A, input B);
  assign A = !B;
endmodule
```

TestBench:



```
module Task1Tb;

    // Inputs
    reg A0;
    reg A1;

    // Outputs
    wire [3:0] G;

    // Instantiate the Unit Under Test (UUT)
    Task1 uut (
        .G(G),
        .A0(A0),
        .A1(A1)
    );

    initial begin
        // Initialize Inputs
        A0 = 0;
        A1 = 0;

        // Wait 100 ns for global reset to finish
        #100;
        A0 = 0;
        A1 = 0;
        #10
        A0 = 0;
        A1 = 1;
        #10
        A0 = 1;
        A1 = 0;
        #10
        A0 = 1;
        A1 = 1;

        // Add stimulus here

    end

endmodule
```

module Task1Tb;

// Inputs

reg A0;

reg A1;

// Outputs

wire [3:0] G;

// Instantiate the Unit Under Test (UUT)

Task1 uut (

.G(G),

.A0(A0),

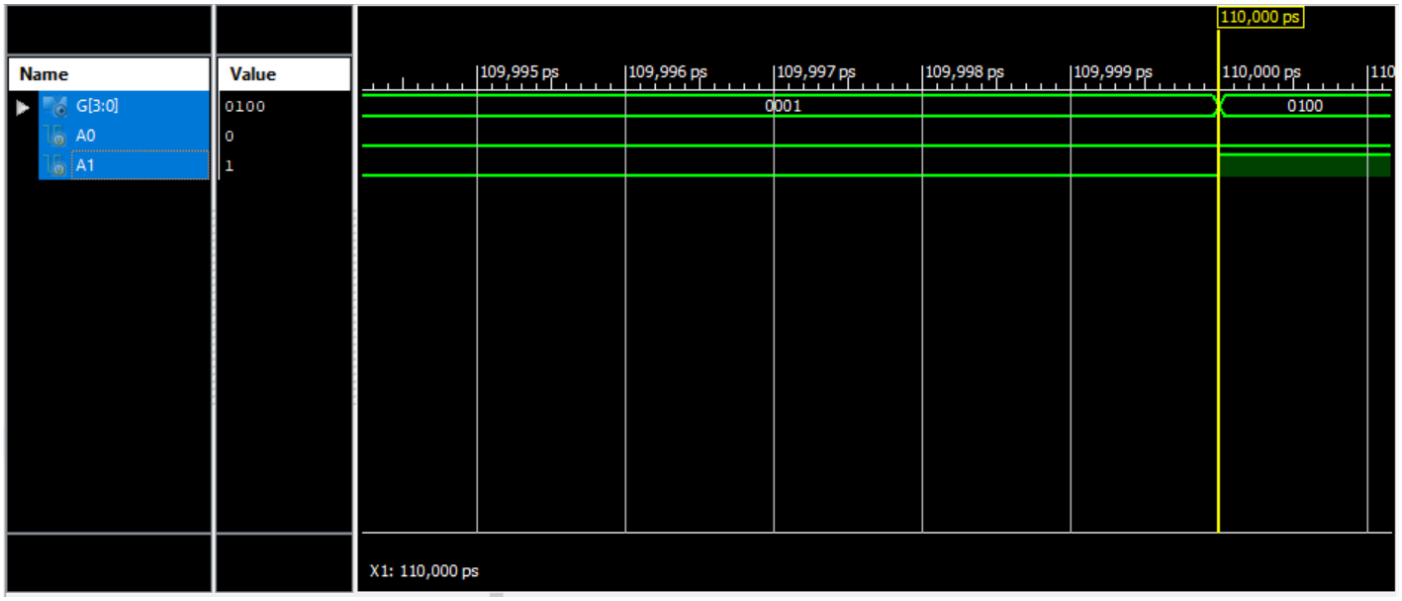
.A1(A1)

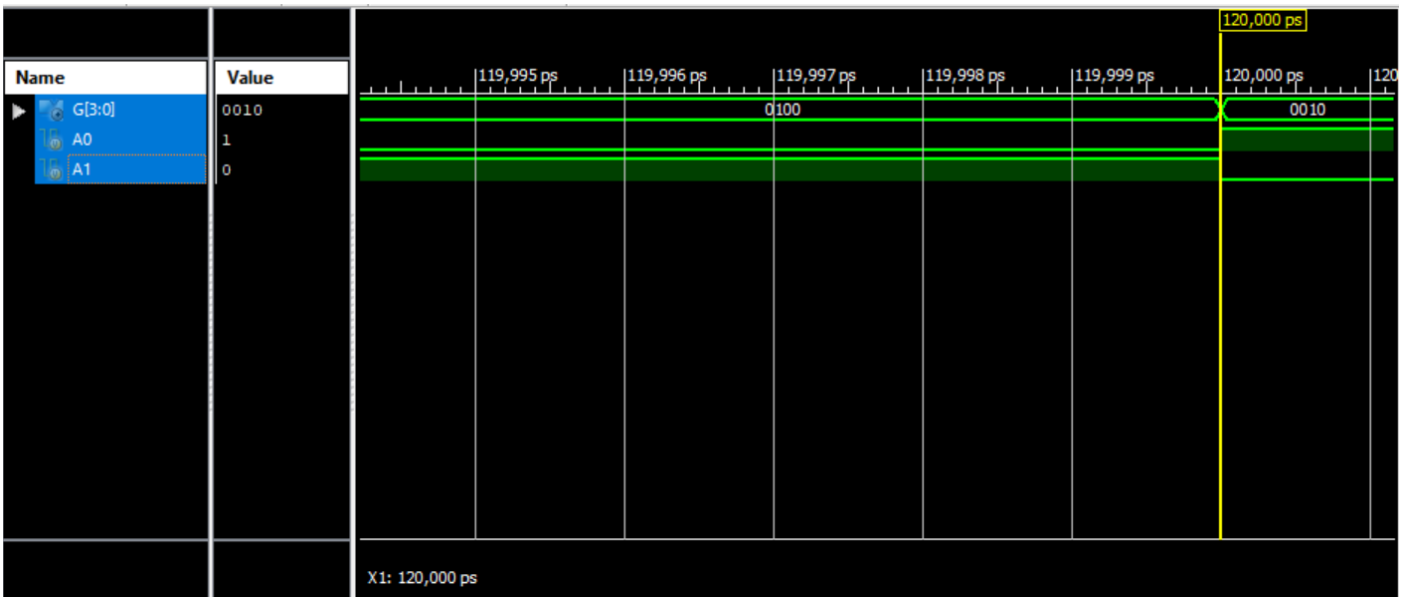
);

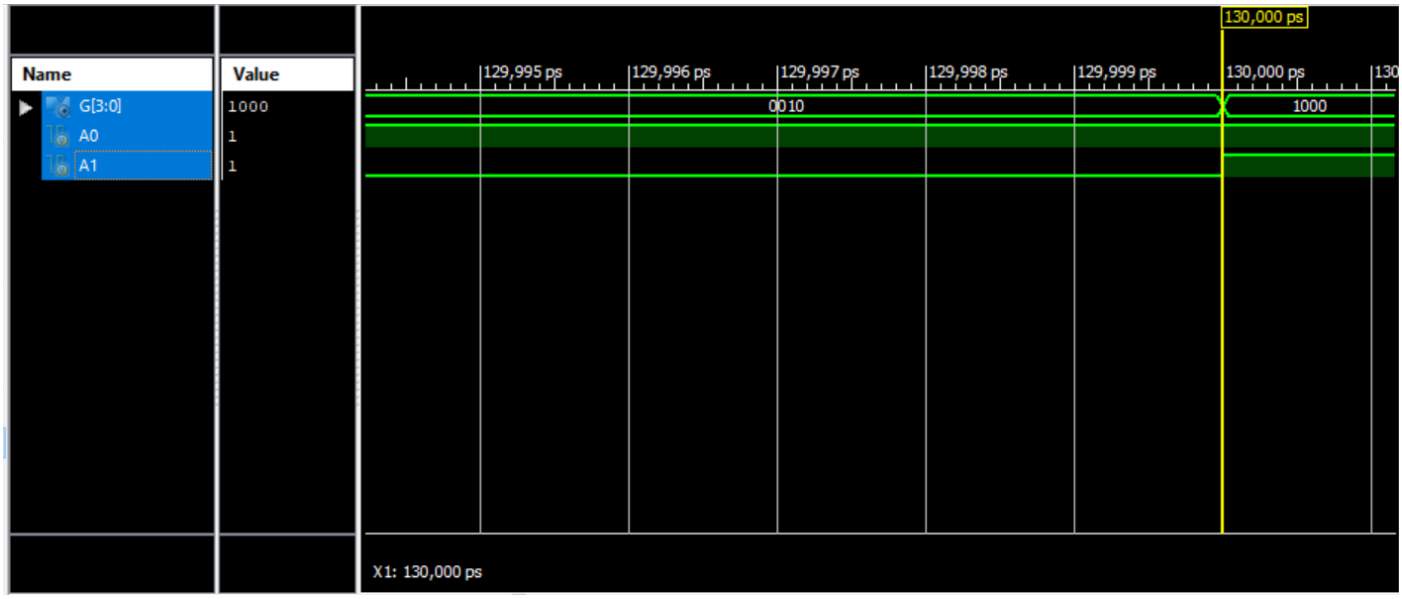
initial begin

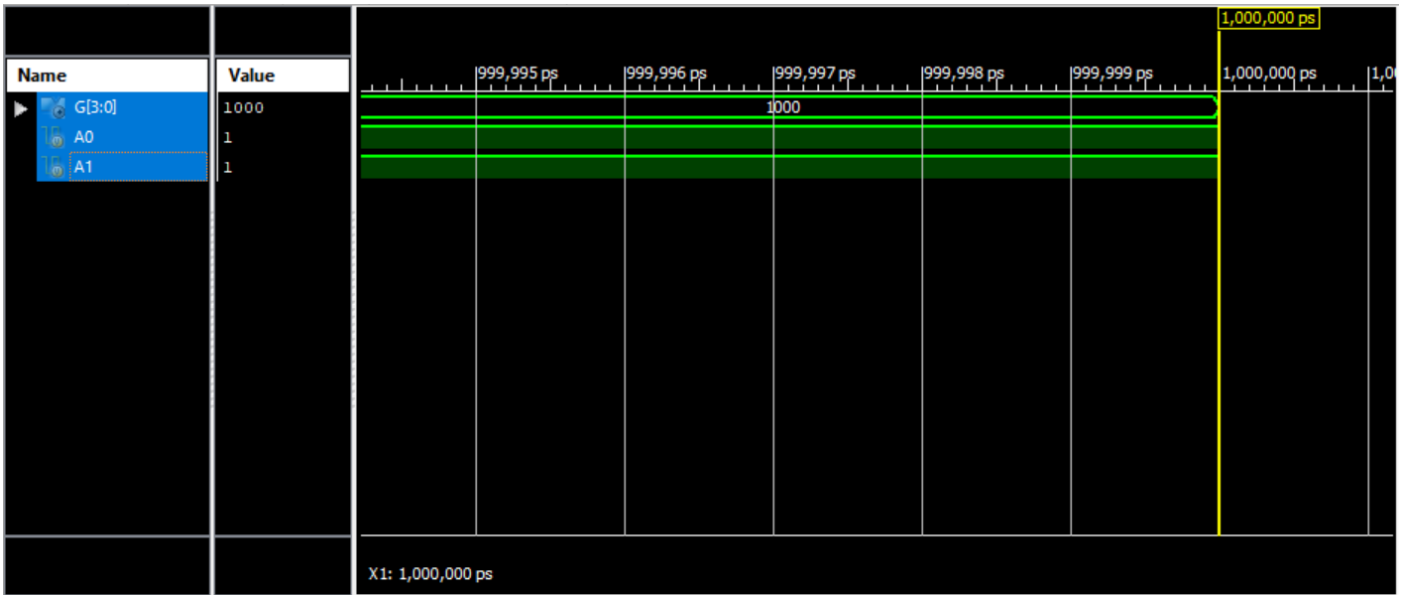
```
        // Initialize Inputs
        A0 = 0;
        A1 = 0;

        // Wait 100 ns for global reset to finish
        #100;
        A0 = 0;
        A1 = 0;
        #10
        A0 = 0;
        A1 = 1;
        #10
        A0 = 1;
        A1 = 0;
        #10
        A0 = 1;
        A1 = 1;

        // Add stimulus here

    end

endmodule
```

Simulations :


Name
Value
G[3:0]
A0
A1
0100
0
1
110,000 ps
109,995 ps
109,996 ps
109,997 ps
109,998 ps
109,999 ps
110,000 ps
0001
0100
X1: 110,000 ps


Name
Value
G[3:0]
A0
A1
0010
1
0
120,000 ps
119,995 ps
119,996 ps
119,997 ps
119,998 ps
119,999 ps
120,000 ps
0100
0010
X1: 120,000 ps


Name
Value
G[3:0]
A0
A1
1000
1
1
130,000 ps
129,995 ps
129,996 ps
129,997 ps
129,998 ps
129,999 ps
130,000 ps
0010
1000
X1: 130,000 ps



Part B:

b) Consider propagation delay of inverter **N1** is **5 time-units** and propagation delay of the and gate **A1** is **6 time-units** and change input values in testbench after each **5 timeunits**. Analyze the output.

Code:

```
module Task1b(output [3:0] G, input A0, A1);
  wire w1, w2, w3, w4;
  assign #5 w1=!A1;
  not_gate1 n2 (.A(w2), .B(A1));
  assign #6 G[0]=w1&w2;
  and_gate1 a2 (.Y(G[1]), .X(A0), .Z(w2));
  and_gate1 a3 (.Y(G[2]), .X(A1), .Z(w1));
  and_gate1 a4 (.Y(G[3]), .X(A0), .Z(A1));
endmodule

module and_gate1(output Y, input X, Z);
  assign Y = X & Z;
endmodule

module not_gate1(output A, input B);
  assign A = !B;
endmodule
```

module Task1b(output [3:0] G, input A0, A1);

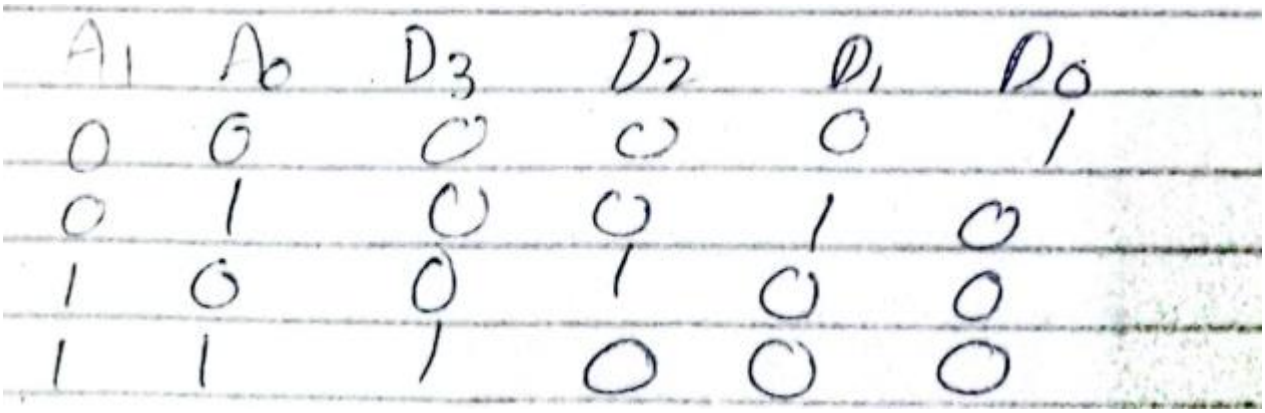
wire w1, w2, w3, w4;

assign #5 w1=!A1;

not_gate1 n2 (.A(w2), .B(A1));

```
  assign #6 G[0]=w1&w2;
  and_gate1 a2 (.Y(G[1]), .X(A0), .Z(w2));
  and_gate1 a3 (.Y(G[2]), .X(A1), .Z(w1));
  and_gate1 a4 (.Y(G[3]), .X(A0), .Z(A1));
endmodule

module and_gate1(output Y, input X, Z);
  assign Y = X & Z;
endmodule

module not_gate1(output A, input B);
  assign A = !B;
endmodule
```

TestBench:



```
module Task1bTb;

    // Inputs
    reg A0;
    reg A1;

    // Outputs
    wire [3:0] G;

    // Instantiate the Unit Under Test (UUT)
    Task1b uut (
        .G(G),
        .A0(A0),
        .A1(A1)
    );

    initial begin
        // Initialize Inputs
        A0 = 0;
        A1 = 0;

        // Wait 100 ns for global reset to finish
        #100;

        #5
        A0 = 0;
        A1 = 1;
        #5
        A0 = 1;
        A1 = 0;
        #5
        A0 = 0;
        A1 = 1;

        // Add stimulus here
```

module Task1bTb;

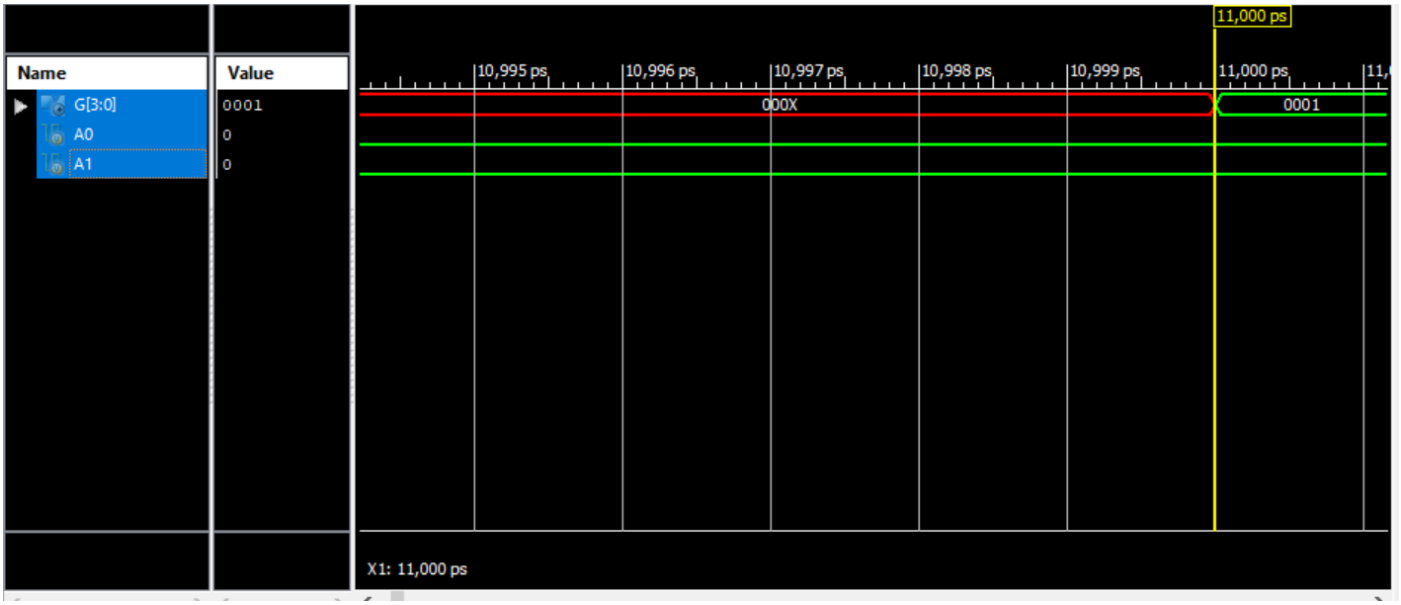
// Inputs
reg A0;
reg A1;

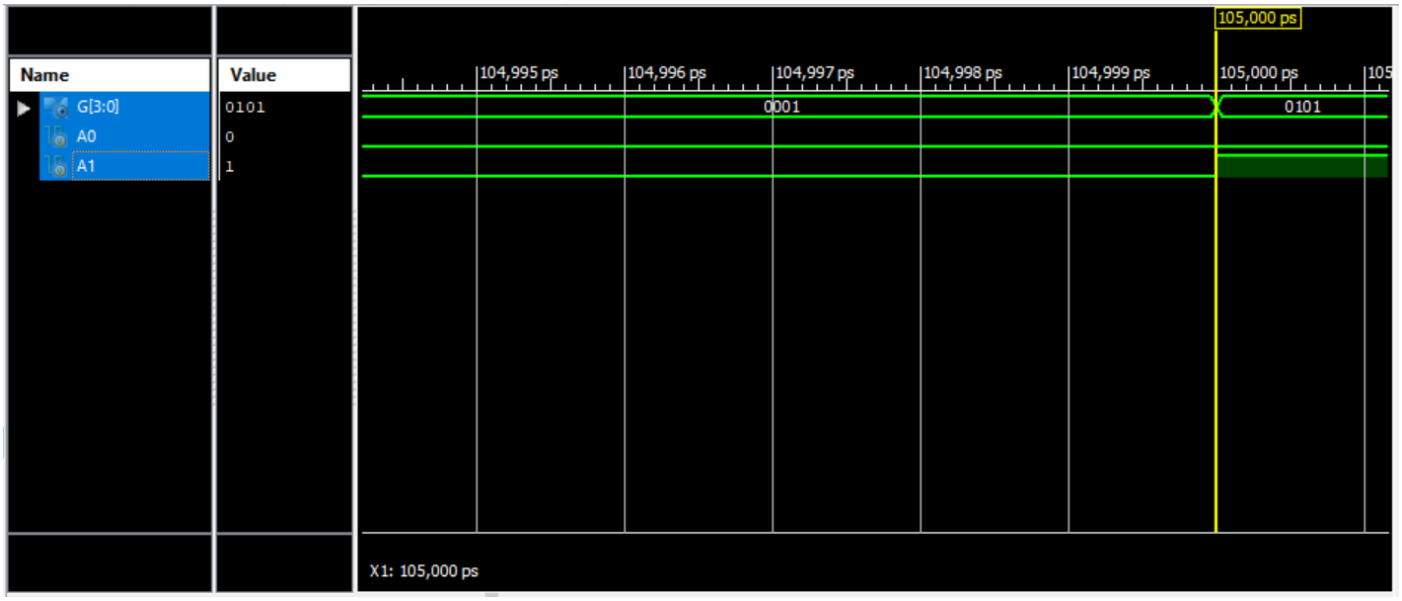
// Outputs
wire [3:0] G;

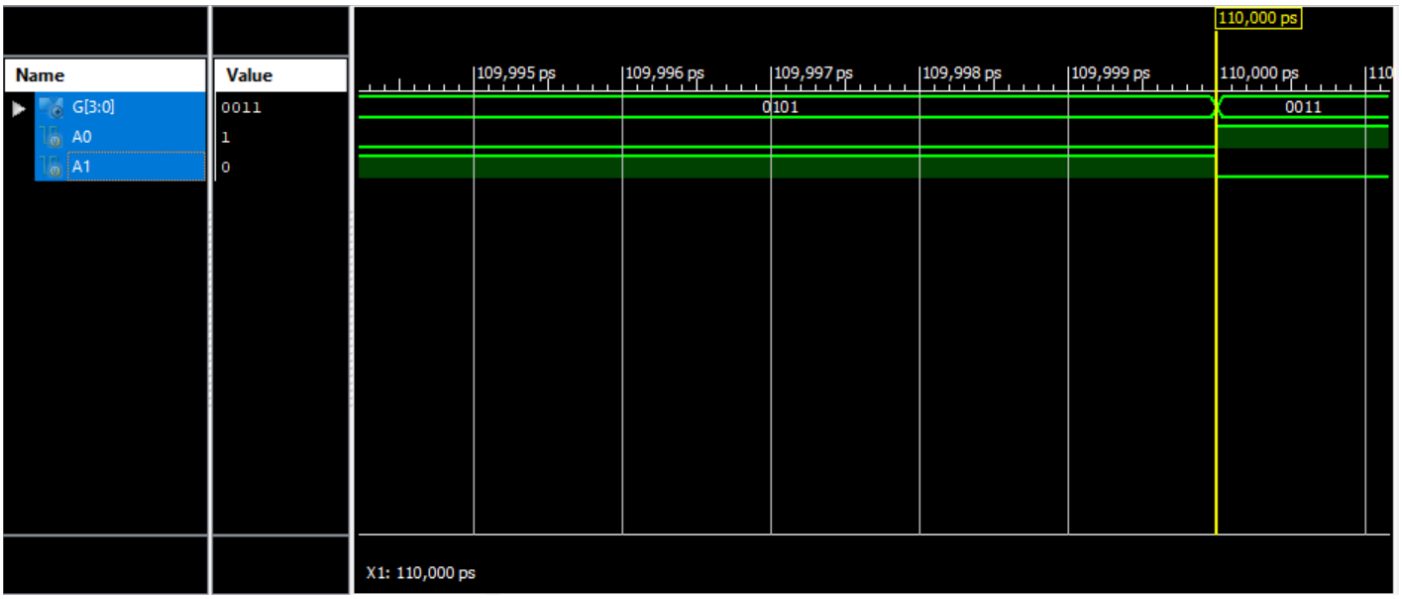
// Instantiate the Unit Under Test (UUT)
Task1b uut (
.G(G),
.A0(A0),
.A1(A1)

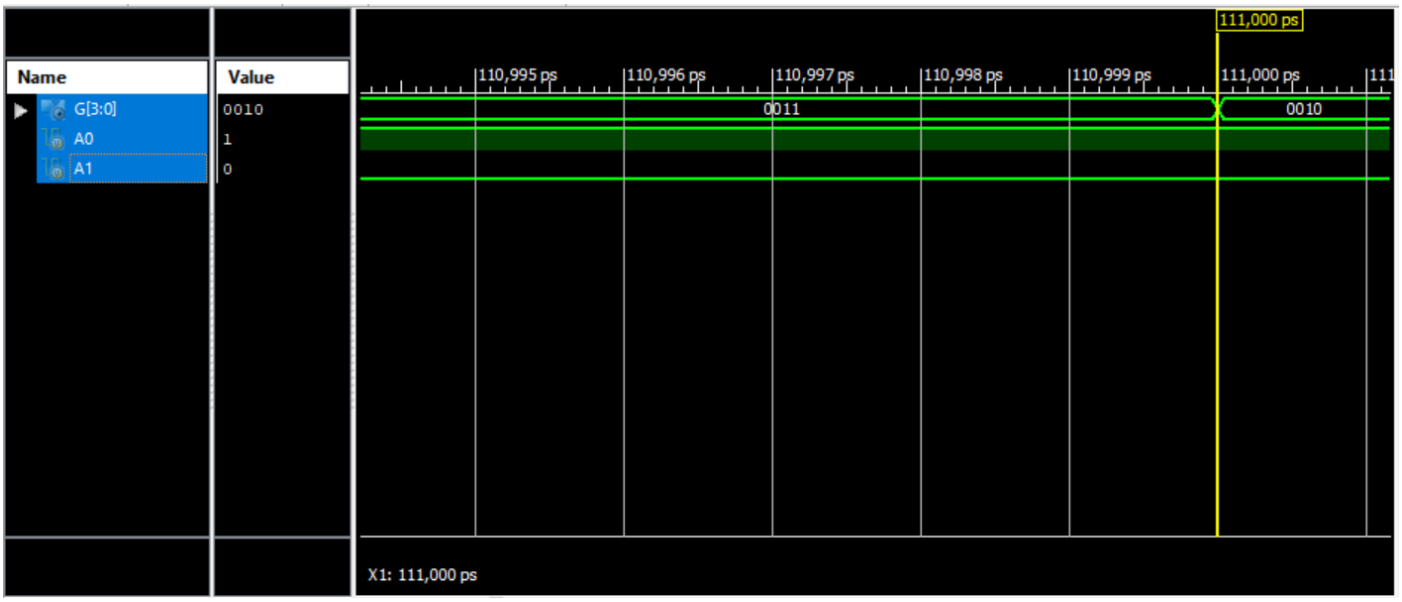
```
    );

    initial begin
        // Initialize Inputs
        A0 = 0;
        A1 = 0;

        // Wait 100 ns for global reset to finish
        #100;

        #5
        A0 = 0;
        A1 = 1;
        #5
        A0 = 1;
        A1 = 0;
        #5
        A0 = 0;
        A1 = 1;

        // Add stimulus here

    end

endmodule
```

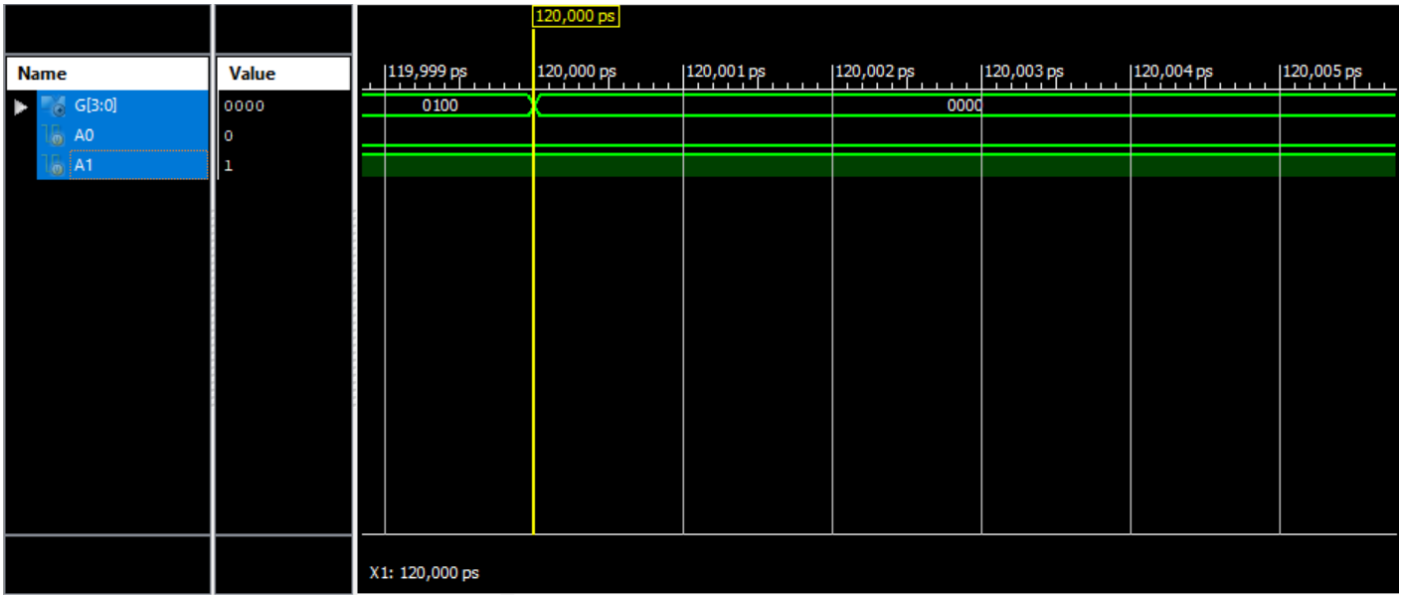
Simulations :


Name
Value
G[3:0]
A0
A1
0001
0
0
11,000 ps
10,995 ps
10,996 ps
10,997 ps
10,998 ps
10,999 ps
11,000 ps
000X
0001
X1: 11,000 ps


Name
Value
G[3:0]
A0
A1
0101
0
1
105,000 ps
104,995 ps
104,996 ps
104,997 ps
104,998 ps
104,999 ps
105,000 ps
0001
0101
X1: 105,000 ps


Name
Value
G[3:0]
A0
A1
0011
1
0
110,000 ps
109,995 ps
109,996 ps
109,997 ps
109,998 ps
109,999 ps
110,000 ps
0101
0011
X1: 110,000 ps


Name
Value
G[3:0]
A0
A1
0010
1
0
111,000 ps
110,995 ps
110,996 ps
110,997 ps
110,998 ps
110,999 ps
111,000 ps
0011
0010
X1: 111,000 ps


Name
Value
G[3:0]
A0
A1
0000
0
1
120,000 ps
119,999 ps
120,000 ps
120,001 ps
120,002 ps
120,003 ps
120,004 ps
120,005 ps
0100
0000
X1: 120,000 ps

Part C:

c) Now change the propagation delay of **A1** to **5 time-units** as well and keep changing the input values in testbench after each **5 time-units**. Analyze the output again.

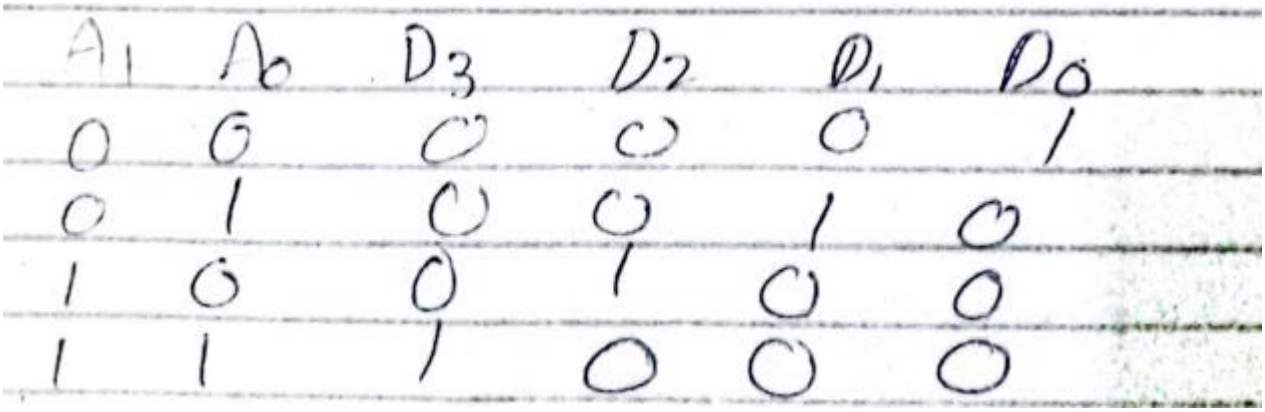
Code:

```
module Task1C(output [3:0] G, input A0, A1);
  wire w1, w2, w3, w4;

  not_gate2 n1 (.A(w1), .B(A0));
  not_gate2 n2 (.A(w2), .B(A1));
  assign #5 G[0]=w1&w2;
  and_gate2 a2 (.Y(G[1]), .X(A0), .Z(w2));
  and_gate2 a3 (.Y(G[2]), .X(A1), .Z(w1));
  and_gate2 a4 (.Y(G[3]), .X(A0), .Z(A1));
endmodule

module and_gate2(output Y, input X, Z);
  assign Y = X & Z;
endmodule

module not_gate2(output A, input B);
  assign A = !B;
endmodule
```


```
module Task1C(output [3:0] G, input A0, A1);
 wire w1, w2, w3, w4;

 not_gate2 n1 (.A(w1), .B(A0));
 not_gate2 n2 (.A(w2), .B(A1));
 assign #5 G[0]=w1&w2;
 and_gate2 a2 (.Y(G[1]), .X(A0), .Z(w2));
 and_gate2 a3 (.Y(G[2]), .X(A1), .Z(w1));
 and_gate2 a4 (.Y(G[3]), .X(A0), .Z(A1));
endmodule
```

```
module and_gate2(output Y, input X, Z);
  assign Y = X & Z;
endmodule

module not_gate2(output A, input B);
  assign A = !B;
endmodule
```

TestBench:

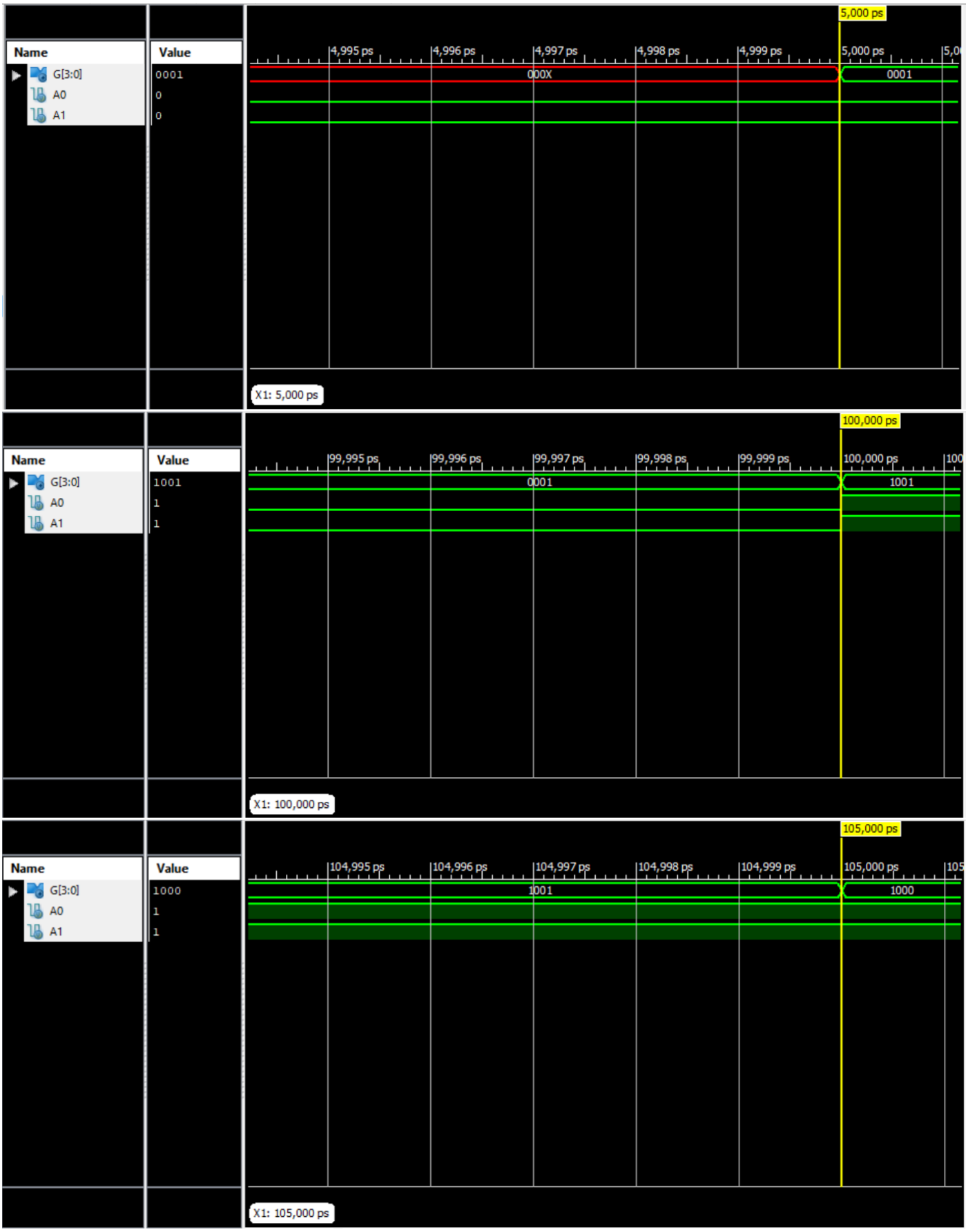


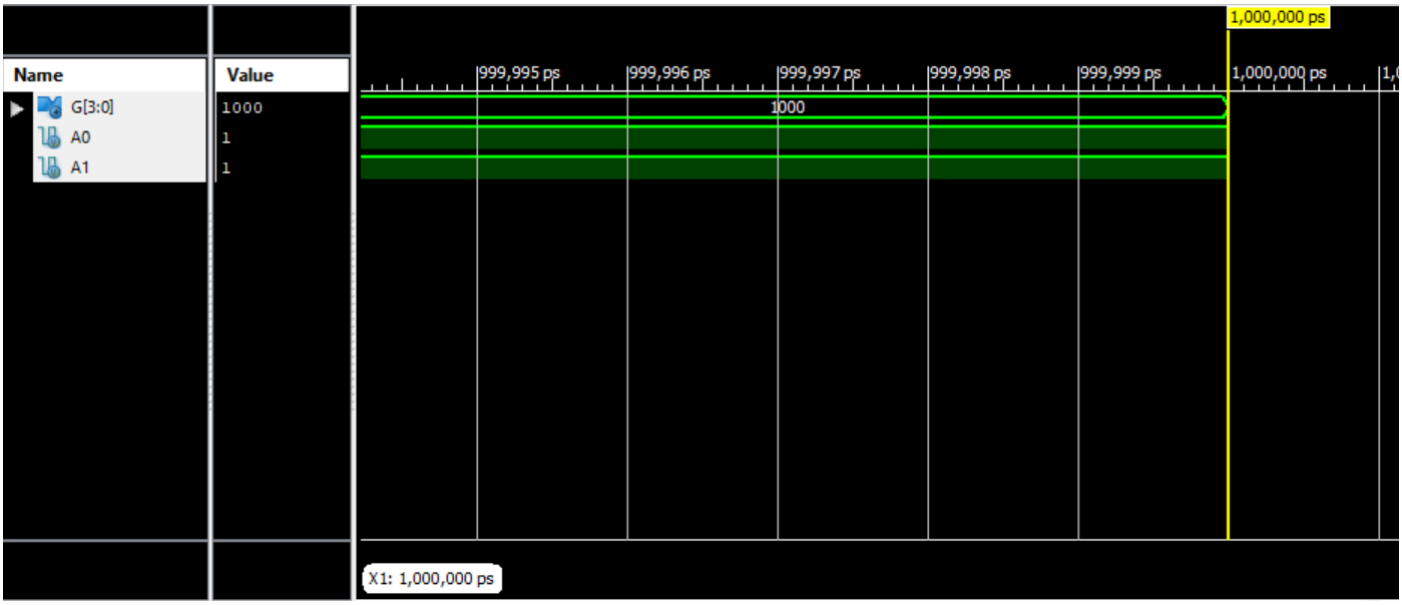

```
module Task1CTb;

    reg A0;
    reg A1;

    // Outputs
    wire [3:0] G;

    // Instantiate the Unit Under Test (UUT)
    Task1C uut (
        .G(G),
        .A0(A0),
        .A1(A1)
    );

    initial begin
        // Initialize Inputs
        A0 = 0;
        A1 = 0;

        // Wait 100 ns for global reset to finish
        #100;
        A0 = 0;
        A1 = 1;
        A0 = 1;
        A1 = 0;
        A0 = 1;
        A1 = 1;

        // Add stimulus here

    end

endmodule
```

module Task1CTb;

```
	reg A0;
	reg A1;

	// Outputs
	wire [3:0] G;

	// Instantiate the Unit Under Test (UUT)
	Task1C uut (
		.G(G),
		.A0(A0),
		.A1(A1)
	);

	initial begin
		// Initialize Inputs
		A0 = 0;
		A1 = 0;

		// Wait 100 ns for global reset to finish
		#100;
		A0 = 0;
		A1 = 1;
		A0 = 1;
		A1 = 0;
		A0 = 1;
		A1 = 1;

		// Add stimulus here

	end

endmodule
```

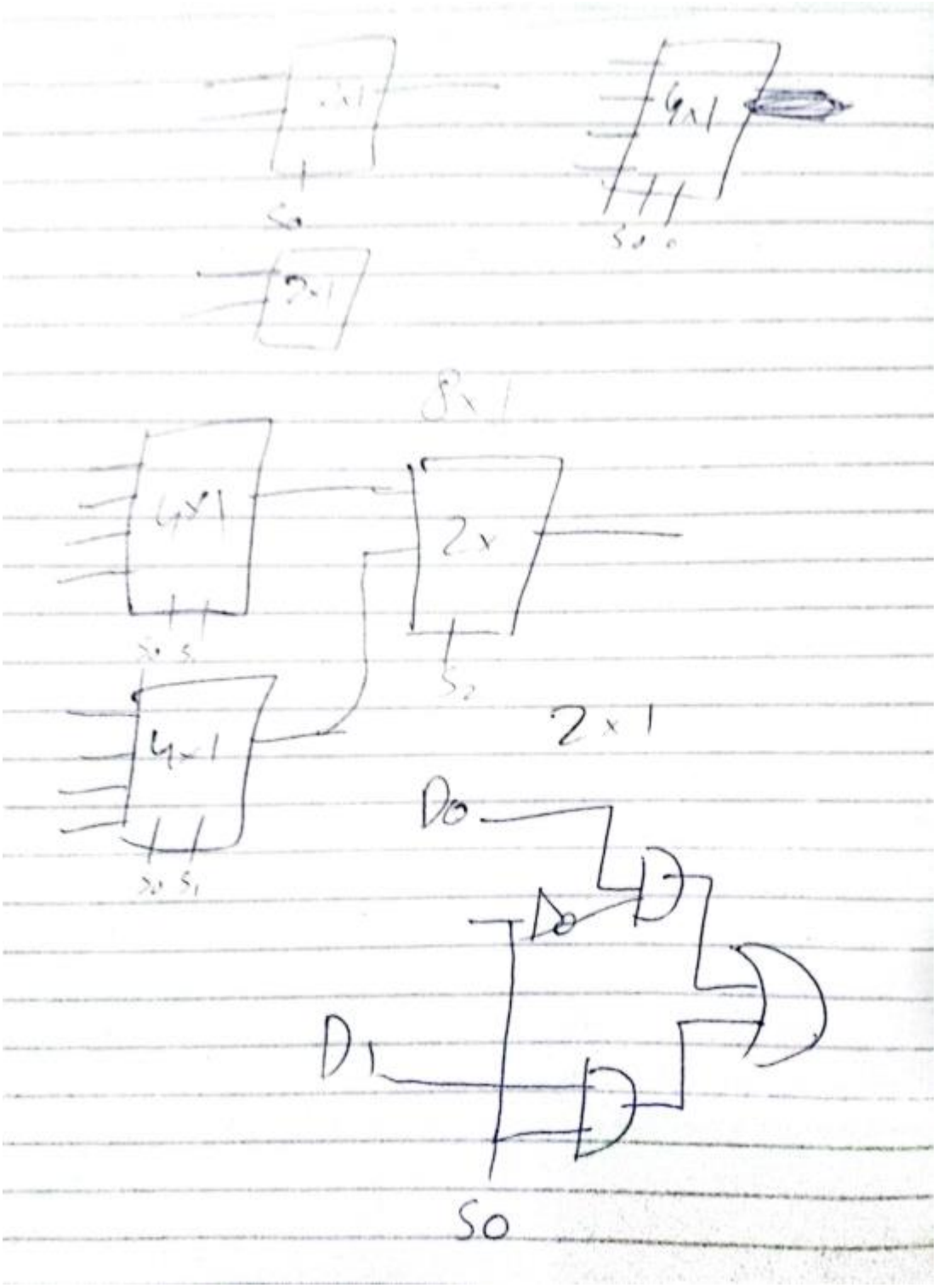
Simulations :



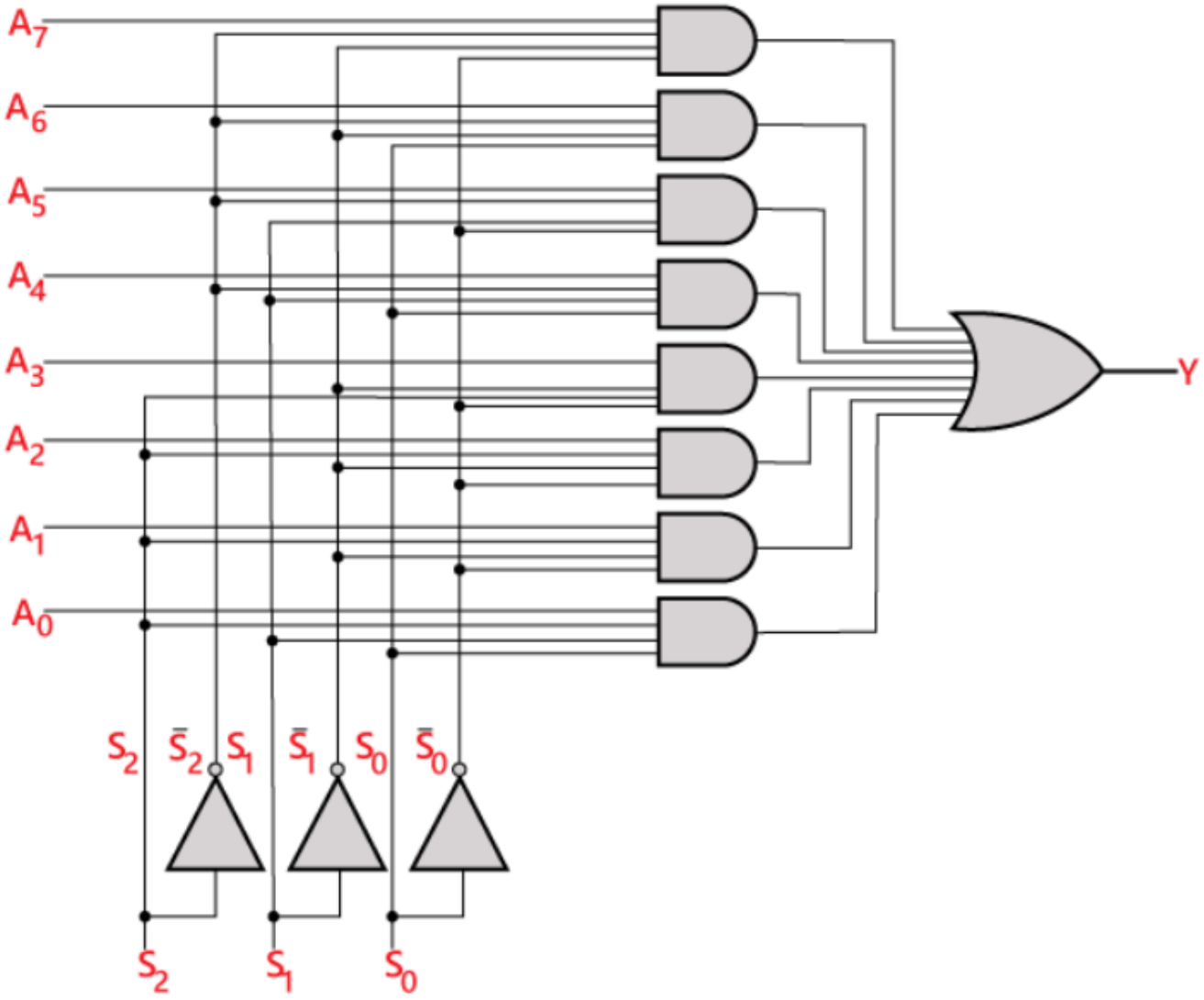


Question No 2:

2. Design and simulate 8 × 1 MUX using conditional operator in data-flow modeling. Show circuit diagram and truth table as well.


8×1
2×1
D0
D1
S0

INPUTS			Output
S_2	S_1	S_0	Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7



Code:

```
module m2x1_mux(output Y,input A0,A1,input S0);
wire w1,w2,w3;
assign Y=S0?A1:A0;

endmodule

module m4x1_mux(output Y,input A0,A1,A2,A3,input S0,S1);
wire w4,w5;
m2x1_mux m1(.Y(w4),.A0(A0),.A1(A1),.S0(S0));
m2x1_mux m2(.Y(w5),.A0(A2),.A1(A3),.S0(S0));
m2x1_mux m3(.Y(Y),.A0(w4),.A1(w5),.S0(S1));

endmodule

module  Task2(output Y,input A0,A1,A2,A3,A4,A5,A6,A7,input S0,S1,S2,S3);
wire w6,w7;
m4x1_mux M4(.Y(w6),.A0(A0),.A1(A1),.A2(A2),.A3(A3),.S0(SO),.S1(S1));
m4x1_mux M5(.Y(w7),.A0(A4),.A1(A5),.A2(A6),.A3(A7),.S0(S0),.S1(S1));
m2x1_mux m6(.Y(Y),.A0(w6),.A1(w7),.S0(S2));

endmodule
```


module m2x1_mux(output Y,input A0,A1,input S0);

wire w1,w2,w3;

assign Y=S0?A1:A0;

endmodule

module m4x1_mux(output Y,input A0,A1,A2,A3,input S0,S1);

wire w4,w5;

m2x1_mux m1(.Y(w4),.A0(A0),.A1(A1),.S0(S0));

m2x1_mux m2(.Y(w5),.A0(A2),.A1(A3),.S0(S0));

m2x1_mux m3(.Y(Y),.A0(w4),.A1(w5),.S0(S1));

endmodule

module Task2(output Y,input A0,A1,A2,A3,A4,A5,A6,A7,input S0,S1,S2,S3);

wire w6,w7;

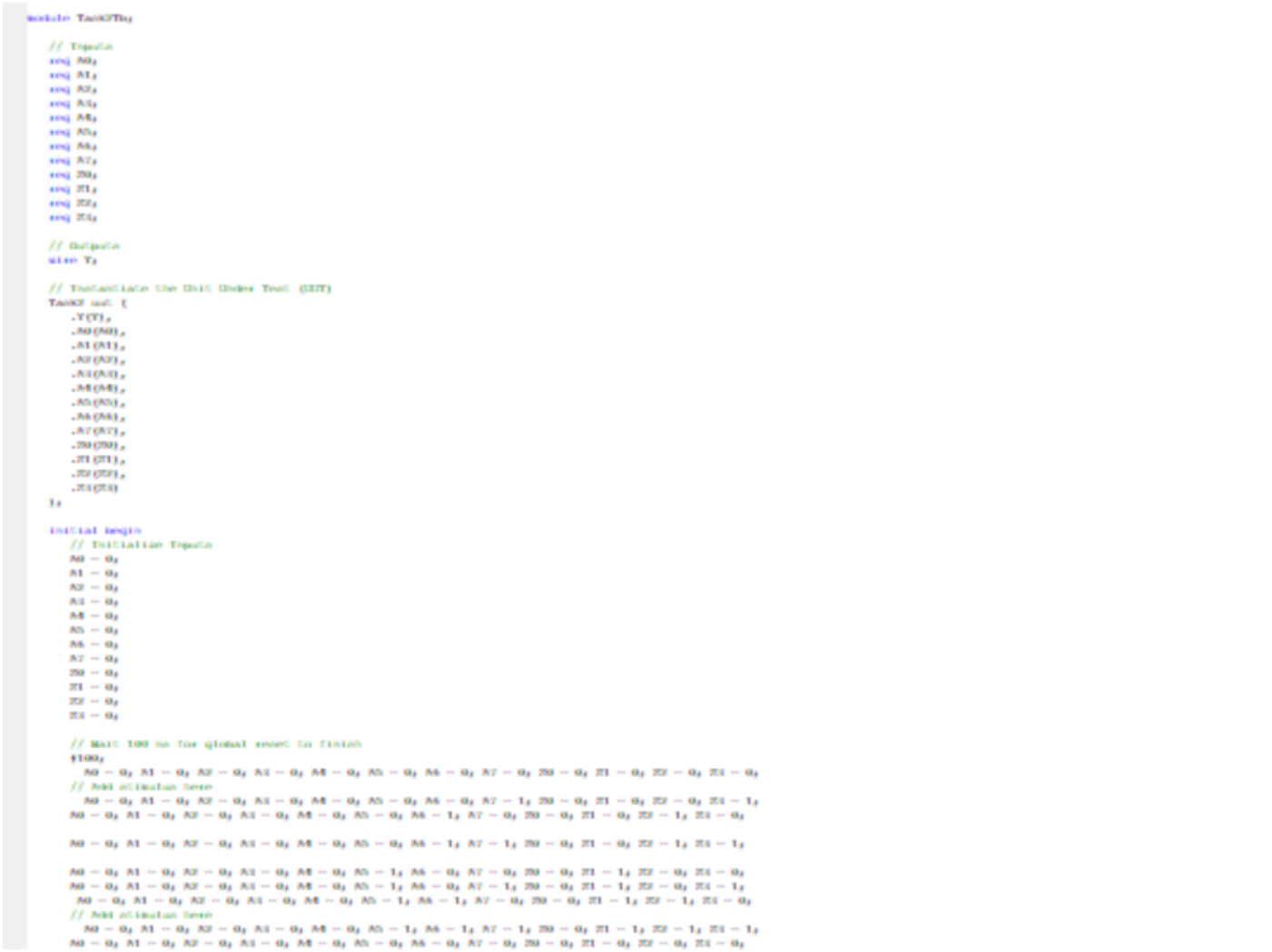
m4x1_mux M4(.Y(w6),.A0(A0),.A1(A1),.A2(A2),.A3(A3),.S0(SO),.S1(S1));

m4x1_mux M5(.Y(w7),.A0(A4),.A1(A5),.A2(A6),.A3(A7),.S0(S0),.S1(S1));

m2x1_mux m6(.Y(Y),.A0(w6),.A1(w7),.S0(S2));

endmodule

TestBench:



module Task2Tb;

// Inputs

reg A0;

reg A1;

reg A2;

reg A3;

reg A4;

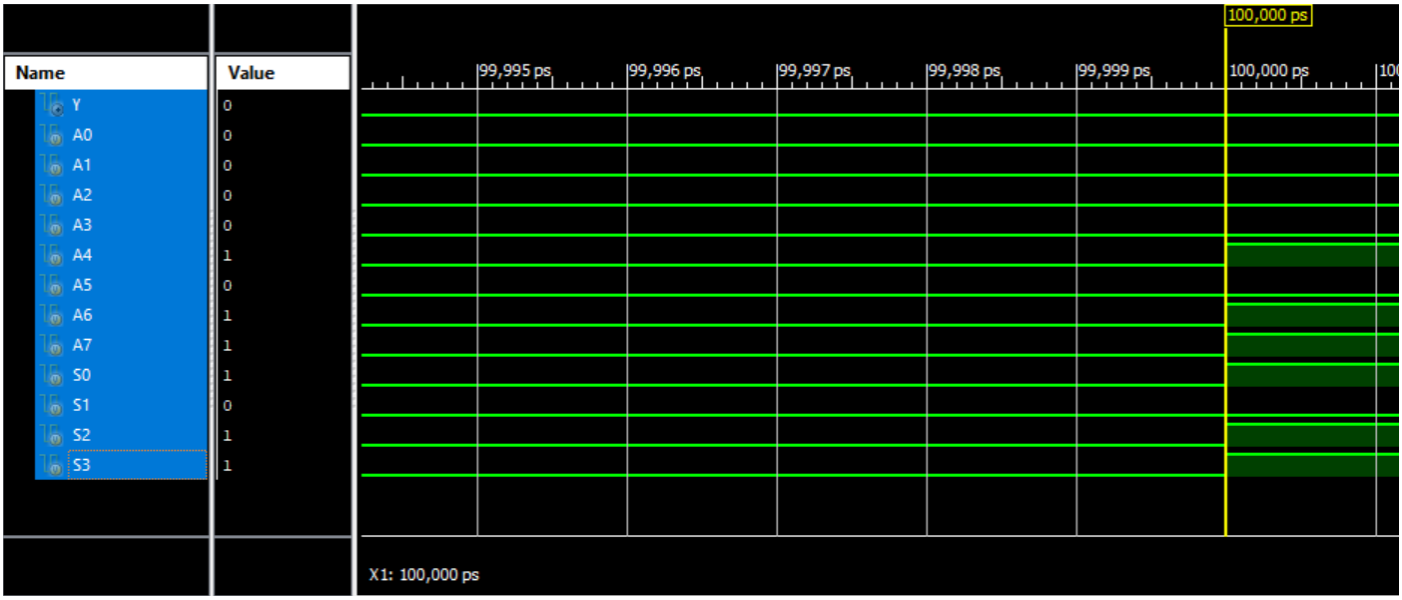
reg A5;

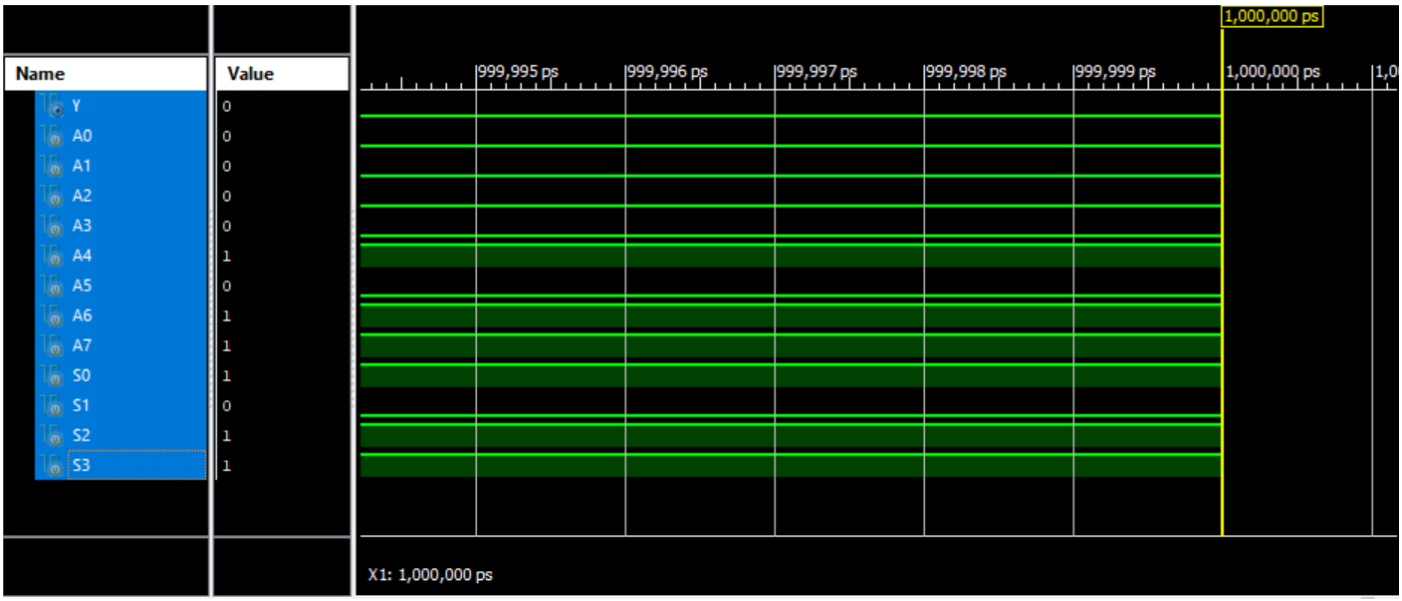
reg A6;

```
reg A7;
reg S0;
reg S1;
reg S2;
reg S3;

// Outputs
wire Y;

// Instantiate the Unit Under Test (UUT)
Task2 uut (
	.Y(Y),
	.A0(A0),
	.A1(A1),
	.A2(A2),
	.A3(A3),
	.A4(A4),
	.A5(A5),
	.A6(A6),
	.A7(A7),
	.S0(S0),
	.S1(S1),
	.S2(S2),
	.S3(S3)
);

initial begin
	// Initialize Inputs
	A0 = 0;
	A1 = 0;
	A2 = 0;
	A3 = 0;
	A4 = 0;
	A5 = 0;
```

```
        A6 = 0;
        A7 = 0;
        S0 = 0;
        S1 = 0;
        S2 = 0;
        S3 = 0;

        // Wait 100 ns for global reset to finish
        #100;
A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 0; A7 = 0; S0 = 0; S1 = 0; S2 = 0; S3 = 0;
        // Add stimulus here
A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 0; A7 = 1; S0 = 0; S1 = 0; S2 = 0; S3 = 1;
        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 1; A7 = 0; S0 = 0; S1 = 0; S2 = 1; S3 = 0;

        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 1; A7 = 1; S0 = 0; S1 = 0; S2 = 1; S3 = 1;

        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 1; A6 = 0; A7 = 0; S0 = 0; S1 = 1; S2 = 0; S3 = 0;
        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 1; A6 = 0; A7 = 1; S0 = 0; S1 = 1; S2 = 0; S3 = 1;
         A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 1; A6 = 1; A7 = 0; S0 = 0; S1 = 1; S2 = 1; S3 = 0;
        // Add stimulus here
A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 1; A6 = 1; A7 = 1; S0 = 0; S1 = 1; S2 = 1; S3 = 1;
        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 0; A7 = 0; S0 = 0; S1 = 0; S2 = 0; S3 = 0;

        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 1; A5 = 0; A6 = 0; A7 = 1; S0 = 1; S1 = 0; S2 = 0; S3 = 1;

        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 1; A5 = 0; A6 = 1; A7 = 0; S0 = 1; S1 = 0; S2 = 1; S3 = 0;
        A0 = 0; A1 = 0; A2 = 0; A3 = 0; A4 = 1; A5 = 0; A6 = 1; A7 = 1; S0 = 1; S1 = 0; S2 = 1; S3 = 1;


    end

endmodule
```

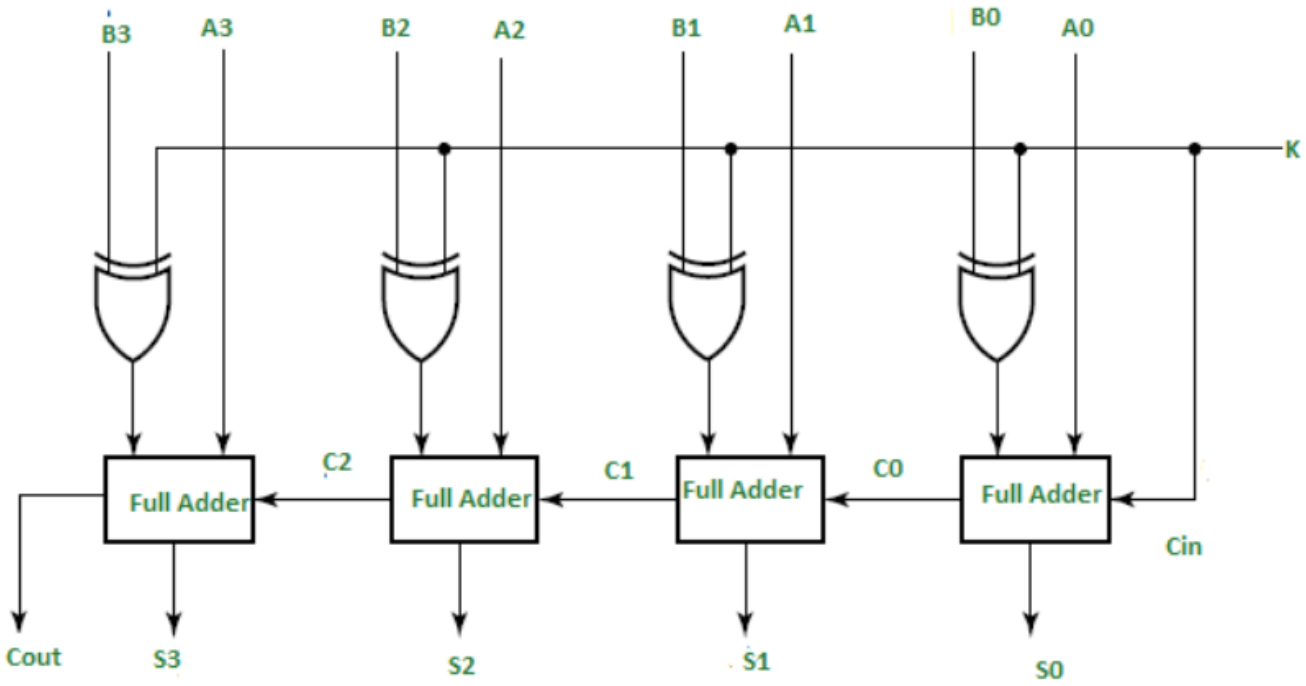
Simulations :





Question:

3. Design and simulate 4-bit **Subtractor**. Show the block diagram and truth table as well.



Cin	A				B				Difference				Borrow
	A3	A2	A1	A0	B3	B2	B1	B0	d_3	d_2	d_1	d_0	Bout
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0
0	0	0	1	1	0	0	1	1	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	0	0	0	0
0	0	1	1	1	0	1	1	1	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0	1	0	0	0	0	0
0	1	0	1	0	1	0	1	0	0	0	0	0	0
0	1	0	1	1	1	0	1	1	0	0	0	0	0
0	1	1	0	0	1	1	0	0	0	0	0	0	0
0	1	1	0	1	1	1	0	1	0	0	0	0	0
0	1	1	1	0	1	1	1	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0

Code:

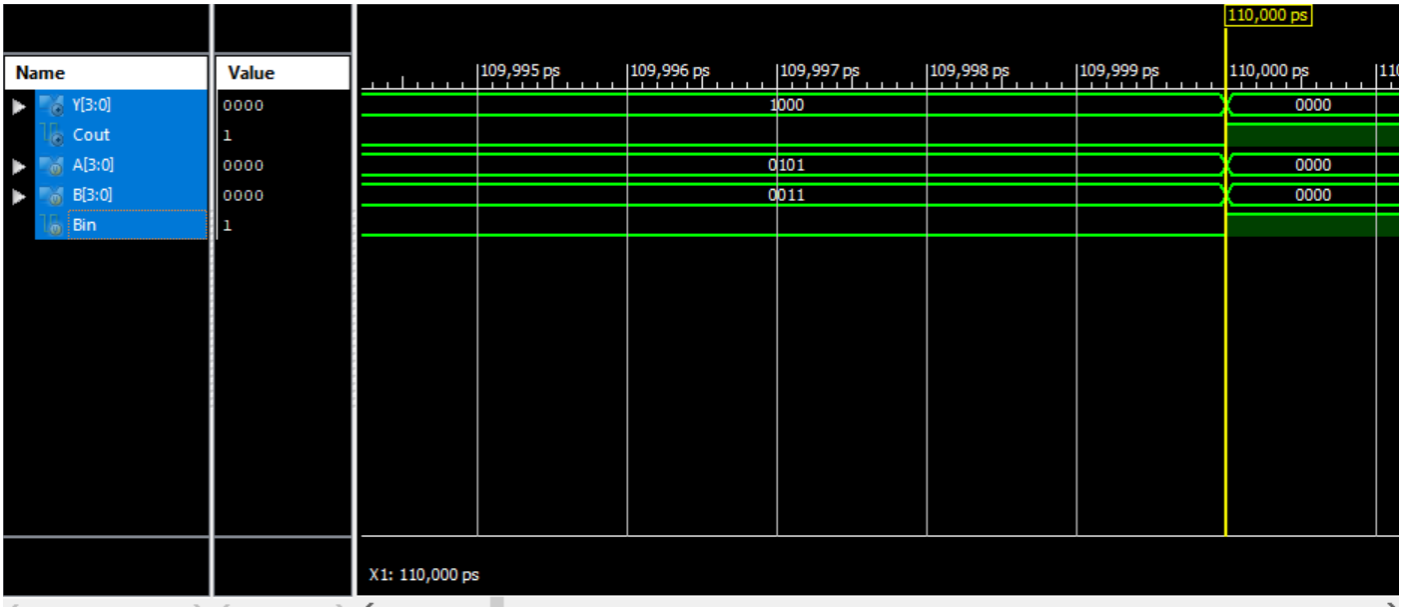
```
module Task3(
output [3:0] Y
,output Cout
,input[3:0] A
,input [3:0] B,
input Bin );
wire [3:0] W;
wire C0,C1,C2,C3;
wire X0,X1,X2,X3;
//xor
assign X3=B[3]^Bin;
assign X2=B[2]^Bin;
assign X1=B[1]^Bin;
assign X0=B[0]^Bin;

full_1_adder fa1(Y[0],C0,A[0],X0,Bin);
full_1_adder fa2(Y[1],C1,A[1],X1,C0);
full_1_adder fa3(Y[2],C2,A[2],X2,C1);
full_1_adder fa4(Y[3],Cout,A[3],X3,C2);



endmodule
module full_1_adder(output Y,Cout,input A,B,Cin);

assign w1=A^B;
assign Y=w1^Cin;
assign w2=w1&Cin;
assign w3=A&B;
assign Cout=w2+w3;



endmodule
```

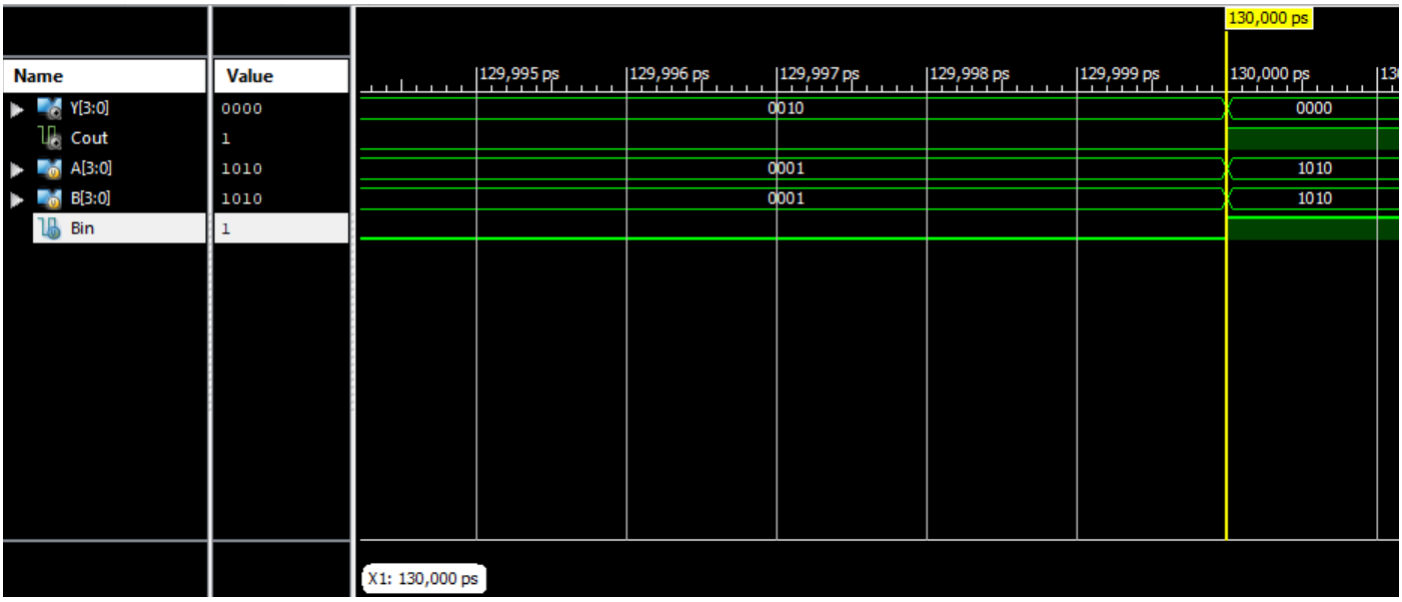
TestBench:

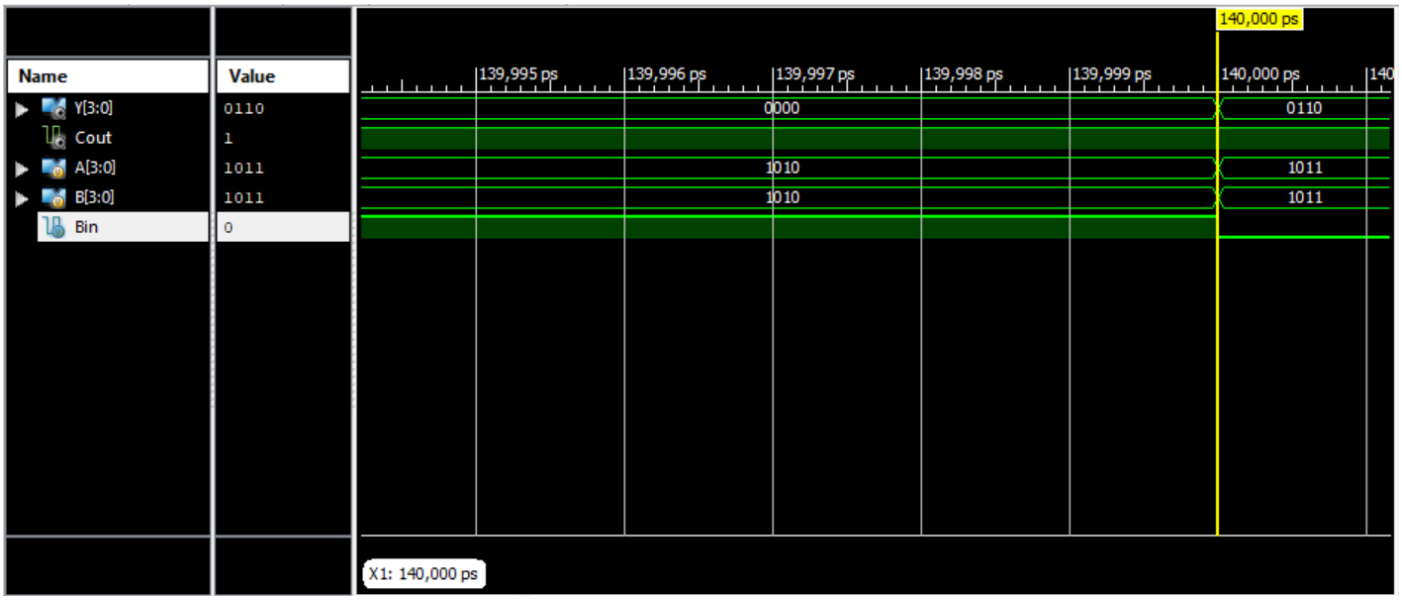
```
module Task3Tb;

	// Inputs
	reg [3:0] A;
	reg [3:0] B;
	reg Bin;

	// Outputs
	wire [3:0] Y;
	wire Cout;

	// Instantiate the Unit Under Test (UUT)
	Task3 uut (
		.Y(Y),
		.Cout(Cout),
		.A(A),
		.B(B),
		.Bin(Bin)
	);

	initial begin
		// Initialize In
		A = 4'b0101;
  B = 4'b0011;
  Bin = 0;
		#100;
		#10	A = 0000;B = 0000;Bin = 1;
		#10	A = 00001;B = 0001;Bin = 0;
		#10	A = 0010;B = 0010;Bin = 1;
		#10	A = 0011;B = 0011;Bin = 0;
		#10	A = 0100;B = 0100;Bin = 1;
		#10	A = 0101;B = 0101;Bin = 0;
		#10	A = 0110;B = 0110;Bin = 1;
```

```
        #10     A = 0111;B = 0111;Bin = 0;


        #10     A = 1000;B = 1000;Bin = 1;
        #10     A = 1001;B = 1001;Bin = 0;
        #10     A = 1010;B = 1010;Bin = 1;
        #10     A = 1011;B = 1011;Bin = 0;
        #10     A = 1100;B = 1100;Bin = 1;
        #10     A = 1101;B = 1101;Bin = 0;
        #10     A = 1110;B = 1110;Bin = 1;
        #10     A = 1111;B = 1111;Bin = 0;

        // Add stimulus here

    end

endmodule
```

Simulations :




Name
Value
Y[3:0]
0000
Cout
1
A[3:0]
1010
B[3:0]
1010
Bin
1
129,995 ps
129,996 ps
129,997 ps
129,998 ps
129,999 ps
130,000 ps
0010
0000
0001
1010
0001
1010
X1: 130,000 ps


Name
Value
Y[3:0]
0110
Cout
1
A[3:0]
1011
B[3:0]
1011
Bin
0
139,995 ps
139,996 ps
139,997 ps
139,998 ps
139,999 ps
140,000 ps
0000
0110
1010
1011
1010
1011
X1: 140,000 ps