DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 8, 2024

DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-1

GIVEN A 4-VARIABLE LOGIC EXPRESSION, SIMPLIFY IT USING APPROPRIATE TECHNIQUE AND SIMULATE THE SAME USING BASIC GATES.

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Diagram No –(1.1).1: LOGICAL EXPRESSION CONSIDERED FOR SIMULATION

4-VARIABLE LOGICAL EXPRESSION CONSIDERED FOR SIMULATION:

Y = (A’B’CD’) + (A’BCD’) + (ABCD’) + (AB’CD’) + (AB’C’D’) + (AB’C’D) + (AB’CD)

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Diagram No –(1.1).2: TRUTH TABLE FOR THE GIVEN 4-VARIABLE LOGIC EXPRESSION

DECIMAL NUMBER	INPUTS				OUTPUT
	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Diagram No –(1.1).2: TRUTH TABLE FOR THE GIVEN 4-VARIABLE LOGIC EXPRESSION

DECIMAL NUMBER	INPUTS				OUTPUT
	A	B	C	D	Y
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Diagram No –(1.1).3: K-MAP SIMPLIFICATION OF THE LOGICAL EXPRESSION CONSIDERED FOR SIMULATION

cd \ ab	a'b' 00	a'b 01	ab 11	ab' 10
c'd' 00	0 (0)	0 (4)	0 (12)	1 (8)
c'd 01	0 (1)	0 (5)	0 (13)	1 (9)
cd 11	0 (3)	0 (7)	0 (15)	1 (11)
cd' 10	1 (2)	1 (6)	1 (14)	1 (10)

Expression by considering the highlighted Group of four 1's = AB'

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Diagram No –(1.1).3: K-MAP SIMPLIFICATION OF THE LOGICAL EXPRESSION CONSIDERED FOR SIMULATION

ab / cd	a'b' 00	a'b 01	ab 11	ab' 10
c'd' 00	0 (0)	0 (4)	0 (12)	1 (8)
c'd 01	0 (1)	0 (5)	0 (13)	1 (9)
cd 11	0 (3)	0 (7)	0 (15)	1 (11)
cd' 10	1 (2)	1 (6)	1 (14)	1 (10)

Expression by considering the highlighted Group of four 1's = CD'

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

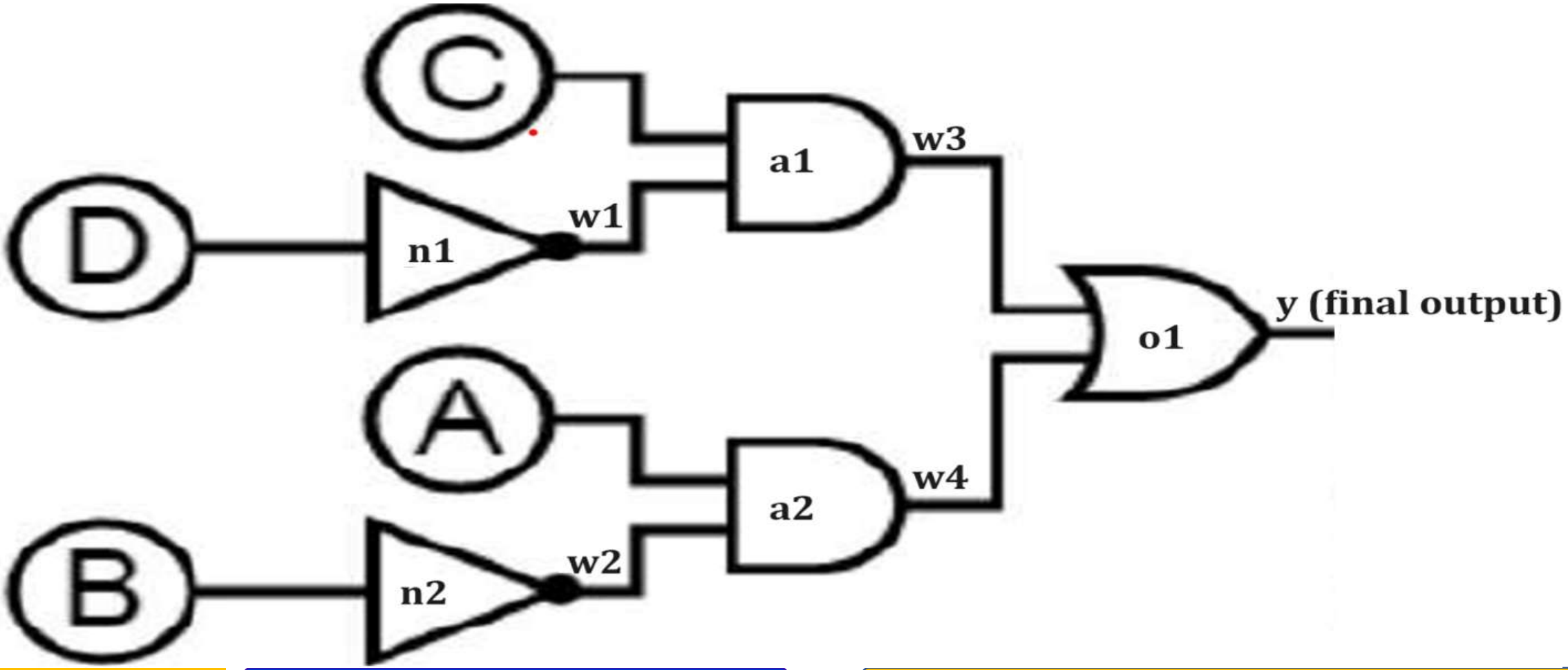
Diagram No –(1.1).4: LOGICAL EXPRESSION CONSIDERED FOR SIMULATION AFTER SIMPLIFICATION

4-VARIABLE LOGICAL EXPRESSION CONSIDERED FOR SIMULATION AFTER REDUCING BY SIMPLIFICATION METHOD:

$$Y = (AB') + (CD')$$

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Diagram No –(1.1).5: LOGICAL CIRCUIT DIAGRAM FOR THE GIVEN 4-VARIABLE LOGIC EXPRESSION USING BASIC GATES AS PER SIMPLIFIED EXPRESSION



PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Circuit description module Verilog Program example for implementing 4-variable logic expression

```
module circuit_4variable(A,B,C,D,Y);
        input A,B,C,D;
        output Y;
        wire w1,w2,w3,w4;

        not n1(w1,D);
        not n2(w2,B);
        and a1(w3,C,w1);
        and a2(w4,A,w2);
        or o1(Y,w3,w4);

endmodule
```

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Test bench waveform module Verilog Program example for implementing 4-variable logic expression

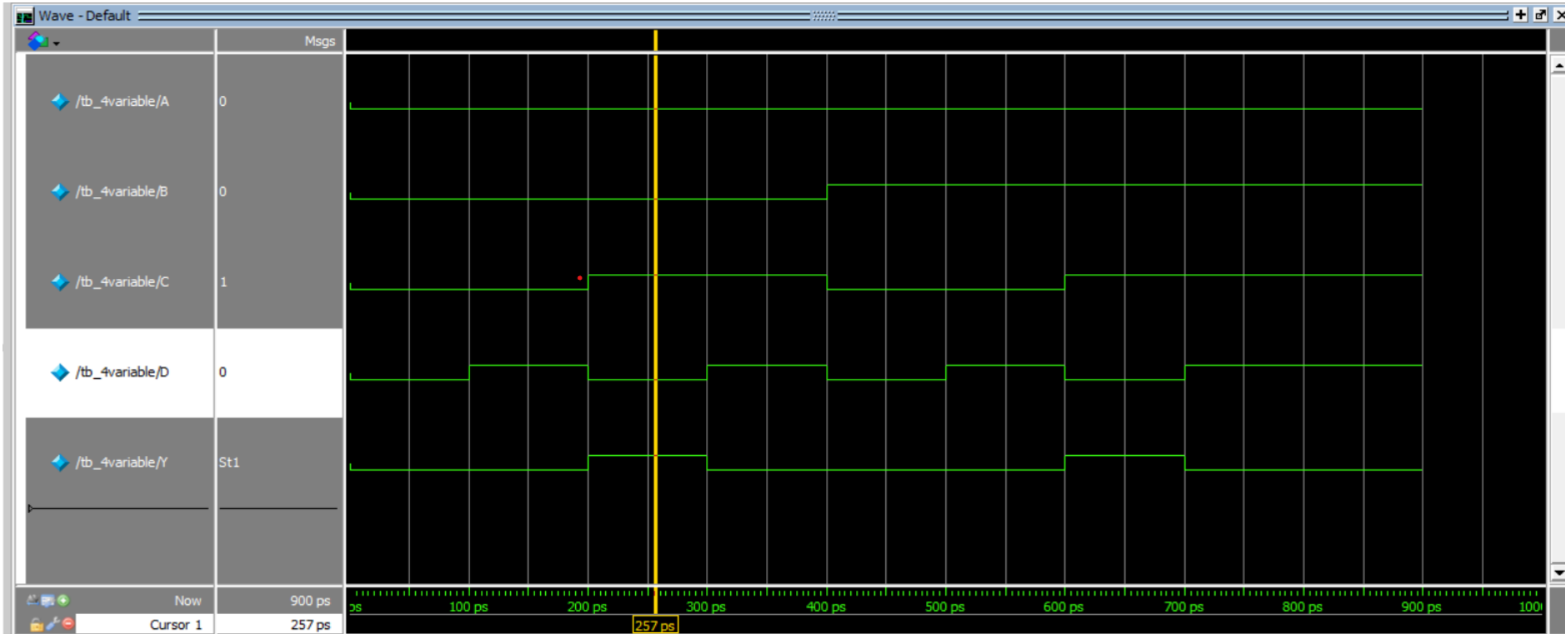
```
module tb_4variable();
   reg A,B,C,D;
   wire Y;

    circuit_4variable  uut(.A(A),  .B(B),  .C(C),  .D(D),
   .Y(Y));

   initial
          begin
                $monitor("A=%b B=%b C=%b
                      D=%b    Y=%b", A,B,C,D,Y);
               A=0; B=0;C=0;D=0;          #100
               A=0; B=0;C=0;D=1;          #100
               A=0; B=0;C=1;D=0;          #100
```

Test bench waveform module Verilog Program example for implementing 4-variable logic expression

```
A=0; B=0;C=1;D=1;  #100
A=0; B=1;C=0;D=0;  #100
A=0; B=1;C=0;D=1;  #100
A=0; B=1;C=1;D=0;  #100
A=0; B=1;C=1;D=1;

          end
endmodule
```

PROGRAM-1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF 4-VARIABLE LOGIC EXPRESSION USING BASIC LOGIC GATES

Diagram No –(1.1).6: output screen shot



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 6, 2024

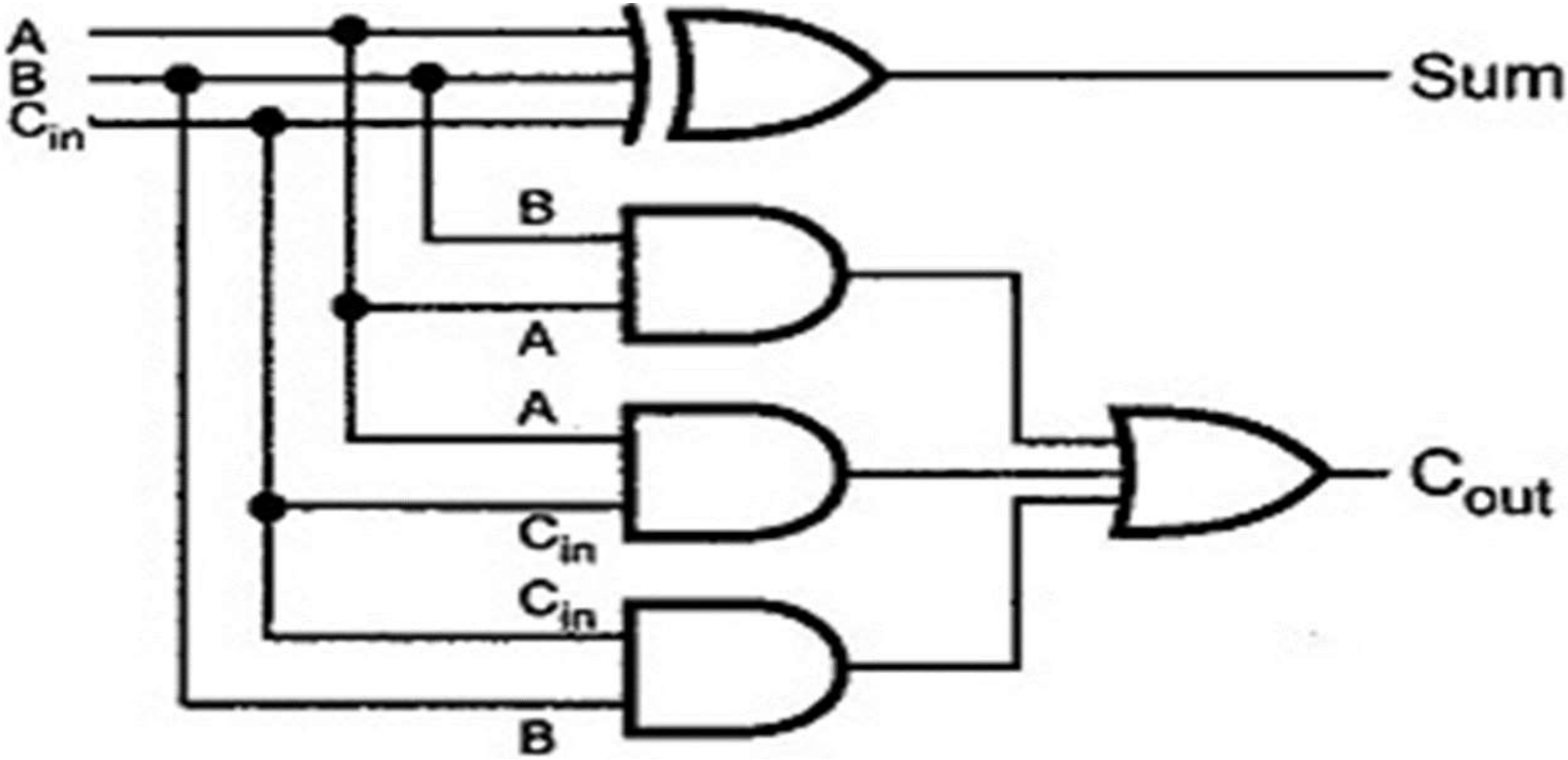
DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-2

DESIGN A 4 BIT FULL ADDER AND SUBTRACTOR AND SIMULATE THE SAME USING BASIC GATES.

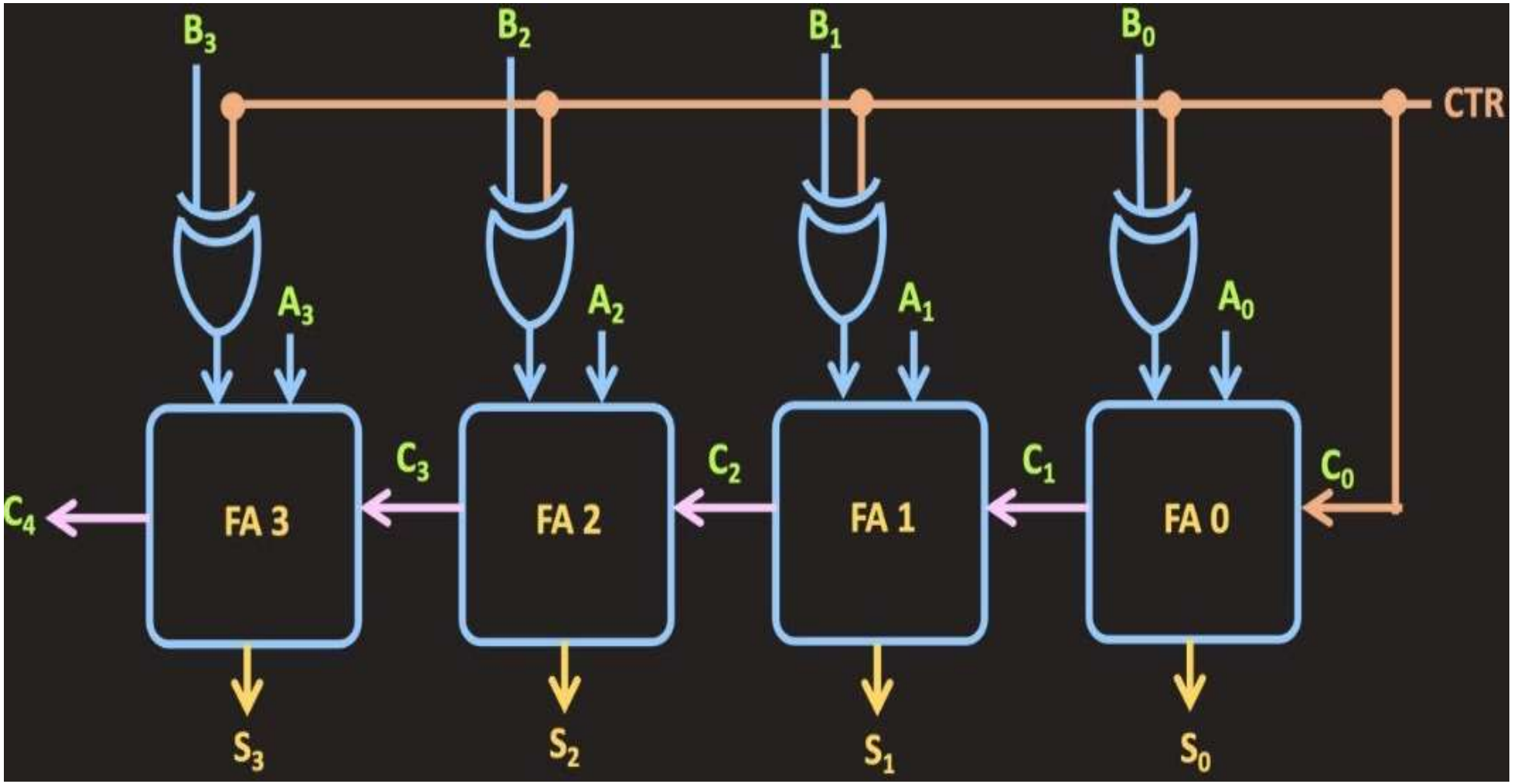
PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).1: LOGICAL CIRCUIT DIAGRAM OF A FULL ADDER



PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).2: block diagram of 4-bit full adder/subtractor



PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

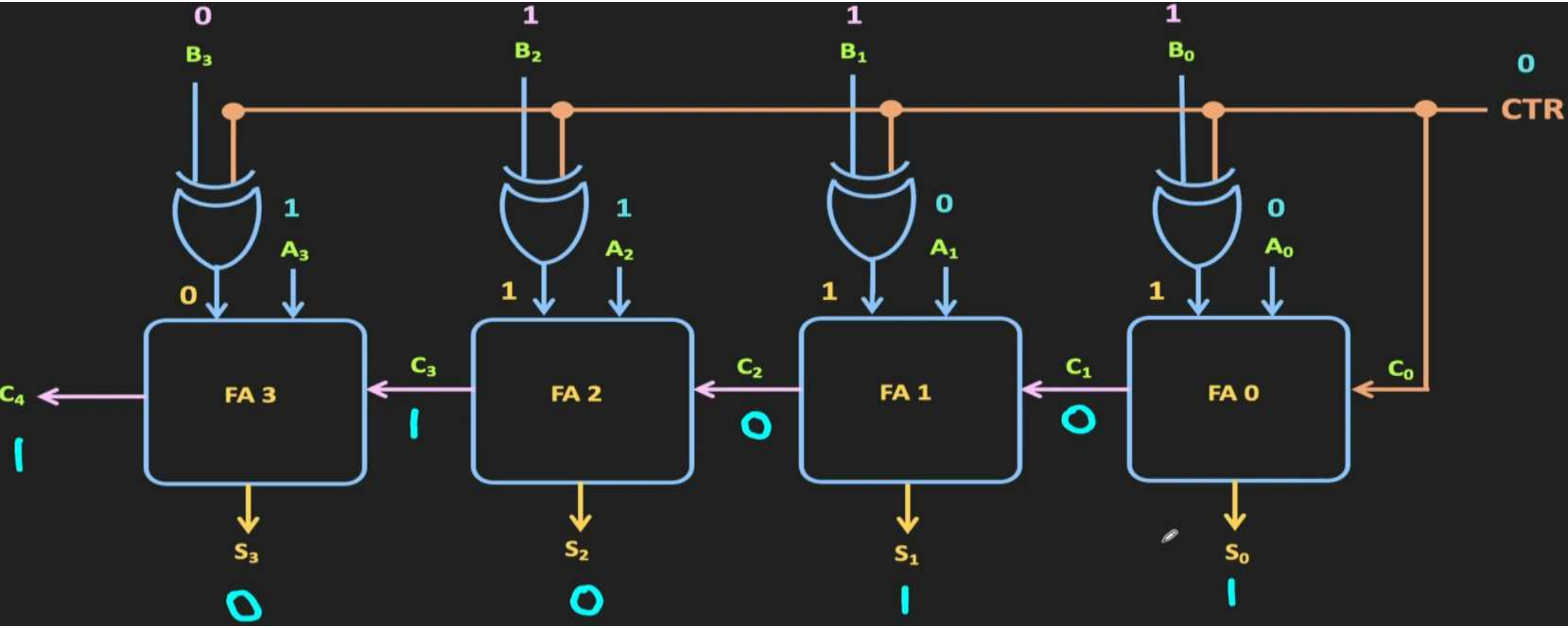
Diagram No –(2.1).3: addition E.g., for 4-bit full adder/subtractor

consider an E.g., of 4-bit full addition as follows:

```
  12 →   1 1 0 0
+  7 →   0 1 1 1
_____________________
  19 → 1 0 0 1 1
```

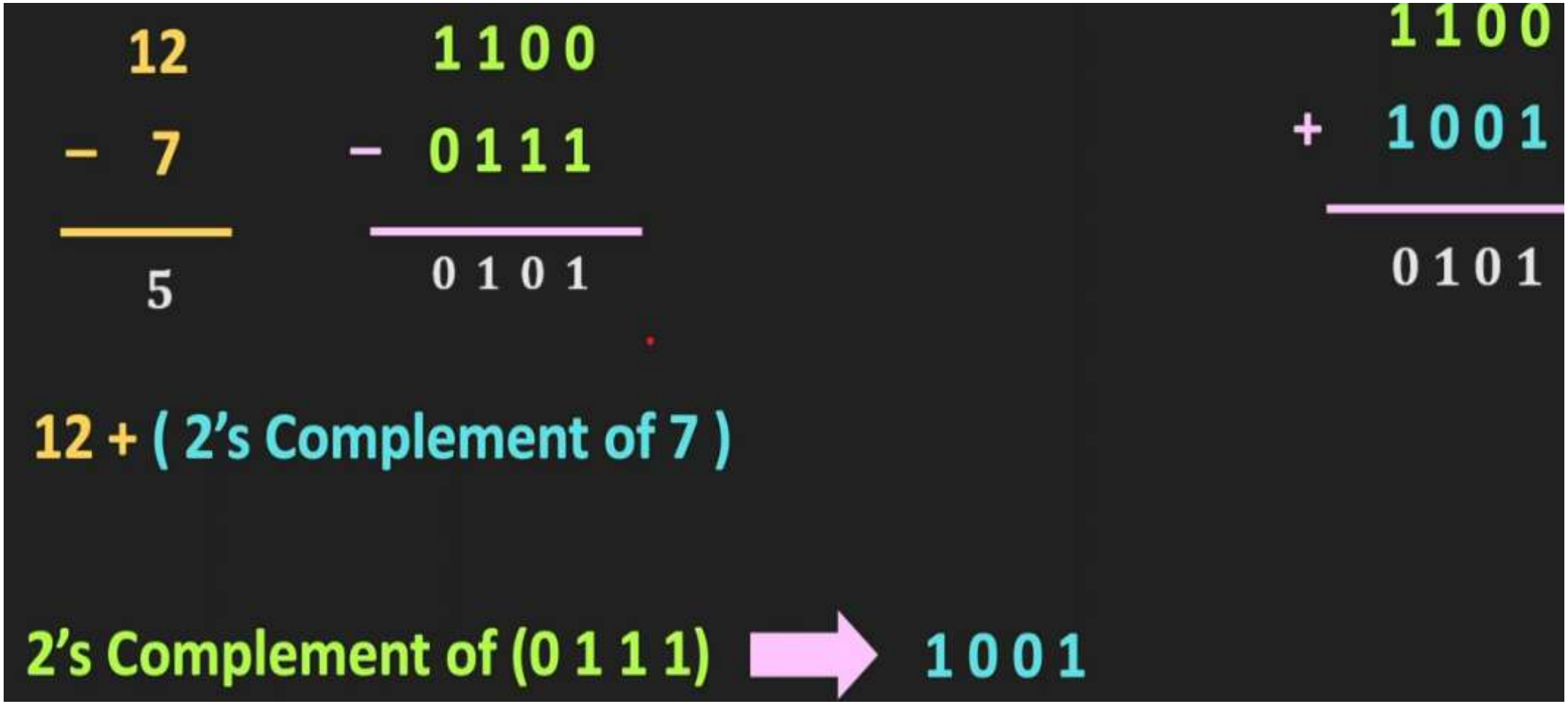
PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).4: block diagram of 4-bit full adder/subtractor for the addition E.g.,



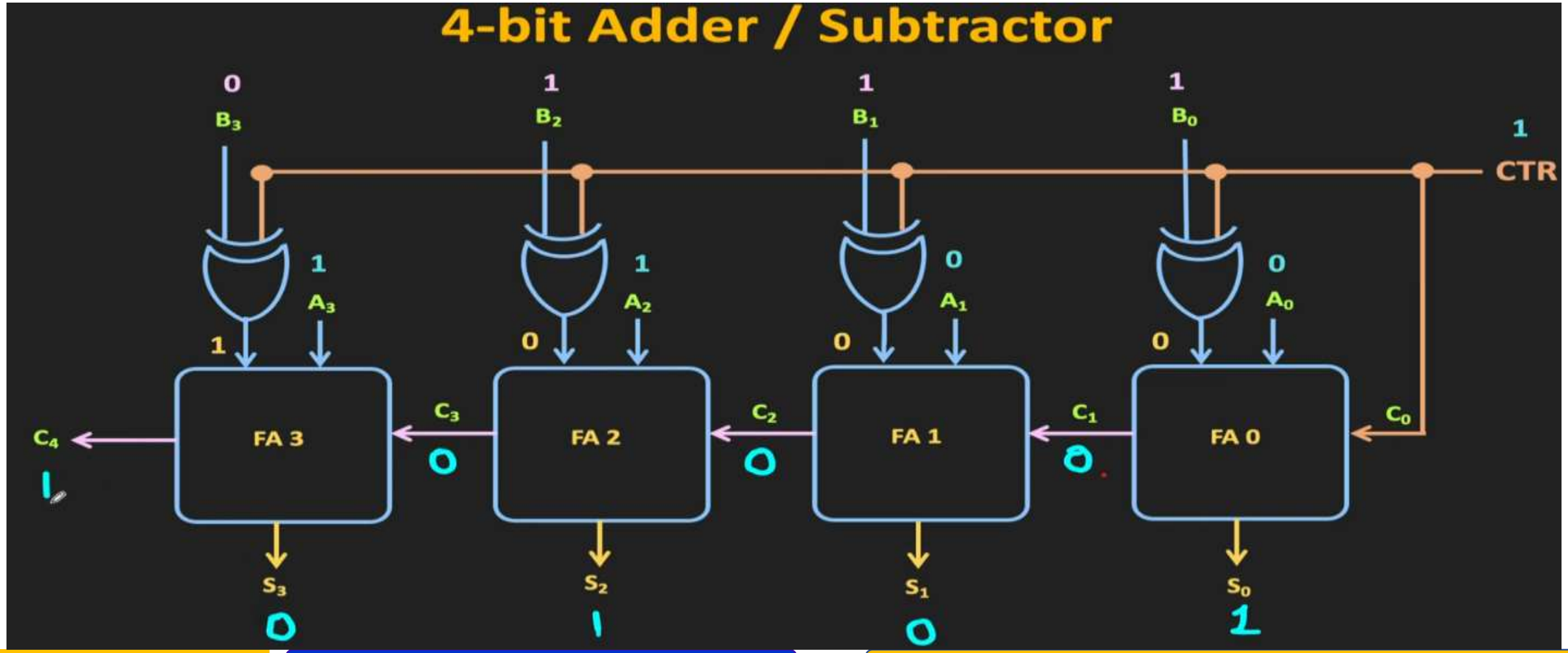
PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).5: addition E.g., for 4-bit full adder/subtractor



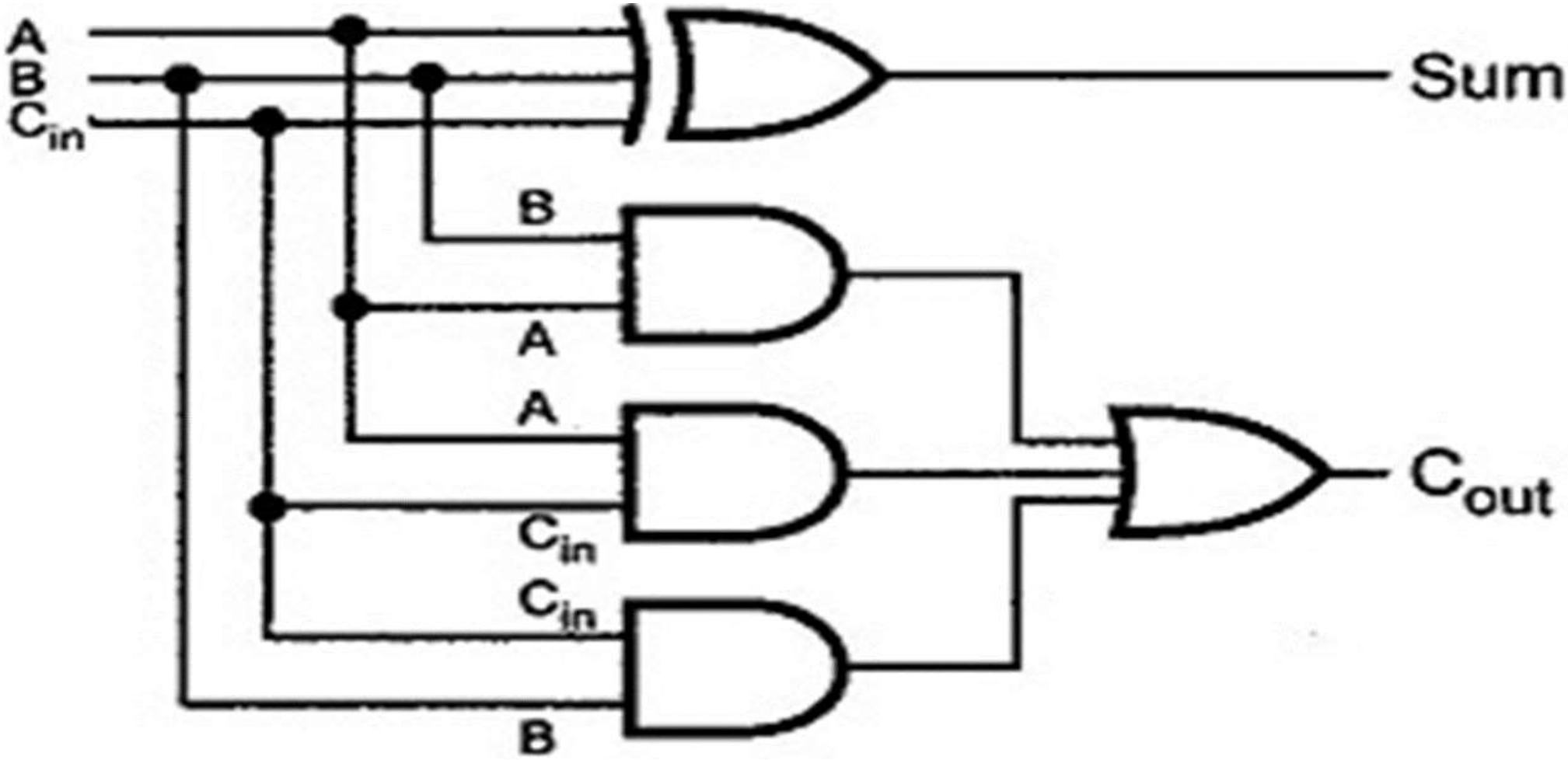
PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).6: block diagram of 4-bit full adder/subtractor for the subtraction E.g.,



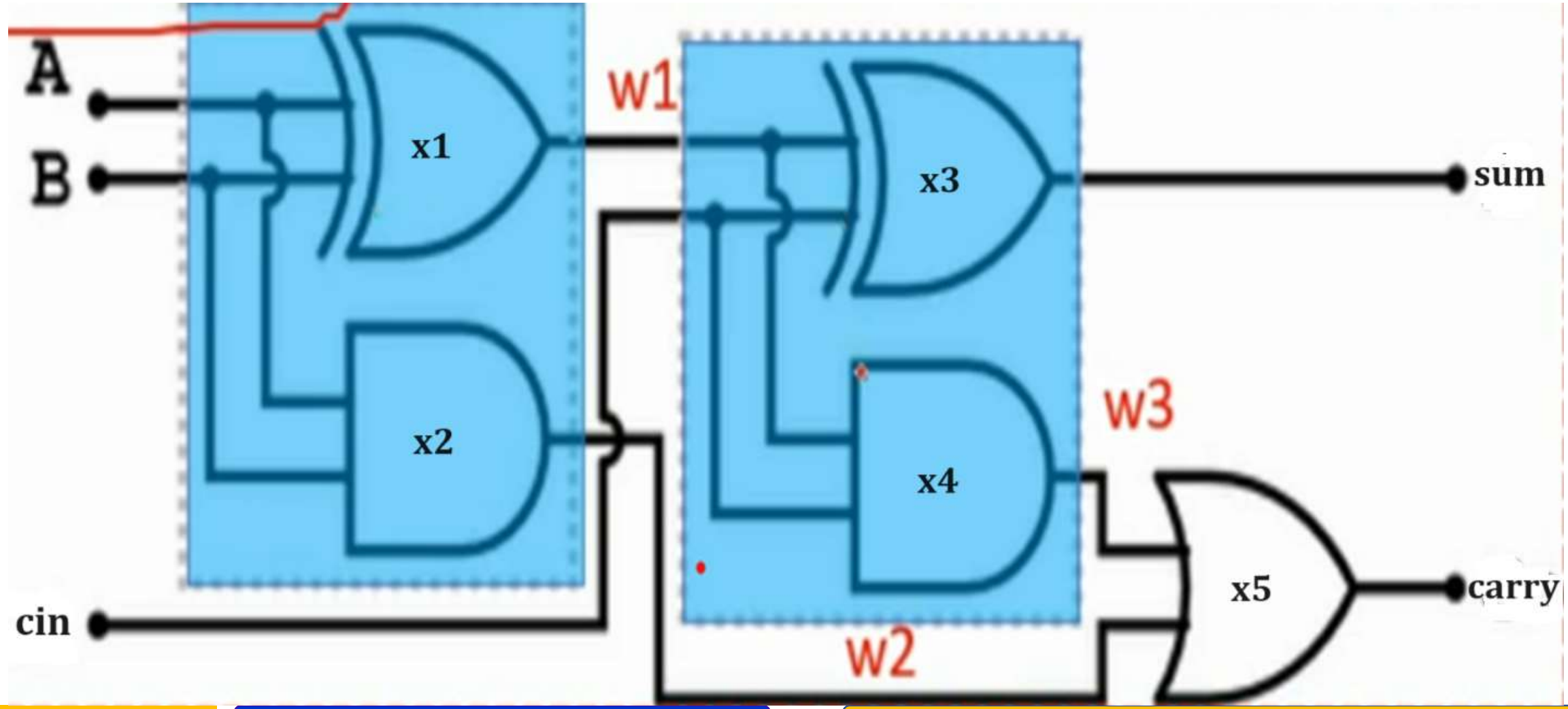
PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).7: LOGICAL CIRCUIT DIAGRAM OF A FULL ADDER



PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).8: full adder logic circuit diagram using 2-half adders



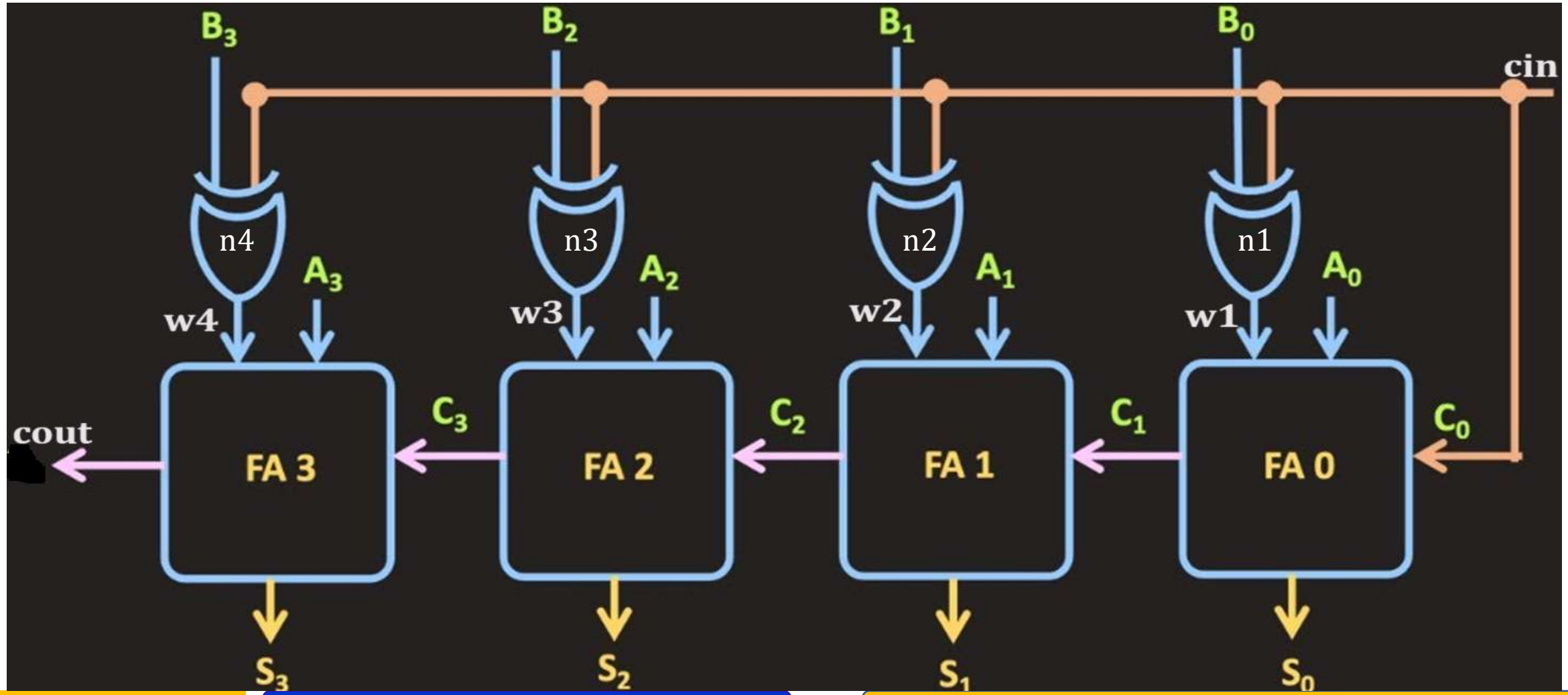
PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Full adder circuit description Verilog Program example for implementing full adder/subtractor

```
module fulladder(sum,carry,a,b,cin);
        output sum,carry;
        input a,b,cin;
        wire w1,w2,w3;
        xor x1(w1,a,b);
        and x2(w2,a,b);
        xor x3(sum,w1,cin);
        and x4(w3,w1,cin);
        or x5(carry,w2,w3);
endmodule
```

PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).2: block diagram of 4-bit full adder/subtractor



PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Full adder/subtractor circuit description Verilog Program example for implementing full adder/subtractor

```
module binary_adder_subtractor(sum,cout,a,b,cin);
        input [3:0]a,b;
        input cin;
        output [3:0]sum;
        output cout;
        wire c0,c1,c2,w1,w2,w3,w4;

        xnor n1(w1,b[0],cin);
        xnor n2(w2,b[1],cin);
        xnor n3(w3,b[2],cin);
        xnor n4(w4,b[3],cin);
```

Full adder/subtractor circuit description Verilog Program example for implementing full adder/subtractor

```
        fulladder FA0(sum[0],c0,a[0],w1,cin);
        fulladder FA1(sum[1],c1,a[1],w2,c0);
        fulladder FA2(sum[2],c2,a[2],w3,c1);
        fulladder FA3(sum[3],cout,a[3],w4,c2);
endmodule
```

PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Test bench waveform Verilog Program example for implementing full adder/subtractor

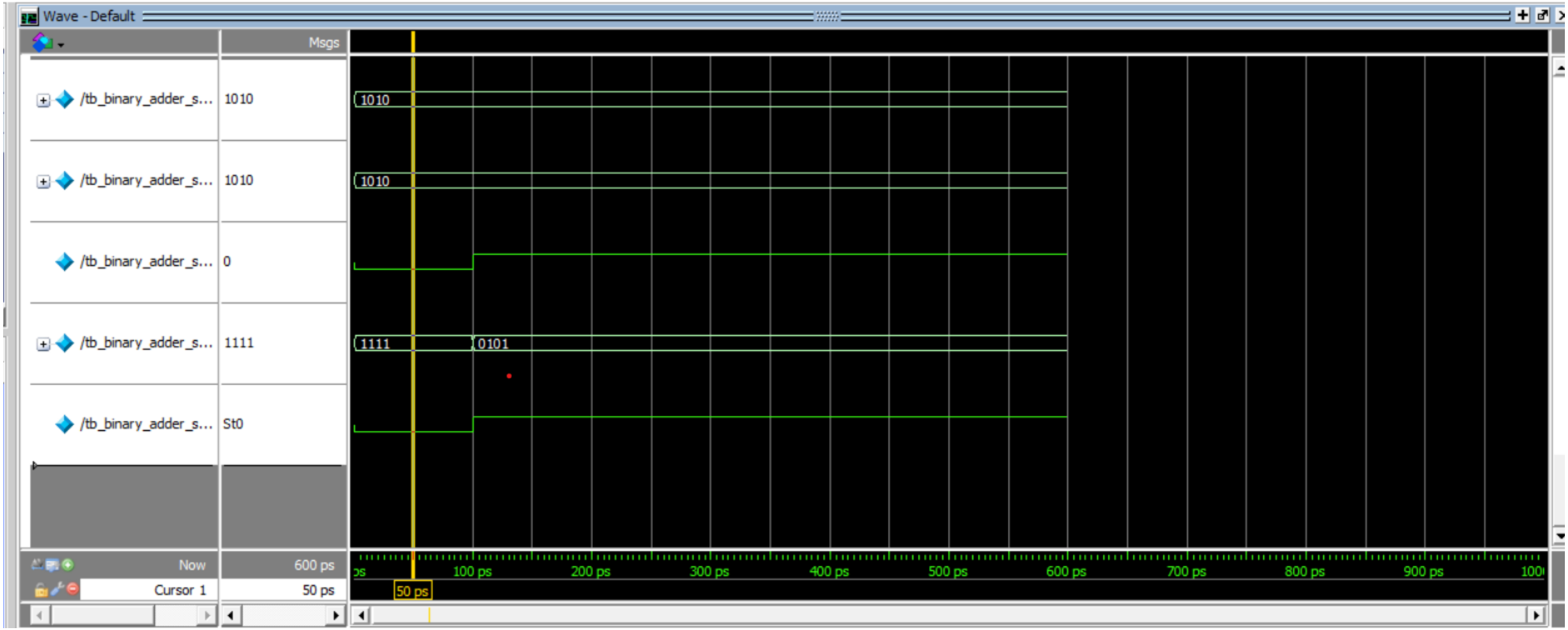
```
module tb_binary_adder_subtractor( );
        reg [3:0] a, b;
        reg cin;
        wire [3:0] sum;
        wire cout;
        binary_adder_subtractor  bas(sum,cout,a,b,cin);
        initial
                begin
                        a=4'd10;
                        b=4'd10;
                        cin=0;

                        #100
```

Test bench waveform Verilog Program example for implementing full adder/subtractor

```
                        a=4'd10;
                        b=4'd10;
                        cin=1;
                end
endmodule
```

PROGRAM-2: DESIGN OF VERILOG CODED FOR IMPLEMENTATION OF 4-BIT FULL ADDER/SUBTRACTOR

Diagram No –(2.1).1: output screen shot of Verilog code for 4-bit full adder/subtractor



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 5, 2024

DIGITAL DESIGN & COMPUTER ORGANISATION LAB

(BCS302)

PROGRAM-3

DESIGN VERILOG HDL TO IMPLEMENT SIMPLE CIRCUITS USING STRUCTURAL, DATA FLOW AND BEHAVIORAL MODEL. [E.G., FULL ADDER]

DIGITAL DESIGN & COMPUTER ORGANISATION LAB

(BCS302)

PROGRAM-3.1

DESIGN VERILOG HDL TO IMPLEMENT SIMPLE CIRCUITS USING STRUCTURAL MODEL. [E.G FULL ADDER]

PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Diagram No –(3.1).1: block diagram of full adder

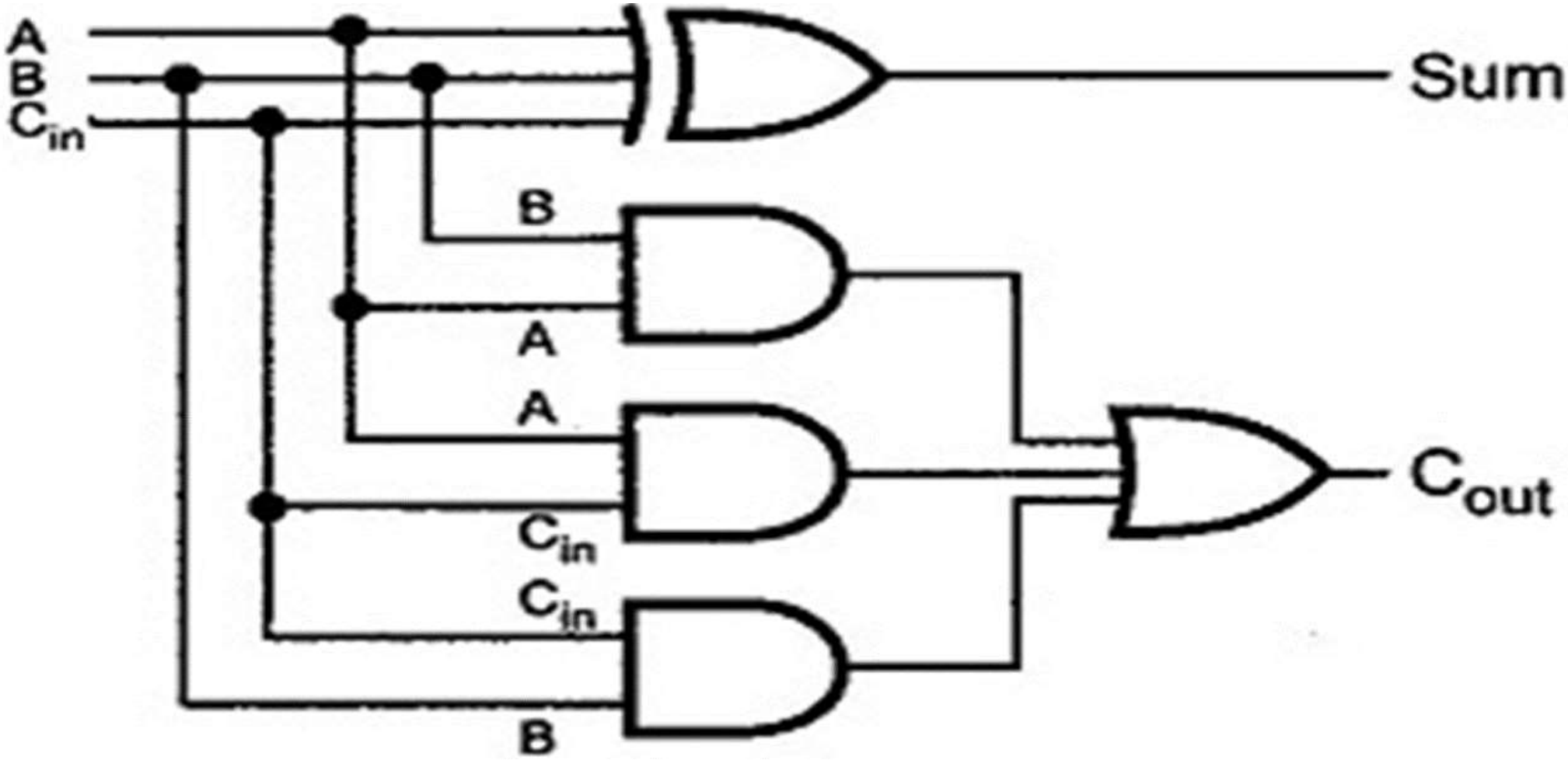
PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Diagram No –(3.1).2: truth table of the full adder

Decimal number	A	B	Cin	SUM	CARRY
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

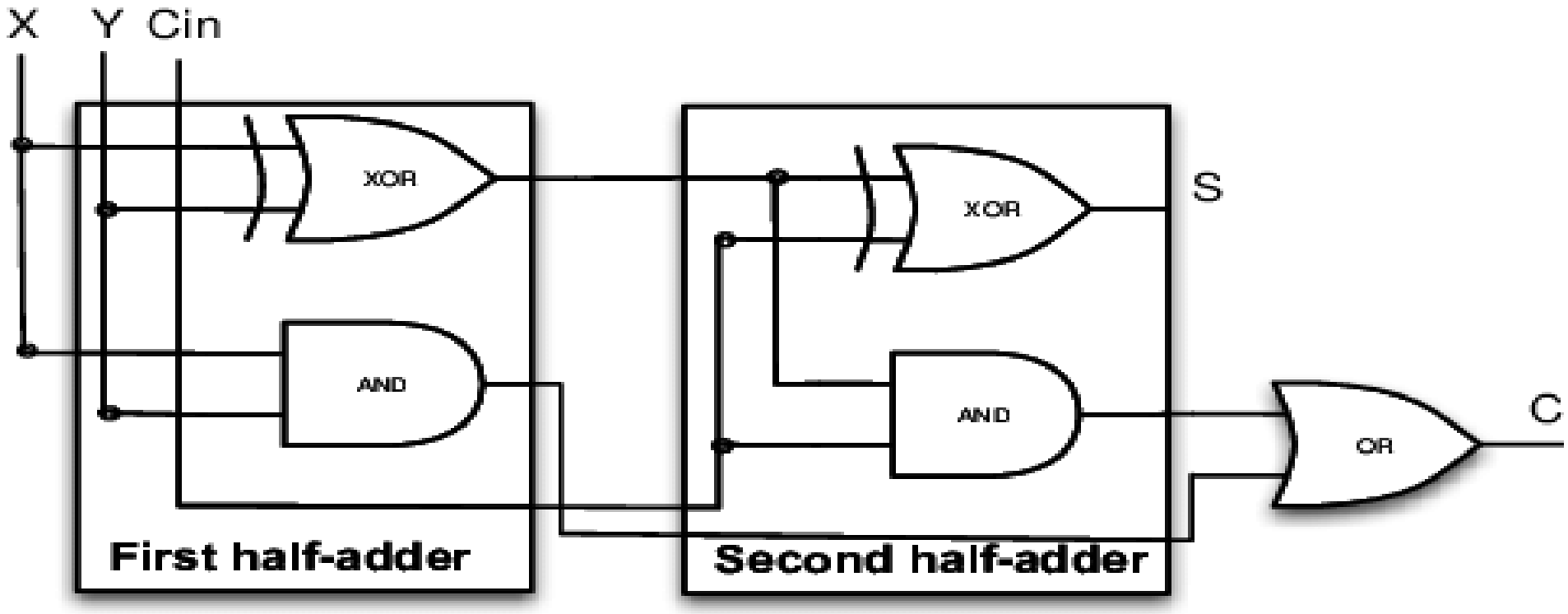
PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Diagram No –(3.1).3: full adder circuit using logic gates



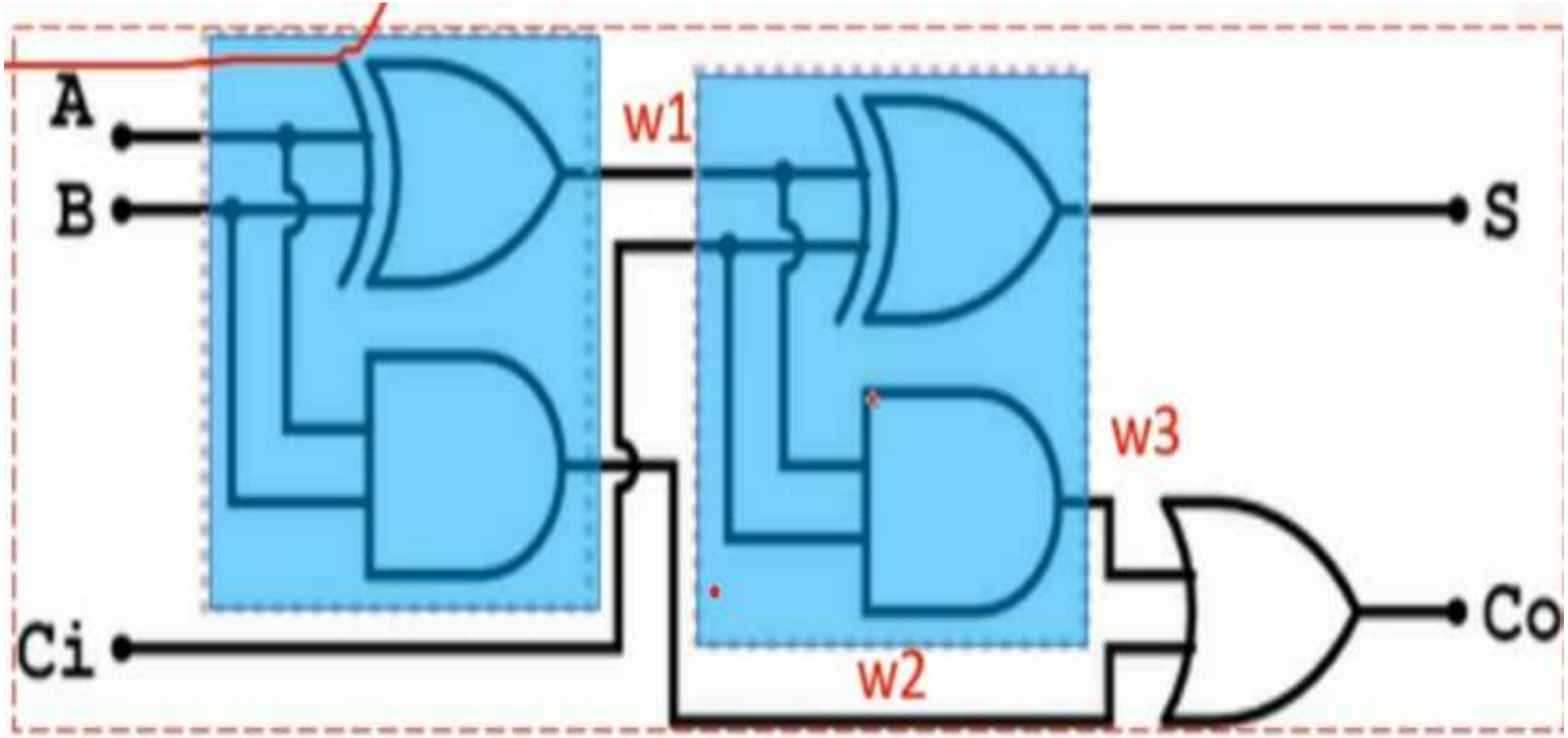
PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Diagram No –(3.1).4: Full adder logical circuit diagram using 2-half adder blocks



PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Diagram No –(3.1).5: Full adder logical circuit diagram using 2-half adder blocks



PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Diagram No –(3.1).6: Output Expressions of full adder circuit using logic gates

EXPRESSIONS FOR THE FULL ADDER USING LOGIC GATES:

SUM = A ^ B ^ Cin

CARRY = (A&B) | (B&C) | (A&C)

PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Circuit description module Program example for full adder simulation

```
module myhalfadder(A,B,S,C);
  input A,B;
  output S,C;
        xor(S,A,B);
        and(C,A,B);
endmodule
```

//////////////

Circuit description module Program example for full adder simulation

```
module FullAdder(S,Co,A,B,Ci);
   input A,B,Ci;
   output S,Co;
   wire w1,w2,w3;
   myhalfadder  h1(A,B,w1,w2);
   myhalfadder  h2(w1,Ci,S,w3);
   or(Co,w2,w3);
endmodule
```

PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Test bench waveform module Program example for full adder simulation

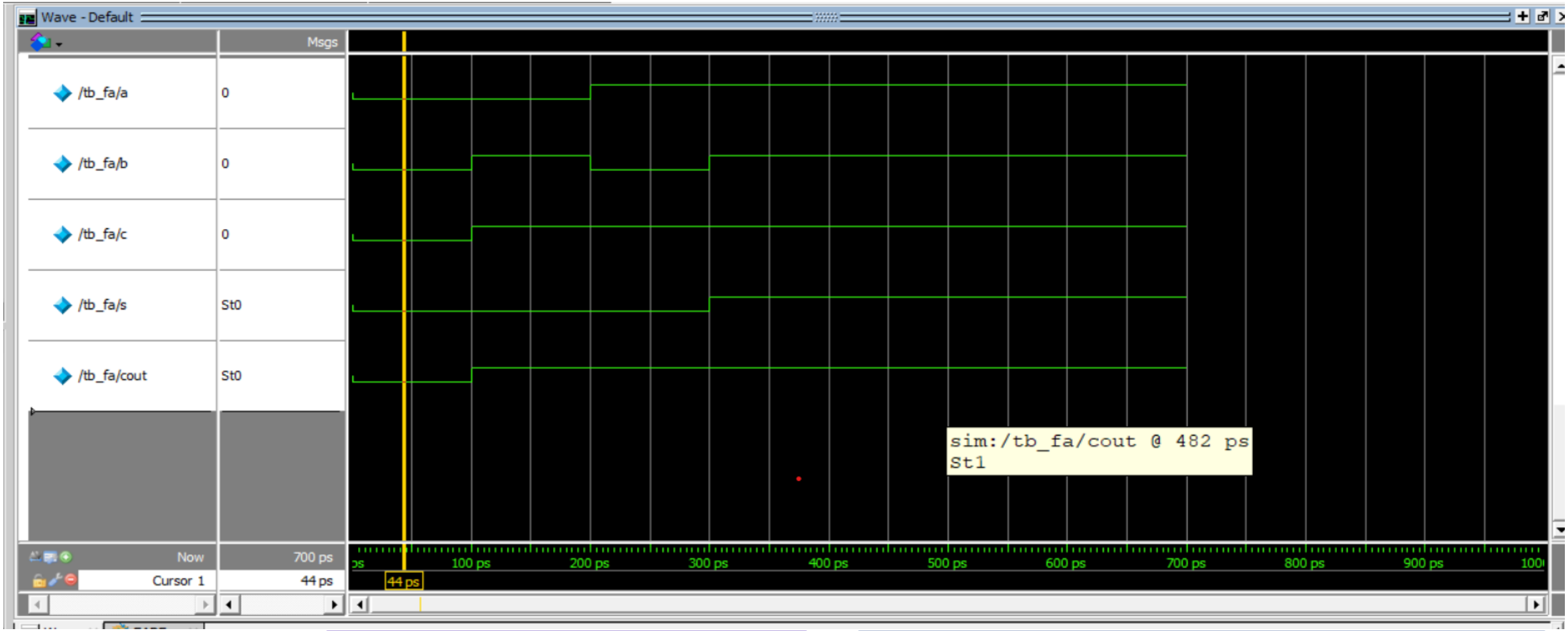
```
module tb_fa( );
  reg a,b,c_i;
  wire s,c_o;

  FullAdder f1(s,c_o,a,b,c_i);
  initial
        begin
              a = 1'b0;  b=1'b0; c_i =1'b0;
              #100
              a = 1'b0;  b=1'b1; c_i =1'b1;
              #100
```

Test bench waveform module Program example for full adder simulation

```
              a = 1'b1;  b=1'b0; c_i =1'b1;
              #100
              a = 1'b1;  b=1'b1; c_i =1'b0;

        end
endmodule
```

PROGRAM-3.1: DESIGN OF FULL ADDER USING STRUCTURAL MODEL

Diagram No –(3.1).7: Output screen shot of Verilog code for full adder circuit using logic gates



DIGITAL DESIGN & COMPUTER ORGANISATION LAB
(BCS302)

PROGRAM-3.2

DESIGN VERILOG HDL TO IMPLEMENT SIMPLE CIRCUITS USING DATA FLOW MODEL. [E.G., FULL ADDER]

PROGRAM-3.2: DESIGN OF FULL ADDER USING DATA FLOW MODEL

Diagram No –(3.2).1: Block diagram of full adder

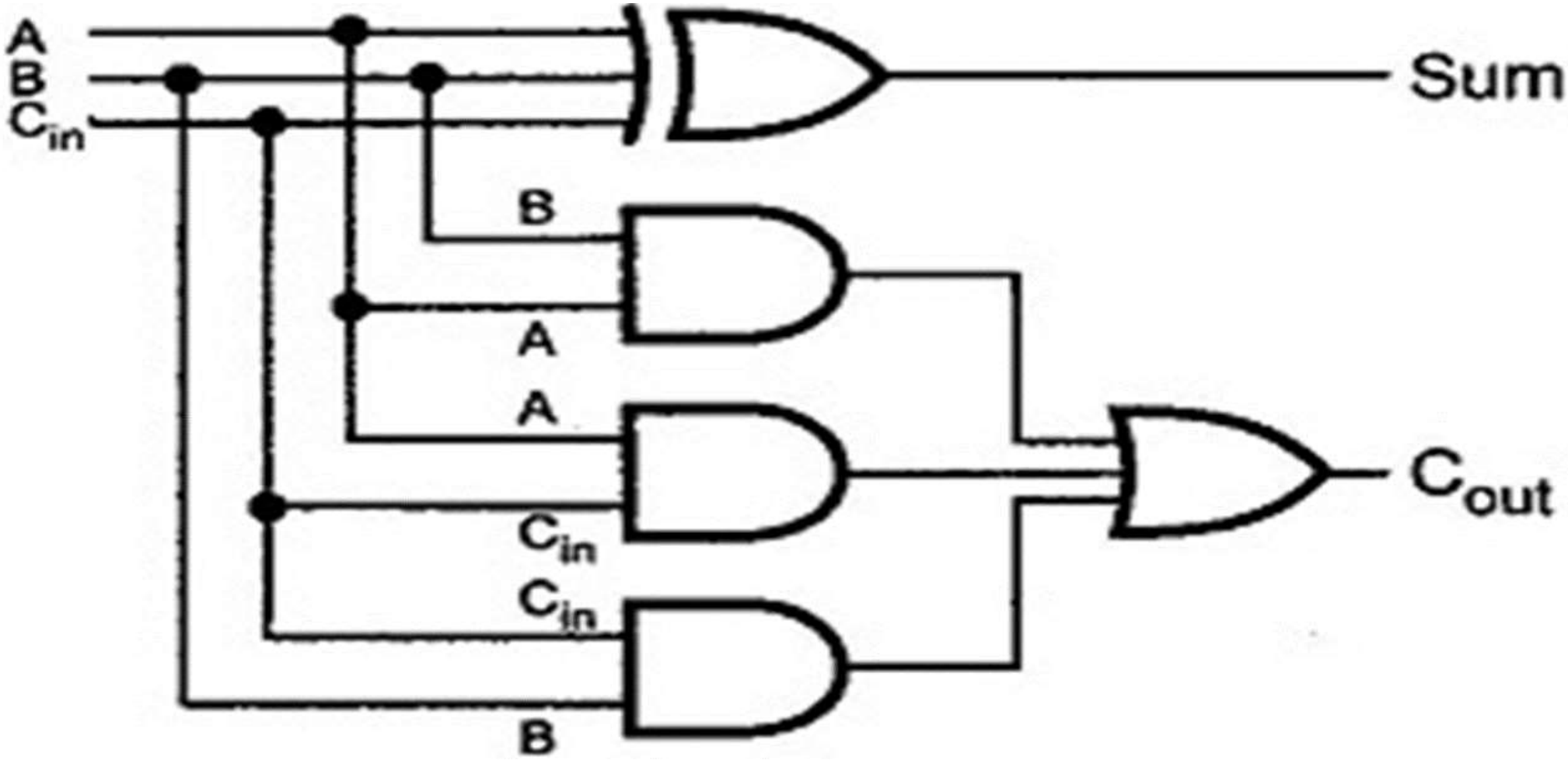
PROGRAM-3.2: DESIGN OF FULL ADDER USING DATA FLOW MODEL

Diagram No –(3.2).2: full adder truth table

Decimal number	A	B	Cin	SUM	CARRY
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

PROGRAM-3.2: DESIGN OF FULL ADDER USING DATA FLOW MODEL

Diagram No –(3.2).3: full adder circuit using logic gates



PROGRAM-3.2: DESIGN OF FULL ADDER USING DATA FLOW MODEL

Diagram No –(3.2).4: Output Expressions of full adder circuit using logic gates

EXPRESSIONS FOR THE FULL ADDER USING LOGIC GATES:

SUM = A ^ B ^ Cin

CARRY = (A&B) | (B&C) | (A&C)

PROGRAM-3.2: DESIGN OF FULL ADDER USING DATA FLOW MODEL

Circuit description module of Verilog Program for implementing full adder using dataflow model

```
module fulladder ( a,b,c,s,cout)
        input a, b,c;
        output s, cout;
        assign s= a ^ b^c;
        assign cout= (a&b) | (b&c) | (a&c);
 endmodule
```

PROGRAM-3.2: DESIGN OF FULL ADDER USING DATA FLOW MODEL

Test bench module of Verilog Program for implementing full adder using data flow model

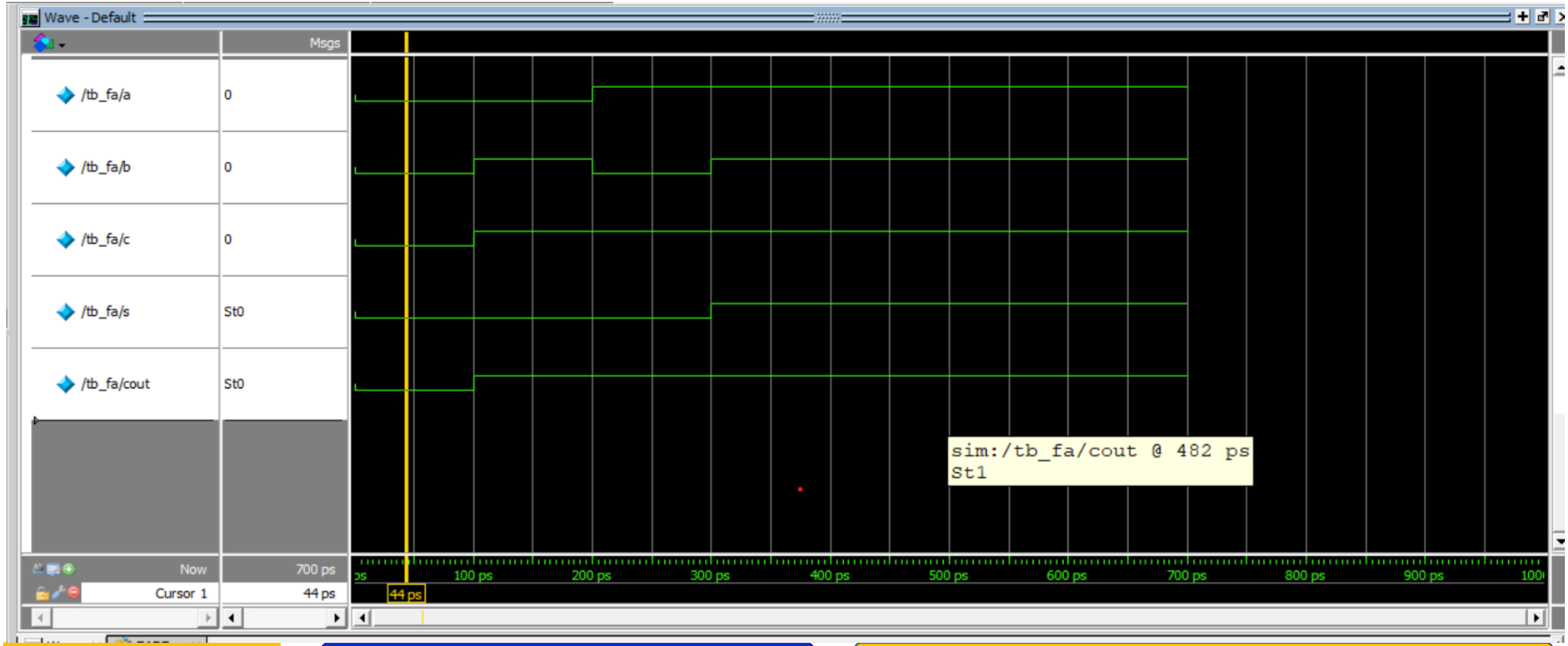
```
module tb_fa( );
        reg a,b,c;
        wire s,cout;

        fulladder f1(a,b,c,s,cout);

        initial
                begin
                        a = 1'b0;  b=1'b0; c=1'b0;
                        #100
                        a = 1'b0;  b=1'b1; c=1'b1;
```

Test bench module of Verilog Program for implementing full adder using data flow model

```
                        #100
                        a = 1'b1;  b=1'b0; c=1'b1;
                        #100
                        a = 1'b1;  b=1'b1; c=1'b1;

                end
endmodule
```

PROGRAM-3.2: DESIGN OF FULL ADDER USING DATA FLOW MODEL

Diagram No –(3.2).5: Output screen shot of Verilog code for full adder circuit using logic gates



DIGITAL DESIGN & COMPUTER ORGANISATION LAB

(BCS302)

PROGRAM-3.3

DESIGN VERILOG HDL CODE TO IMPLEMENT SIMPLE CIRCUITS USING BEHAVIOURAL MODEL.

PROGRAM-3.3: DESIGN OF FULL ADDER USING BEHAVIOURAL MODEL

Diagram No –(3.3).1: Block diagram of full adder

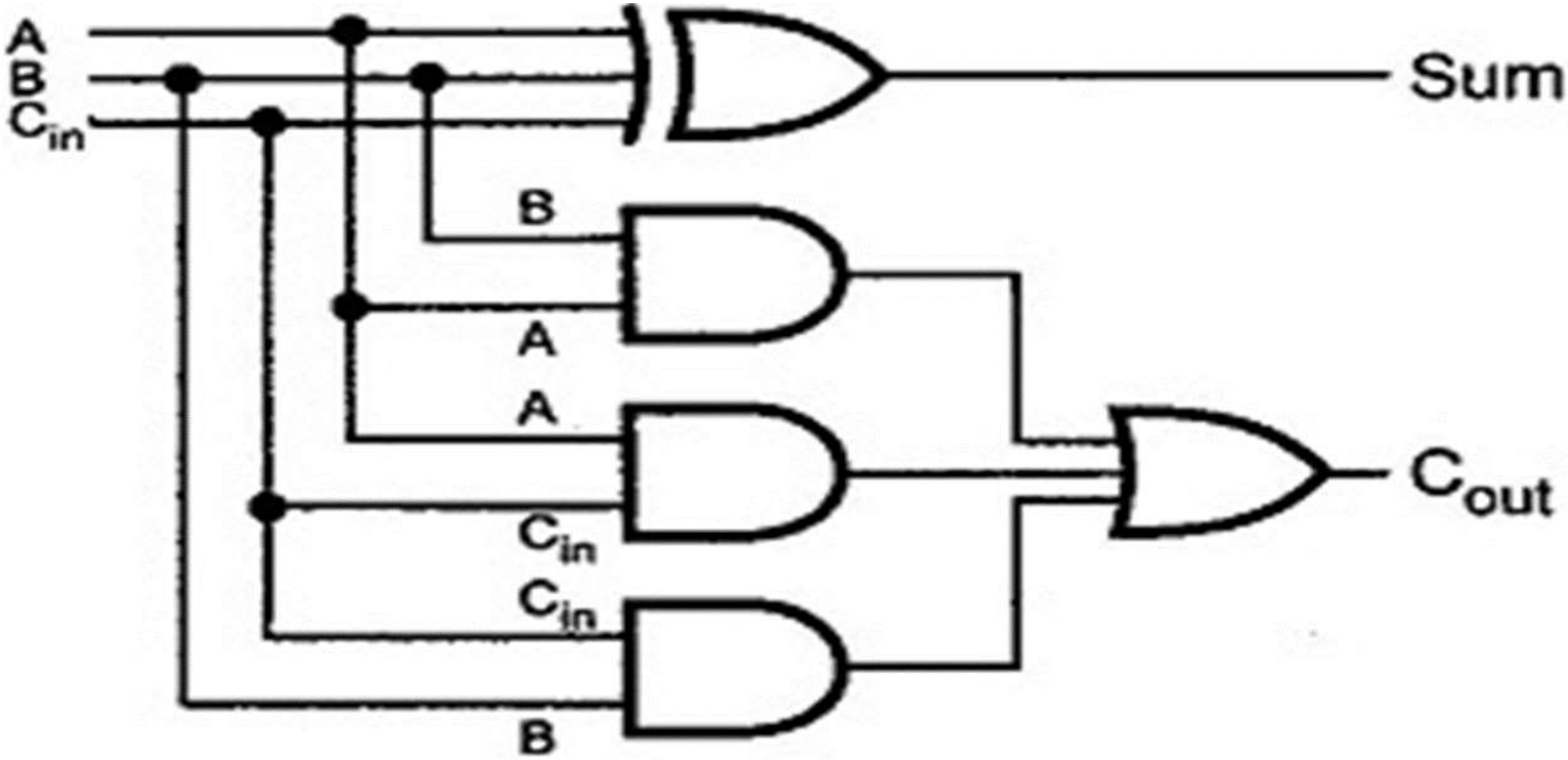
PROGRAM-3.3: DESIGN OF FULL ADDER USING BEHAVIOURAL MODEL

Diagram No –(3.3).2: truth table of full adder

Decimal number	A	B	Cin	SUM	CARRY
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

PROGRAM-3.3: DESIGN OF FULL ADDER USING BEHAVIOURAL MODEL

Diagram No –(3.3).3: full adder circuit using logic gates



PROGRAM-3.3: DESIGN OF FULL ADDER USING BEHAVIOURAL MODEL

Diagram No –(3.3).4: Output Expressions of full adder circuit using logic gates

EXPRESSIONS FOR THE FULL ADDER USING LOGIC GATES:

SUM = A ^ B ^ Cin

CARRY = (A&B) | (B&C) | (A&C)

PROGRAM-3.3: DESIGN OF FULL ADDER USING BEHAVIOURAL MODEL

Circuit description module of Verilog Program for implementing full adder using behavioural model

```
fulladd(cin,a,b,s,co);
        input cin,a,b;
         output s,co;
        reg s,co;
        always@(cin or a or b)
                begin
                        case ({cin,a,b})
                        3'b000:{co,s}='b00;
                        3'b001:{co,s}='b01;
```

Circuit description module of Verilog Program for implementing full adder using behavioural model

```
                        3'b010:{co,s}='b01;
                        3'b011:{co,s}='b10;
                        3'b100:{co,s}='b01;
                        3'b101:{co,s}='b10;
                        3'b110:{co,s}='b10;
                        3'b111:{co,s}='b11;
                        endcase
                end
endmodule
/////
```

PROGRAM-3.3: DESIGN OF FULL ADDER USING BEHAVIOURAL MODEL

Test bench waveform module of Verilog Program for implementing full adder using behavioural model

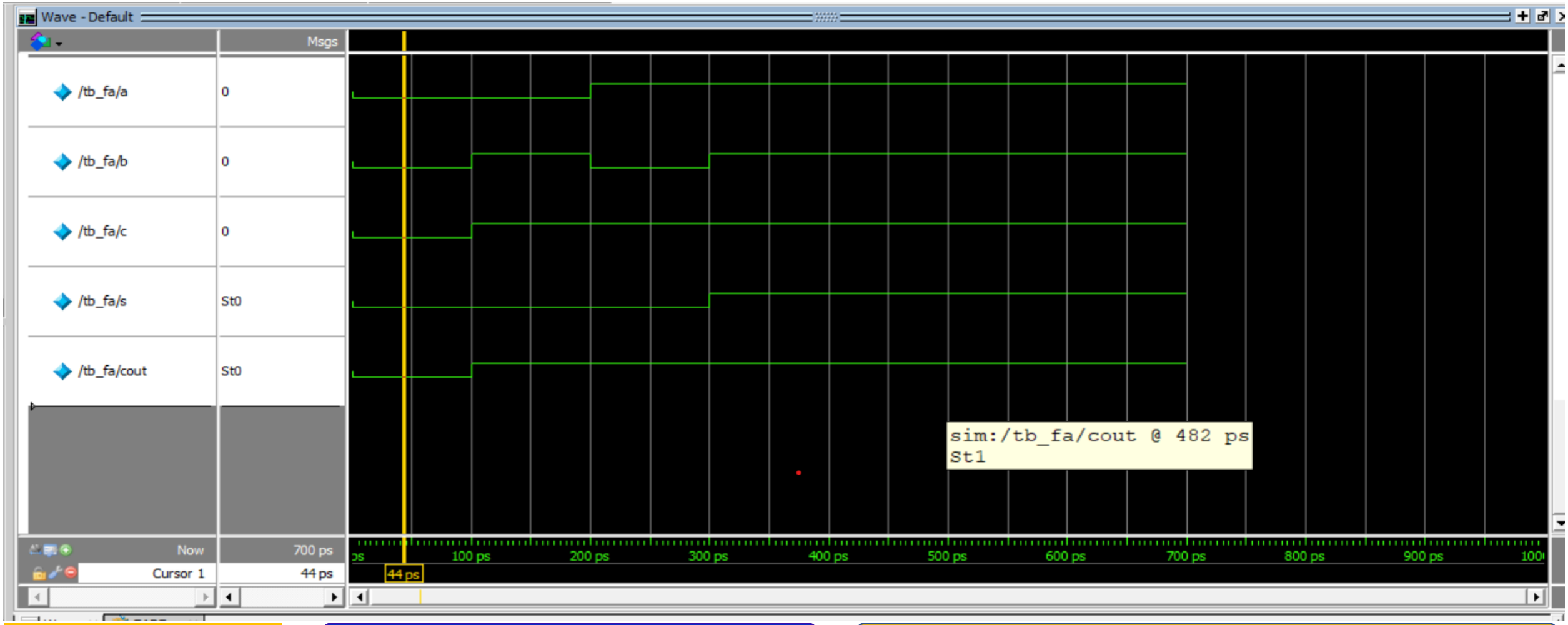
```
module tb_fa( );
        reg cin,a,b;
        wire s,co;
        fulladd f1(cin,a,b,s,co);
        initial
                begin
                        cin = 1'b0;  a=1'b0; b=1'b0;
                        #100
                        cin = 1'b0;  a=1'b1; b =1'b1;
                        #100
```

Test bench waveform module of Verilog Program for implementing full adder using behavioural model

```
                        cin = 1'b1;  a=1'b0; b =1'b1;
                        #100
                        cin = 1'b1;  a=1'b1; b =1'b0;
                        #100
                        cin = 1'b1;  a=1'b1; b =1'b1;

                end
endmodule
```

PROGRAM-3.3: DESIGN OF FULL ADDER USING BEHAVIOURAL MODEL

Diagram No –(3.3).5: full adder circuit Verilog code output screen shot



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 5, 2024

DIGITAL DESIGN & COMPUTER ORGANISATION LAB
(BCS302)

PROGRAM-4

Design Verilog HDL to implement Binary Adder-Subtractor – Half and Full Adder, Half and Full Subtractor.

DIGITAL DESIGN & COMPUTER ORGANISATION LAB
(BCS302)

PROGRAM-4.1

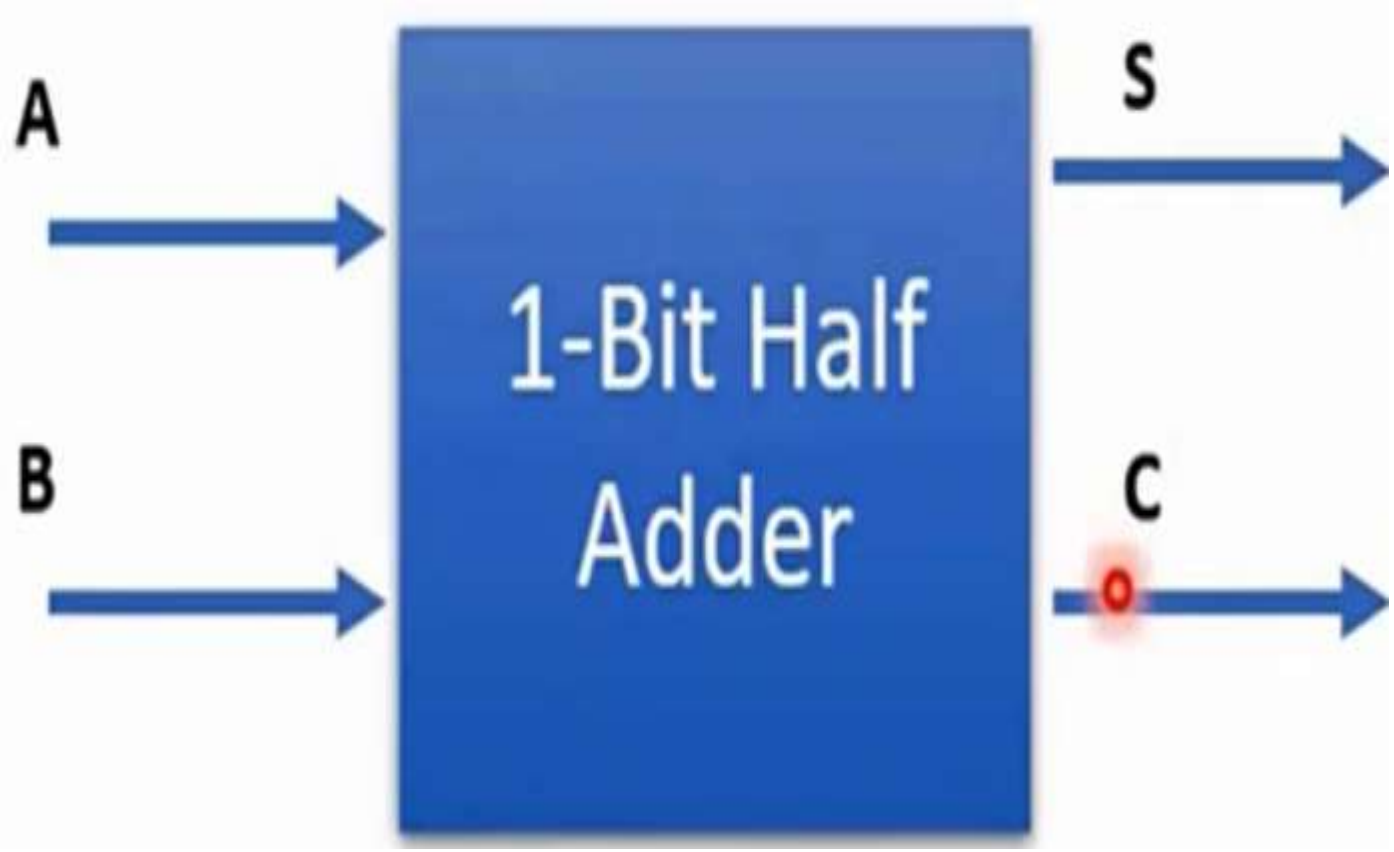
DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF HALF ADDER

Mr. PRASHANTH KUMAR S P

Department of Computer Science & Engineering

PROGRAM-4.1: DESIGN VERILOG HDL TO IMPLEMENT BINARY HALF ADDER

Diagram No –(4.1).1: half adder block diagram



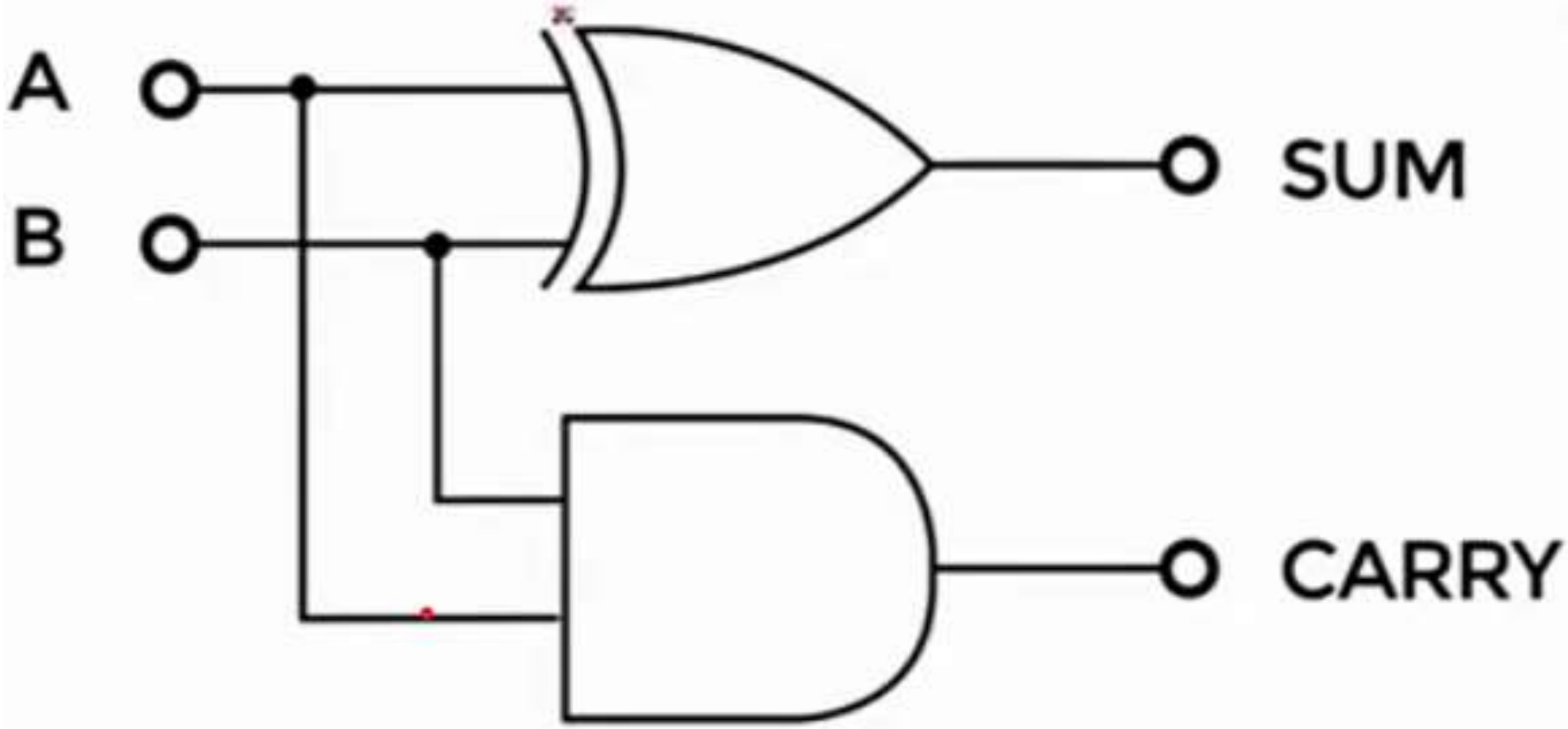
PROGRAM-4.1: DESIGN VERILOG HDL TO IMPLEMENT BINARY HALF ADDER

Diagram No –(4.1).2: Half adder truth table

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

PROGRAM-4.1: DESIGN VERILOG HDL TO IMPLEMENT BINARY HALF ADDER

Diagram No –(4.1).3: Half adder logical circuit diagram



PROGRAM-4.1: DESIGN VERILOG HDL TO IMPLEMENT BINARY HALF ADDER

Diagram No –(4.1).4: Half adder logical expressions

Logical expressions of half adder

Sum = a ^ b

Carry = a & b

PROGRAM-4.1: DESIGN VERILOG HDL TO IMPLEMENT BINARY HALF ADDER

Verilog Program example for half adder simulation

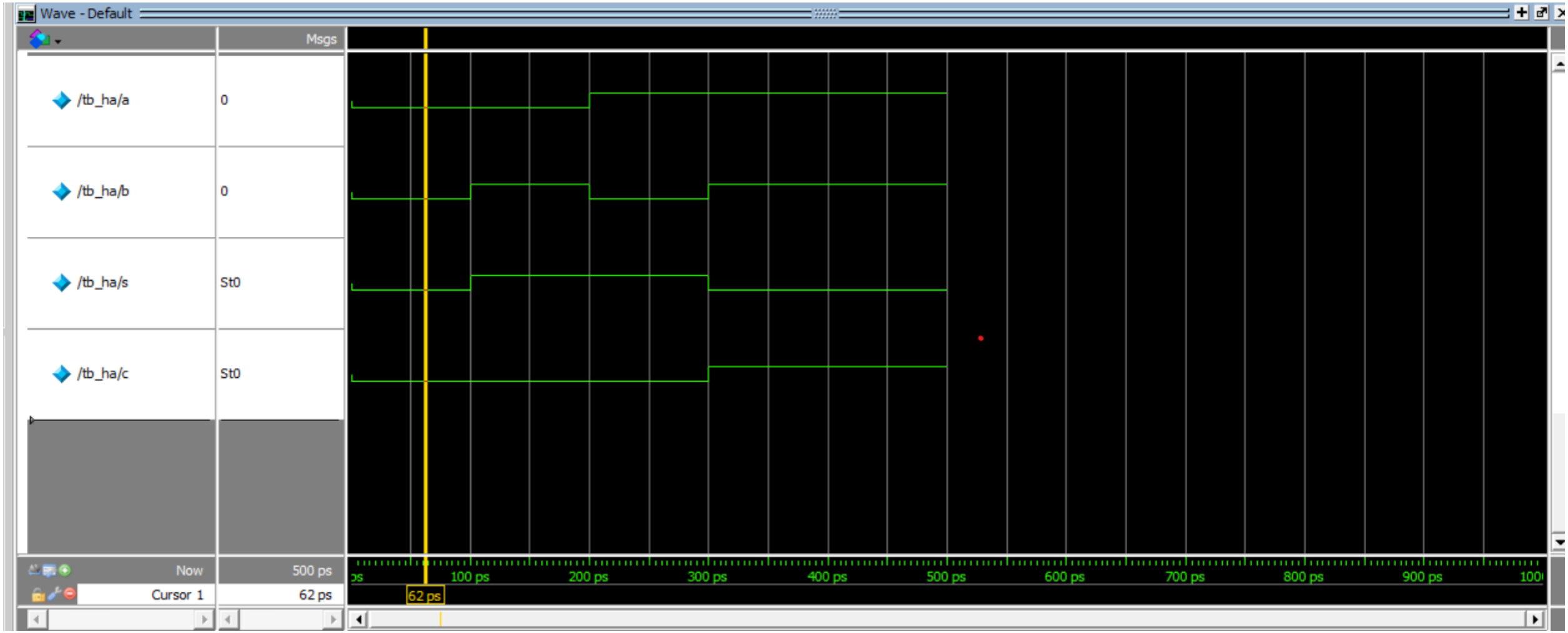
```
/*circuit description module*/
module myhalfadder(A,B,S,C);
   input A,B;
   output S,C;
            xor(S,A,B);
            and(C,A,B);
endmodule

////////////

/*test bench waveform*/
module tb_ha( );
   reg a,b;
   wire s,c;
   myhalfadder a1(a,b,s,c);
```

Verilog Program example for half adder simulation

```
initial
        begin
                a = 1'b0;  b=1'b0;
                #100
                a = 1'b0;  b=1'b1;
                #100
                a = 1'b1;  b=1'b0;
                #100
                a = 1'b1;  b=1'b1;

        end
endmodule
```

PROGRAM-4.1: DESIGN VERILOG HDL TO IMPLEMENT BINARY HALF ADDER

Diagram No –(4.1).6: half adder Verilog code simulation OUTPUT screen shot



DIGITAL DESIGN & COMPUTER ORGANISATION LAB
(BCS302)

PROGRAM-4.2

DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER CIRCUIT

Mr. PRASHANTH KUMAR S P

Department of Computer Science & Engineering

PROGRAM-4.2: DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER

Diagram No –(4.2).1: Block diagram of Full Adder

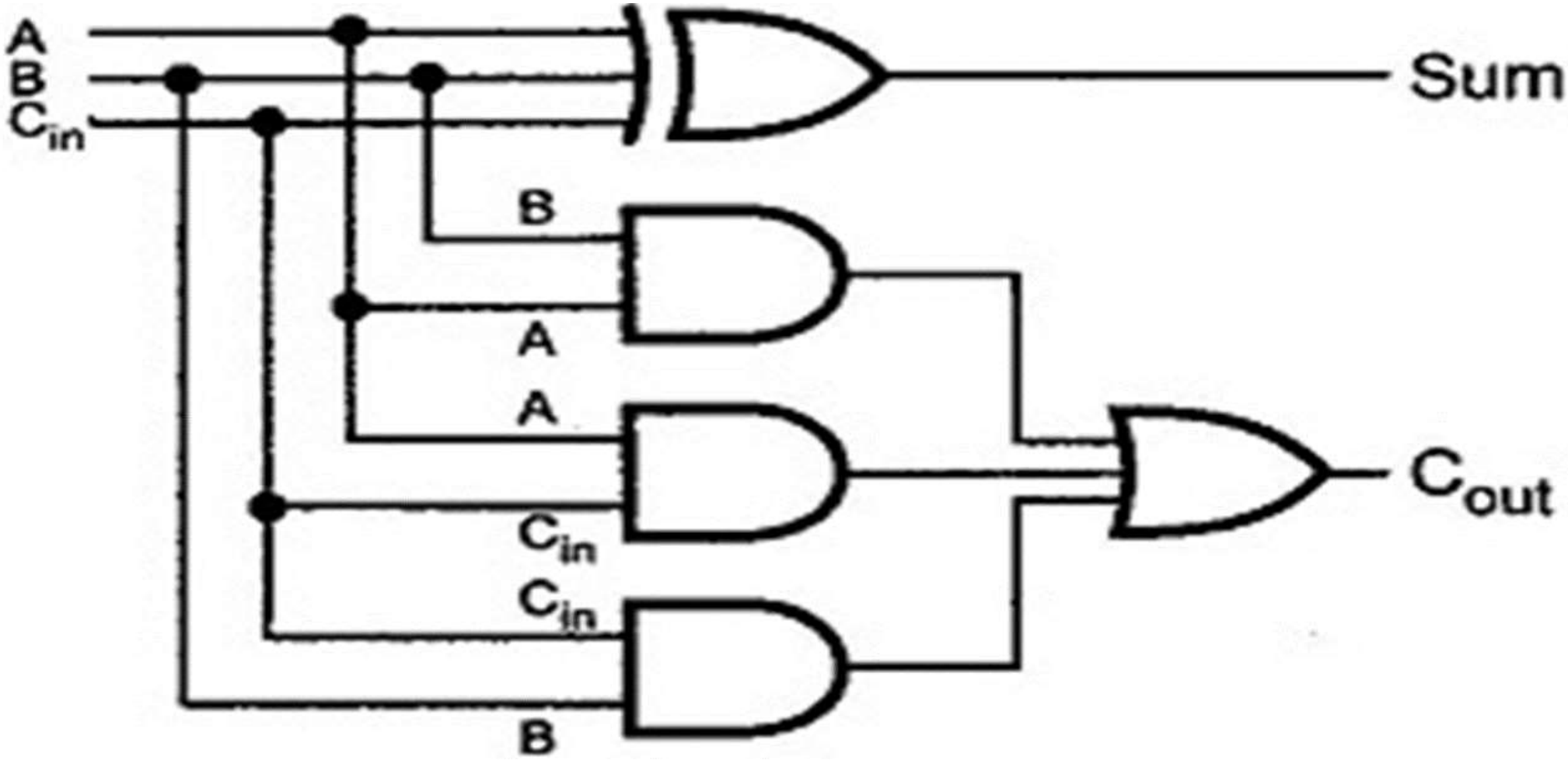
PROGRAM-4.2: DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER

Diagram No –(4.2).2: Truth table of full adder circuit

Decimal number	A	B	Cin	SUM	CARRY
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

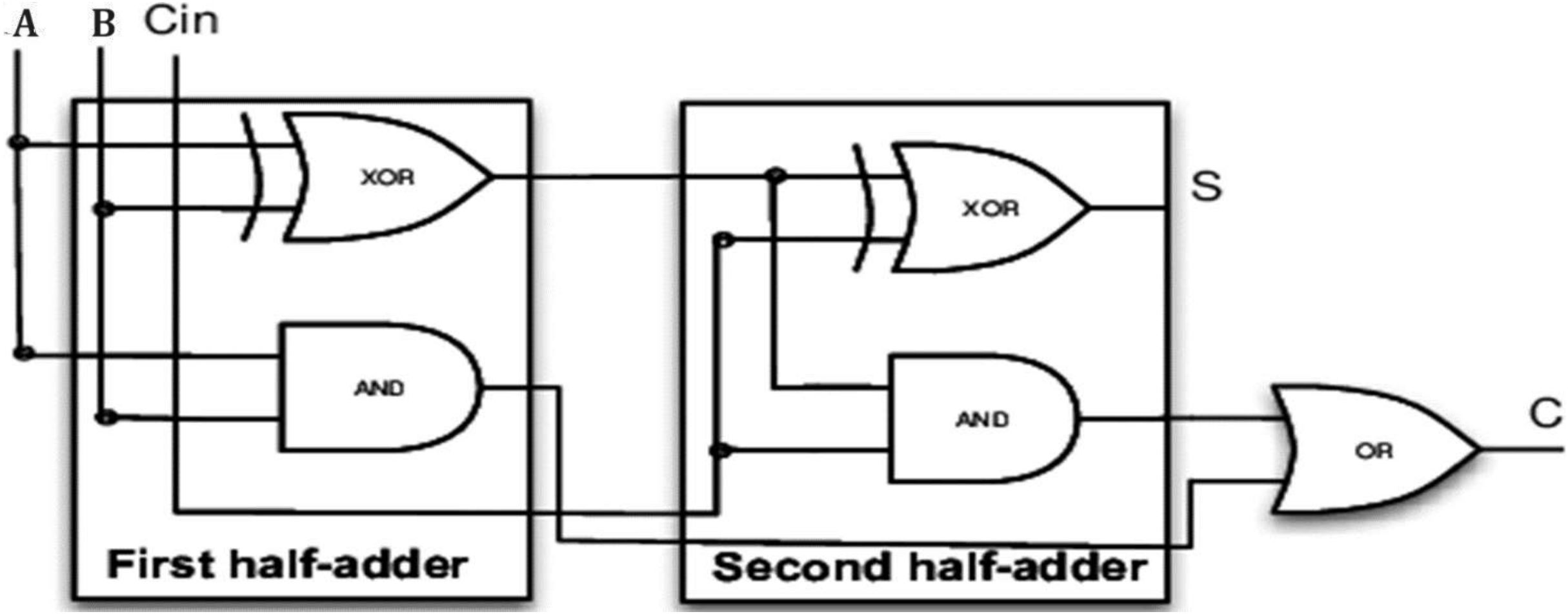
PROGRAM-4.2: DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER

Diagram No –(4.2).3: full adder circuit using logic gates



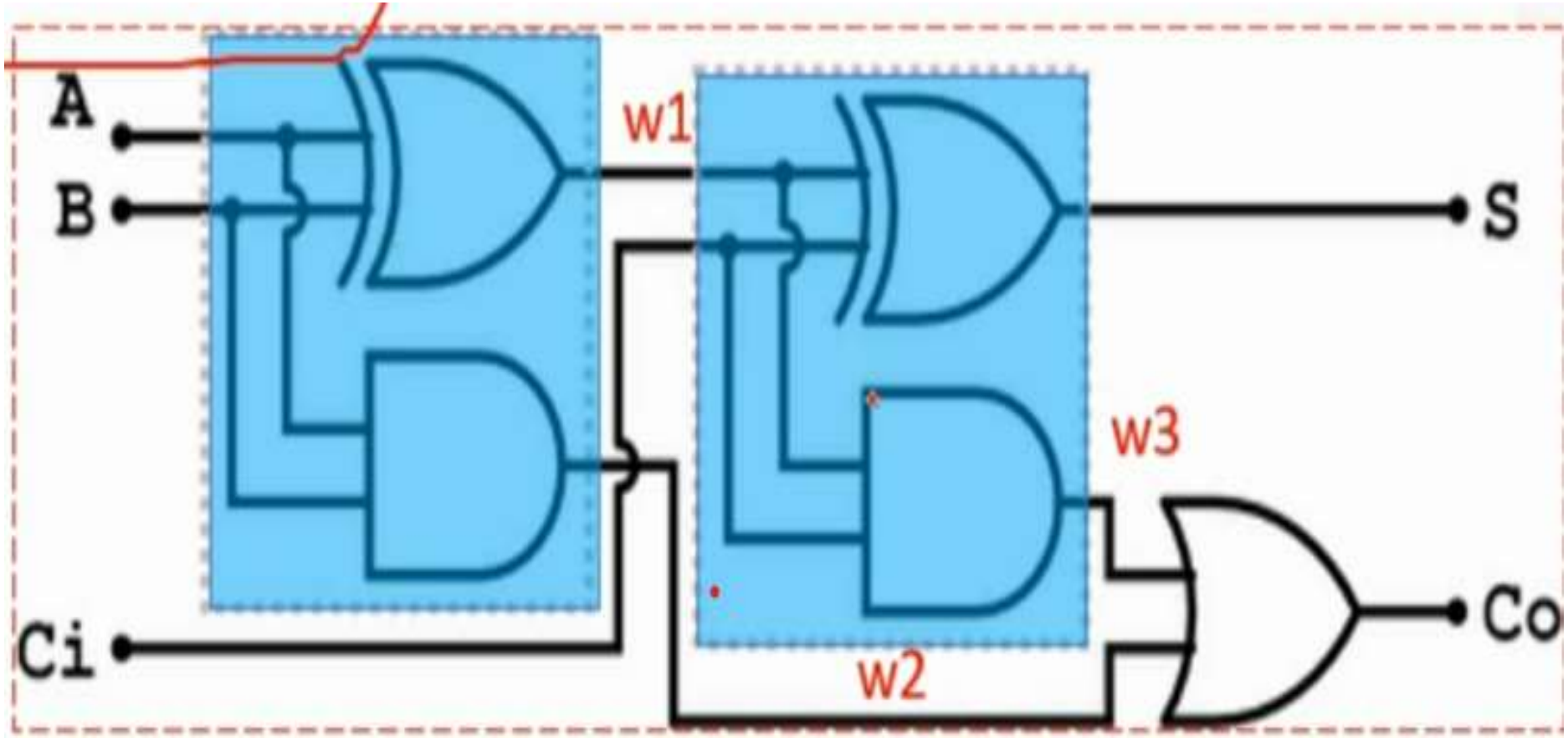
PROGRAM-4.2: DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER

Diagram No –(4.2).4: logical circuit diagram of the full adder circuit using 2-half adder blocks



PROGRAM-4.2: DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER

Diagram No –(4.2).5: logical circuit diagram of the full adder circuit using 2-half adder blocks



PROGRAM-4.1: DESIGN VERILOG HDL TO IMPLEMENT BINARY HALF ADDER

Diagram No –(4.2).6: Full adder logical expressions

Logical expressions of Full adder With logic gates circuit

Sum = A ^ B ^ Cin

Carry = (A & B) | (B & Cin) | (A & Cin)

PROGRAM-4.2: DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER

Verilog Program example for full adder simulation

```
/*circuit description module half
  adder*/

module myhalfadder(A,B,S,C);
  input A,B;
  output S,C;
        xor(S,A,B);
        and(C,A,B);
endmodule

//////////////
```

Program example for full adder simulation

```
/*circuit description module full adder
   using 2-half adders*/

module FullAdder(S,Co,A,B,Ci);
   input A,B,Ci;
   output S,Co;
   wire w1,w2,w3;
   myhalfadder  h1(A,B,w1,w2);
   myhalfadder  h2(w1,Ci,S,w3);
   or(Co,w2,w3);
endmodule
```

PROGRAM-4.2: DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL ADDER

Verilog test bench module Program example for full adder simulation

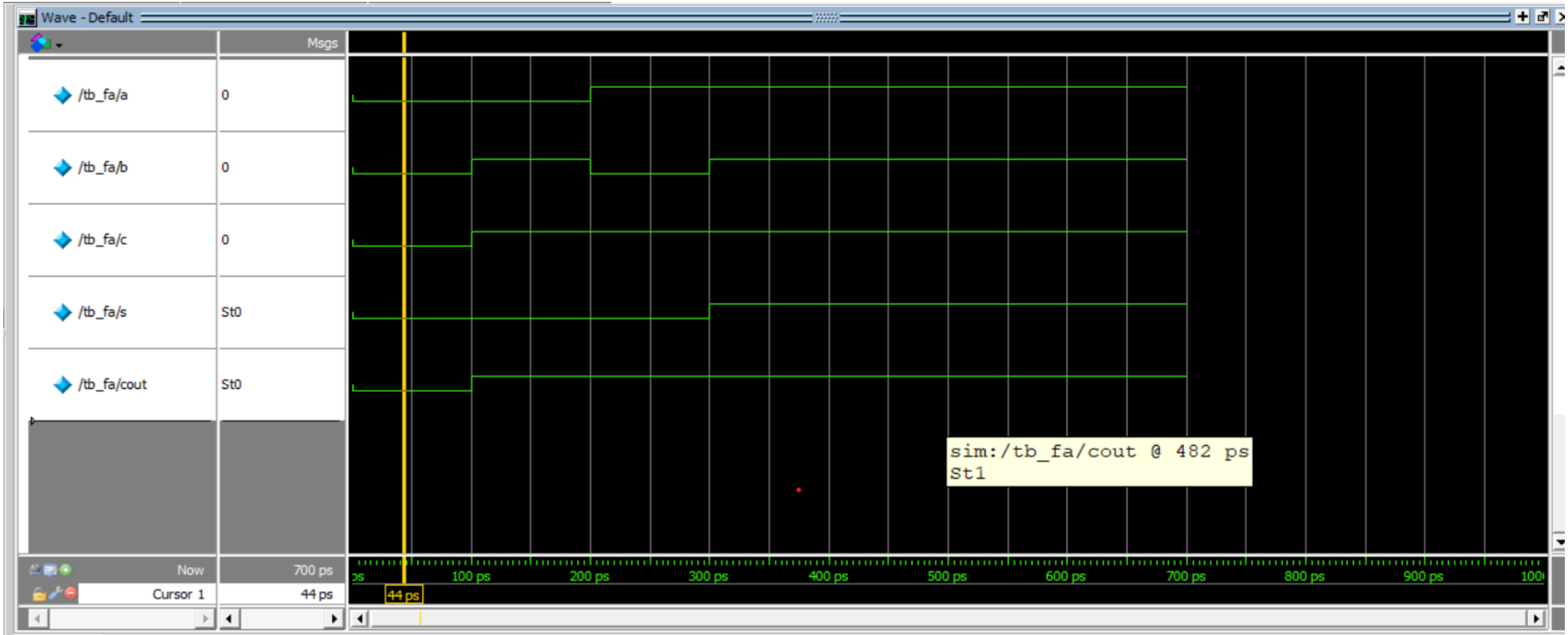
```
module tb_fa( );
   reg a,b,c_i;
   wire s,c_o;

   FullAdder f1(s,c_o,a,b,c_i);

   initial
          begin
                a = 1'b0;  b=1'b0; c_i =1'b0;
                #100
                a = 1'b0;  b=1'b1; c_i =1'b1;
                #100
                a = 1'b1;  b=1'b0; c_i =1'b1;
                #100
                a = 1'b1;  b=1'b1; c_i =1'b0;
```

Verilog test bench module Program example for full adder simulation

```
                end
endmodule
```

PROGRAM-4.2: DESIGN VERILOG HDL TO IMPLEMENT BINARY FULL ADDER

Diagram No –(4.2).7: Full adder Verilog code simulation OUTPUT screen shot



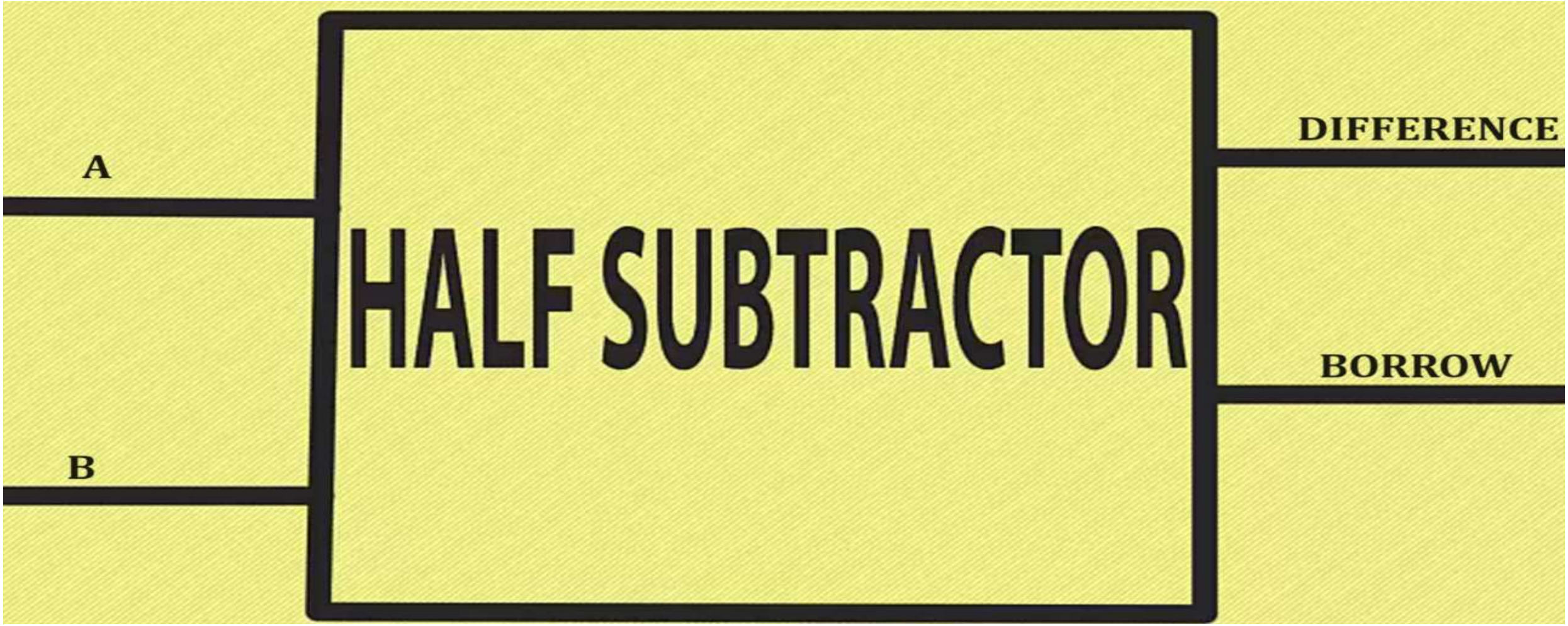
DIGITAL DESIGN & COMPUTER ORGANISATION LAB
(BCS302)

PROGRAM-4.3

DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF HALF SUBTRACTOR

PROGRAM-4.3: DESIGN OF VERILOG CODE FOR HALF SUBTRACTOR IMPLEMENTATION

Diagram No –(4.3).1: Block diagram of half subtractor



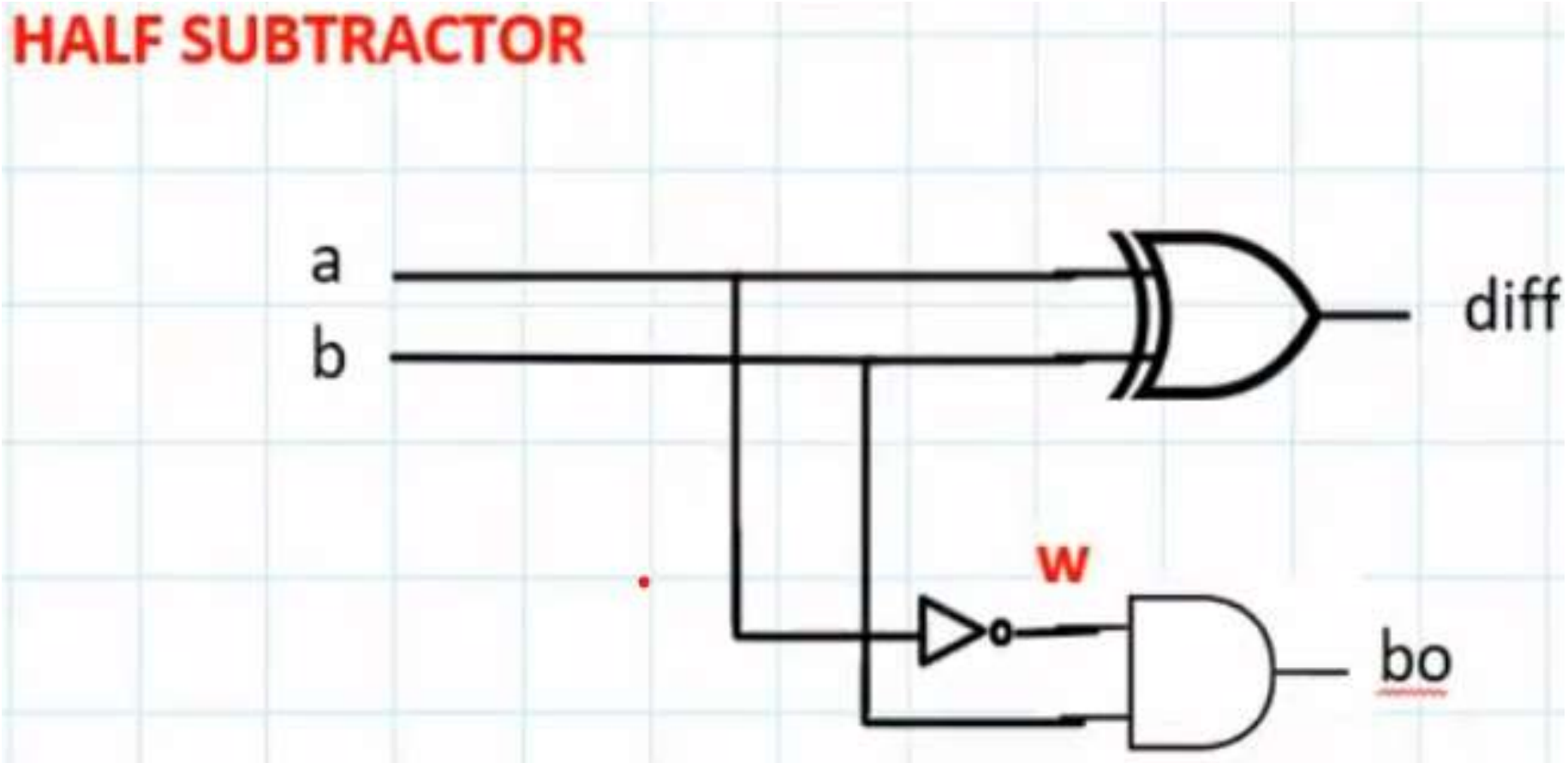
PROGRAM-4.3: DESIGN OF VERILOG CODE FOR HALF SUBTRACTOR IMPLEMENTATION

Diagram No –(4.3).2: Truth Table of half subtractor

A	B	difference	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

PROGRAM-4.3: DESIGN OF VERILOG CODE FOR HALF SUBTRACTOR IMPLEMENTATION

Diagram No –(4.3).3: Logical circuit diagram of half subtractor



PROGRAM-4.3: DESIGN OF VERILOG CODE FOR HALF SUBTRACTOR IMPLEMENTATION

Diagram No –(4.3).4: Logical expression of half subtractor

LOGICAL EXPRESSIONS OF HALF SUBTRACTOR:

DIFFERENCE = A ^ B

BORROW = (~A) & B

PROGRAM-4.3: DESIGN OF VERILOG CODE FOR HALF SUBTRACTOR IMPLEMENTATION

Circuit description module Program example for half subtractor simulation

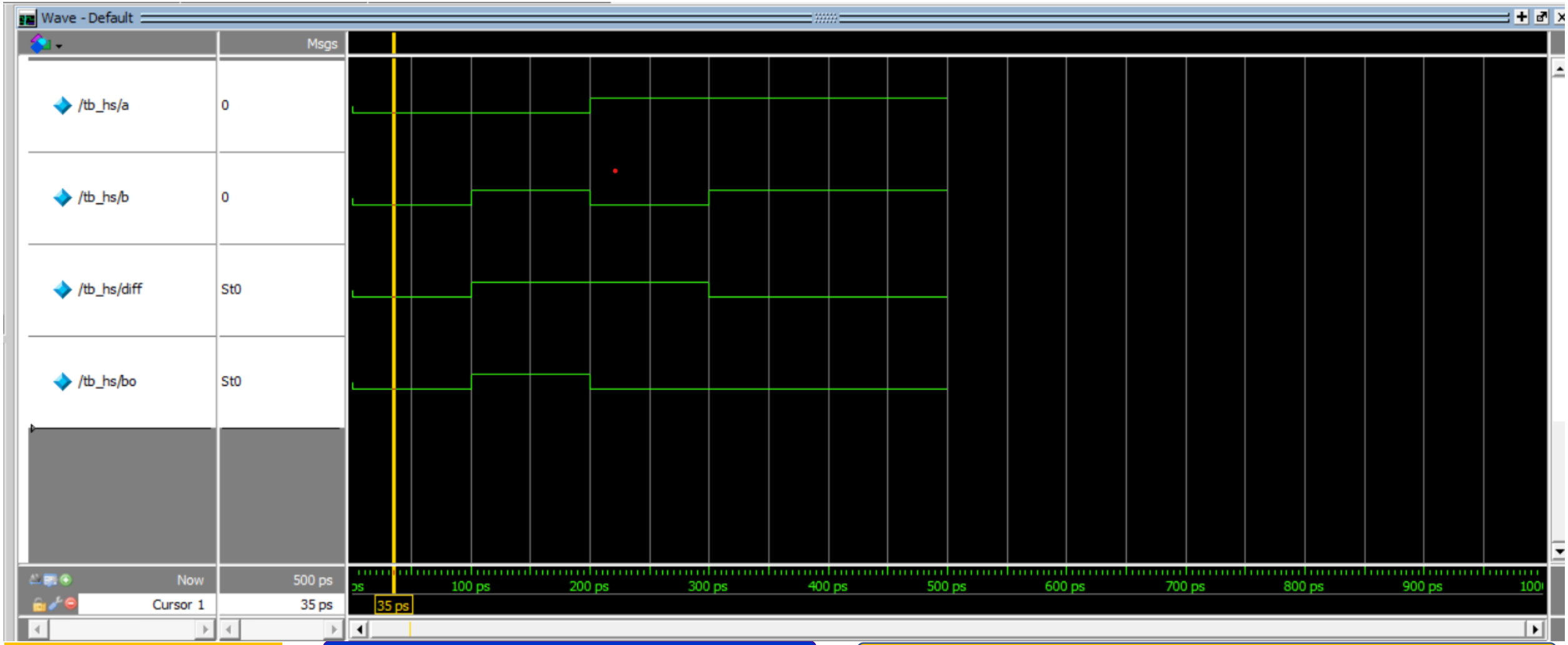
```
module hs(diff,bo,a,b);
        input a,b;
        output diff,bo;
        wire w;
        xor(diff,a,b);
        not(w,a);
        and(bo,w,b);
endmodule
```

Test bench waveform module Program example for half subtractor simulation

```
module tb_hs();
        reg a,b;
        wire diff,bo;
        hs hs1(diff,bo,a,b);
        initial
                begin
                        a=1'b0; b=1'b0;
                        #100
                        a=1'b0; b=1'b1;
                        #100
                        a=1'b1; b=1'b0;
                        #100
                        a=1'b1; b=1'b1;
                end
endmodule
```

PROGRAM-4.3: DESIGN OF VERILOG CODE FOR HALF SUBTRACTOR IMPLEMENTATION

Diagram No –(4.3).5: Half subtractor Verilog code simulation OUTPUT screen shot



DIGITAL DESIGN & COMPUTER ORGANISATION LAB
(BCS302)

PROGRAM-4.4

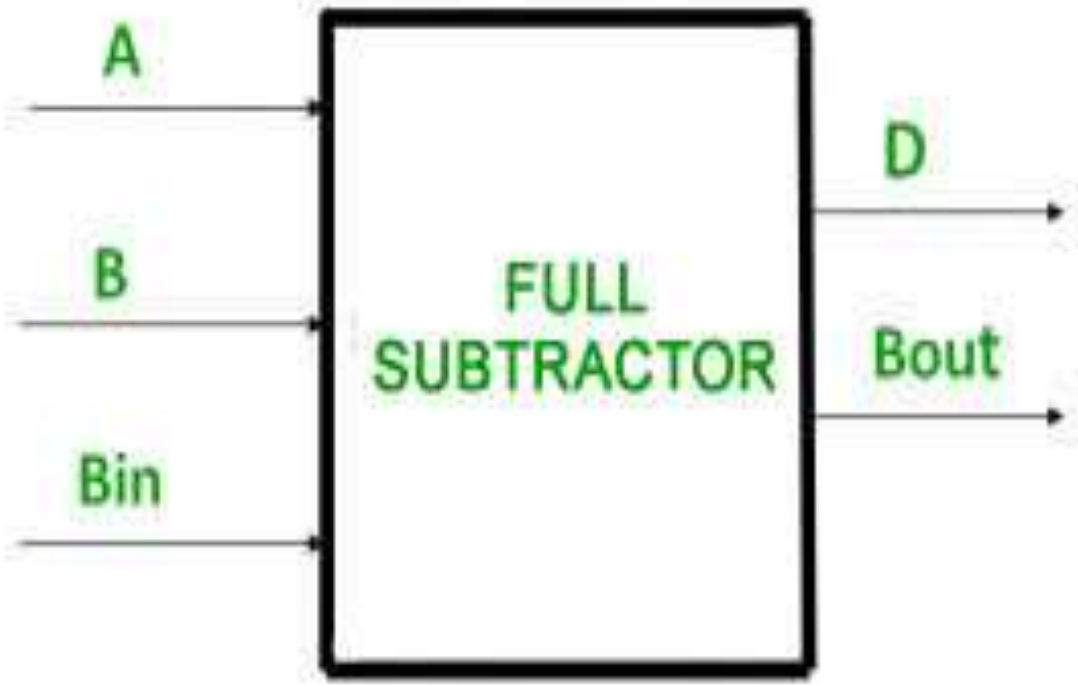
DESIGN OF VERILOG CODE FOR IMPLEMENTATION OF FULL SUBTRACTOR

Mr. PRASHANTH KUMAR S P

Department of Computer Science & Engineering

PROGRAM-4.4: DESIGN OF VERILOG CODE FOR FULL SUBTRACTOR IMPLEMENTATION

Diagram No –(4.4).1: Block diagram of full subtractor



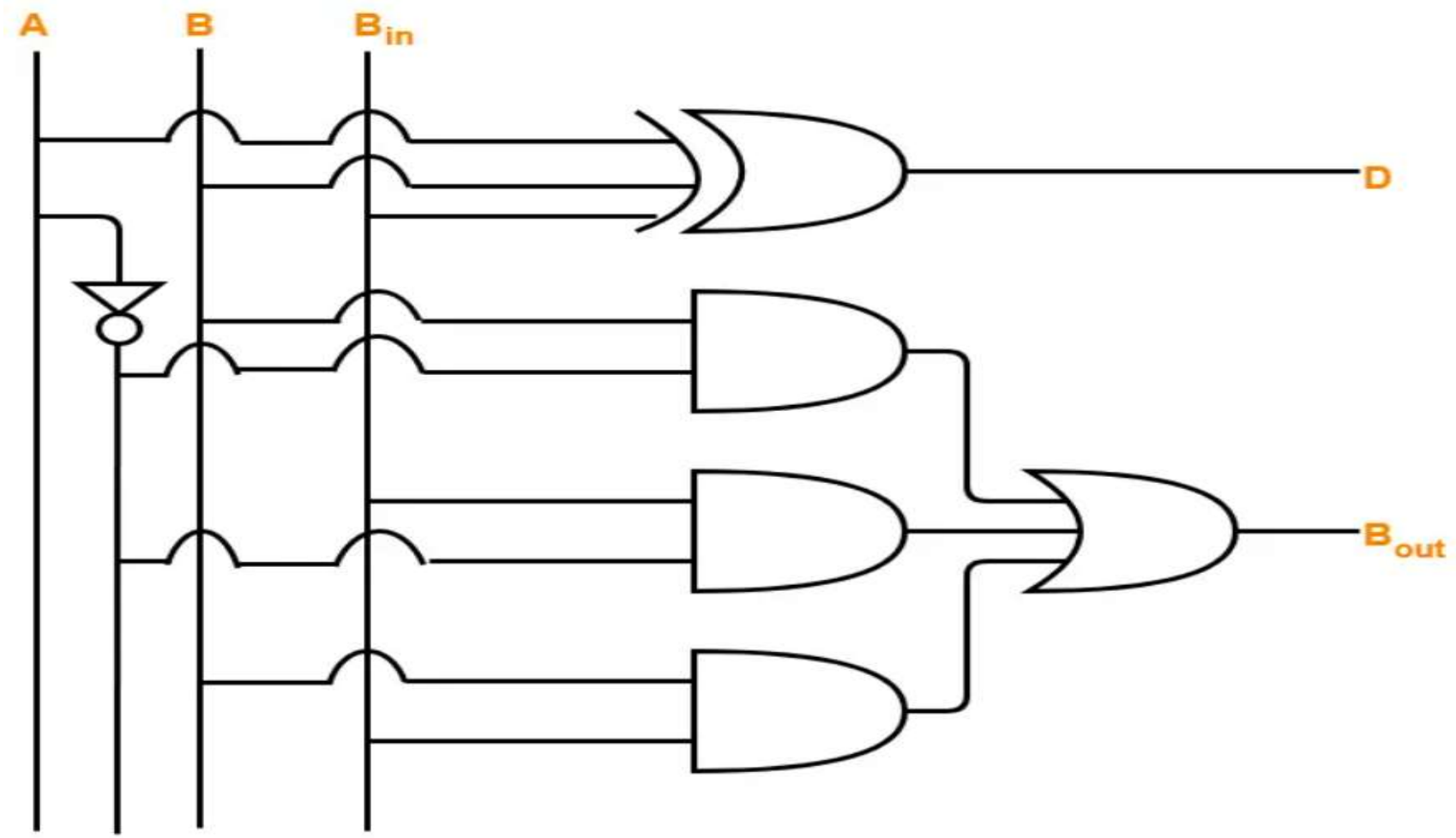
PROGRAM-4.4: DESIGN OF VERILOG CODE FOR FULL SUBTRACTOR IMPLEMENTATION

Diagram No –(4.4).2: Truth Table of Full subtractor

Decimal number	A	B	Borrow in (Bin)	Difference (D)	Borrow out (Bout)
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

PROGRAM-4.4: DESIGN OF VERILOG CODE FOR FULL SUBTRACTOR IMPLEMENTATION

Diagram No –(4.4).3: Logical circuit diagram of Full subtractor



PROGRAM-4.4: DESIGN OF VERILOG CODE FOR FULL SUBTRACTOR IMPLEMENTATION

Diagram No –(4.4).4: Logical Expressions of Full subtractor

LOGICAL EXPRESSIONS OF FULL SUBTRACTOR:

DIFFERENCE = A ^ B ^ Bin

BORROW = ((~A) & B) | (B & Bin) | ((~A) & Bin)

PROGRAM-4.4: DESIGN OF VERILOG CODE FOR FULL SUBTRACTOR IMPLEMENTATION

Circuit description module Verilog Program example for full subtractor simulation

```
module fs ( a, b, c, d, br);
        input a, b, c;
        output d, br;
        assign d= a ^ b ^ c;
        assign br=(( ~a)& (b ^ c)) | (b  & c);
endmodule
```

PROGRAM-4.4: DESIGN OF VERILOG CODE FOR FULL SUBTRACTOR IMPLEMENTATION

Test bench waveform module Verilog Program example for full subtractor simulation

```
module tbfs;
        reg a;
        reg b;
        reg c;
        wire d;
        wire br;
        fs uut (.a(a), .b(b), .c(c),
                        .d(d), .br(br) );
```

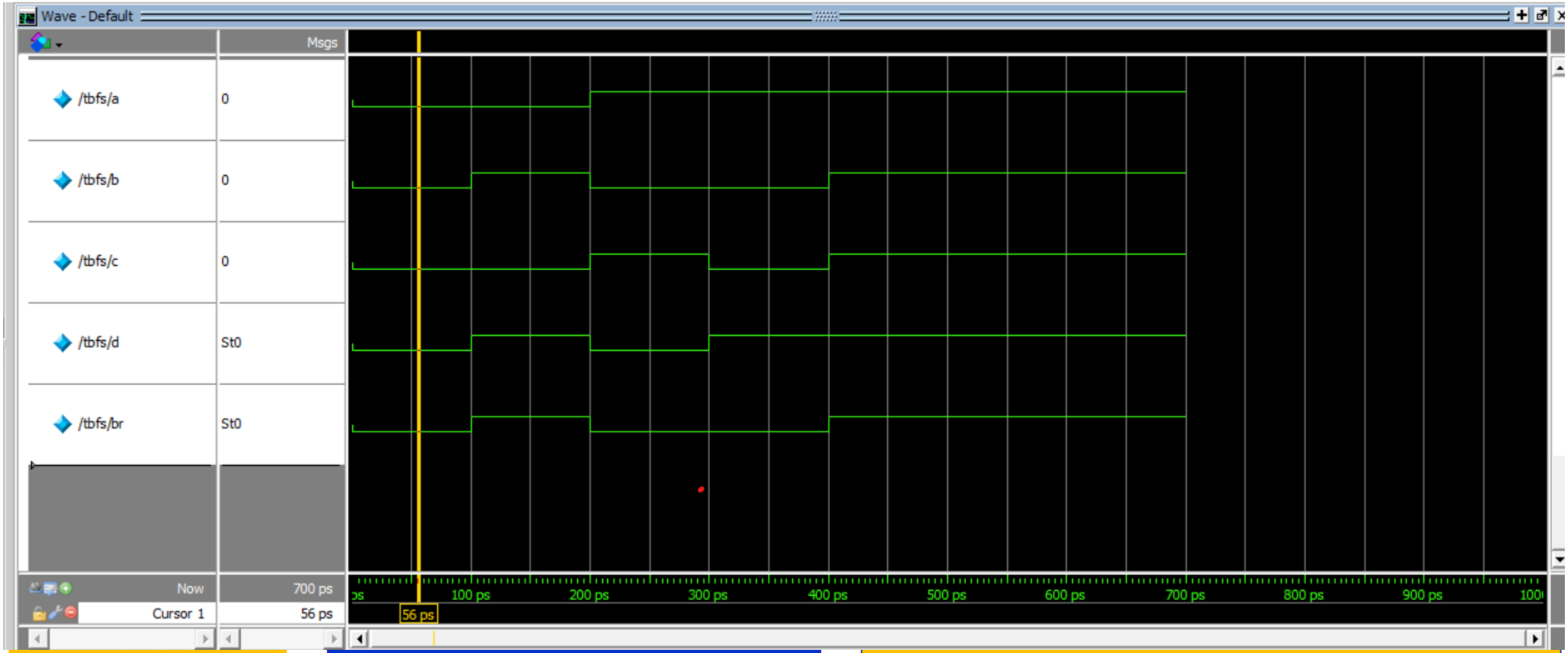
uut = (UNIT UNDER TEST)

Test bench waveform module Verilog Program example for full subtractor simulation

```
initial
        begin
                a = 0;b = 0;c = 0;#100;
                a = 0;b = 1;c = 0;#100;
                a = 1;b = 0;c = 1;#100;
                a = 1;b = 0;c = 0;#100;
                a = 1;b = 1;c = 1;#100;
        end
endmodule
```

PROGRAM-4.4: DESIGN OF VERILOG CODE FOR FULL SUBTRACTOR IMPLEMENTATION

Diagram No –(4.4).5: Full subtractor Verilog code simulation OUTPUT screen shot



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 6, 2024

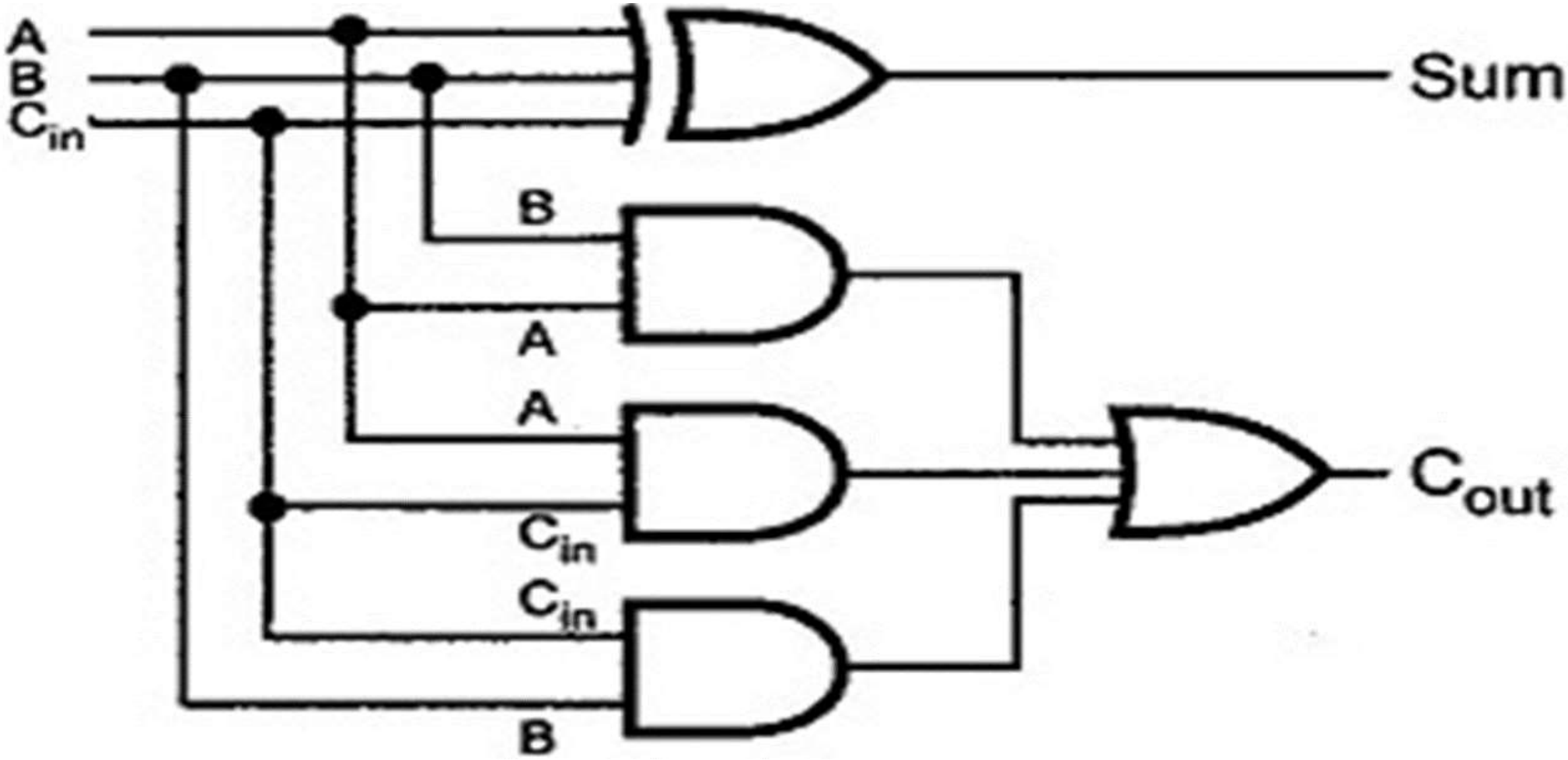
DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-5

DESIGN VERILOG HDL TO IMPLEMENT DECIMAL ADDER/BCD ADDER.

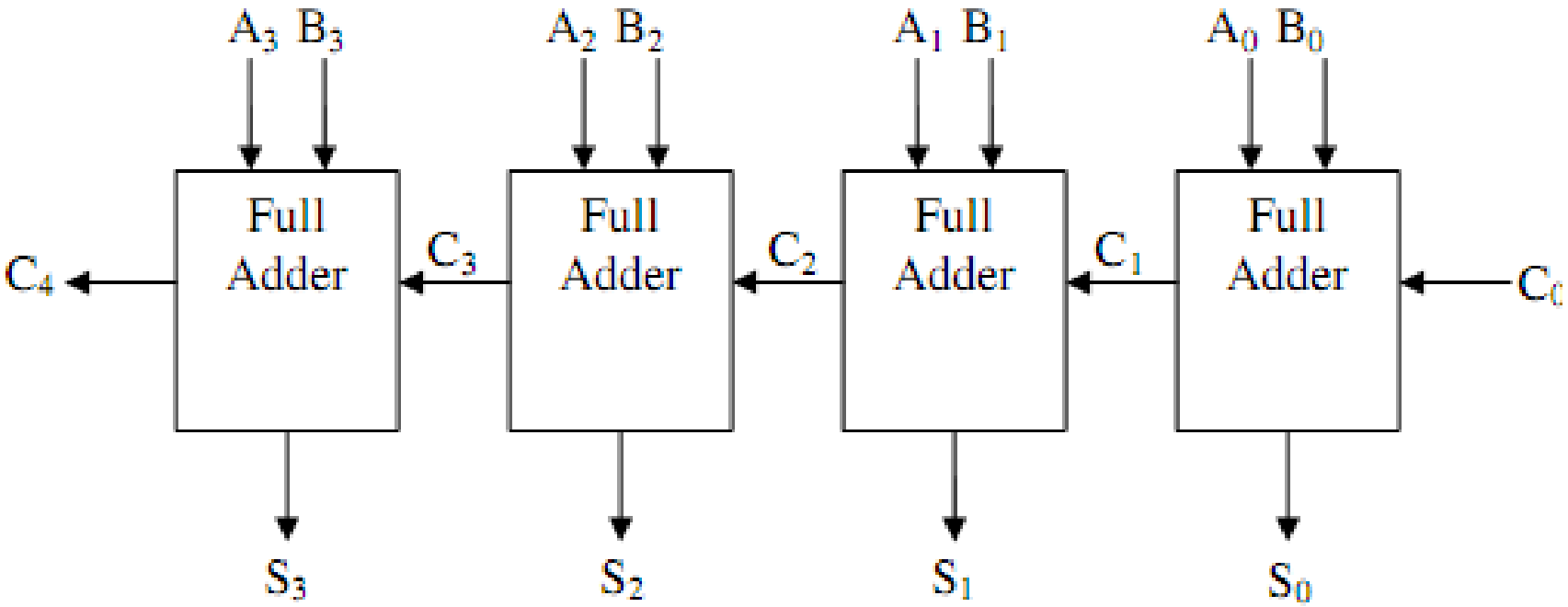
PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Diagram No –(5.1).1: LOGICAL CIRCUIT DIAGRAM OF A FULL ADDER



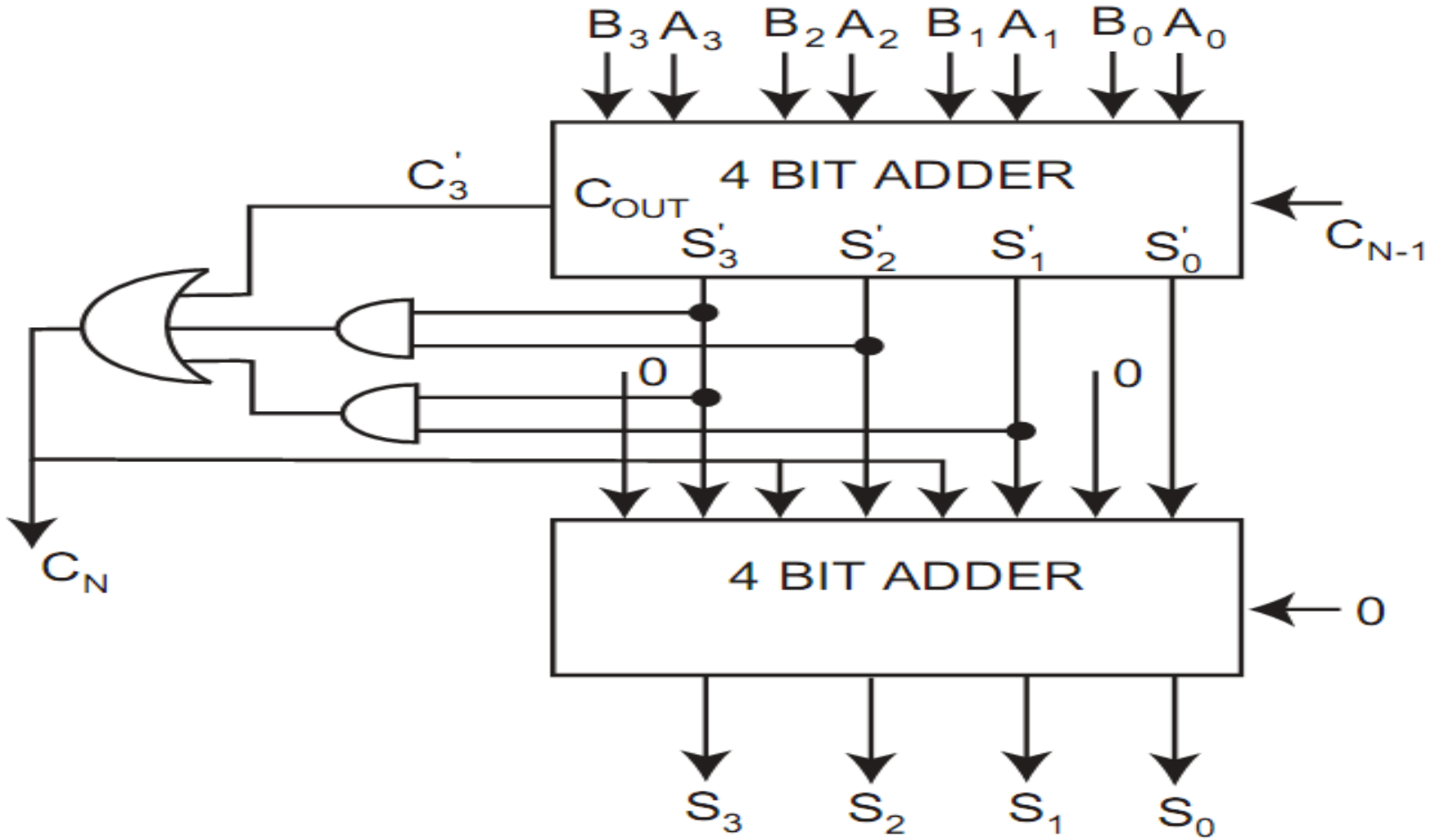
PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Diagram No –(5.1).2: BLOCK DIAGRAM OF 4-BIT FULL ADDER CONSTRUCTED FROM 4 FULL ADDERS



PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Diagram No –(5.1).3: BLOCK DIAGRAM OF BCD ADDER/DECIMAL ADDER CONSTRUCTED BY USING 4-BIT ADDERS



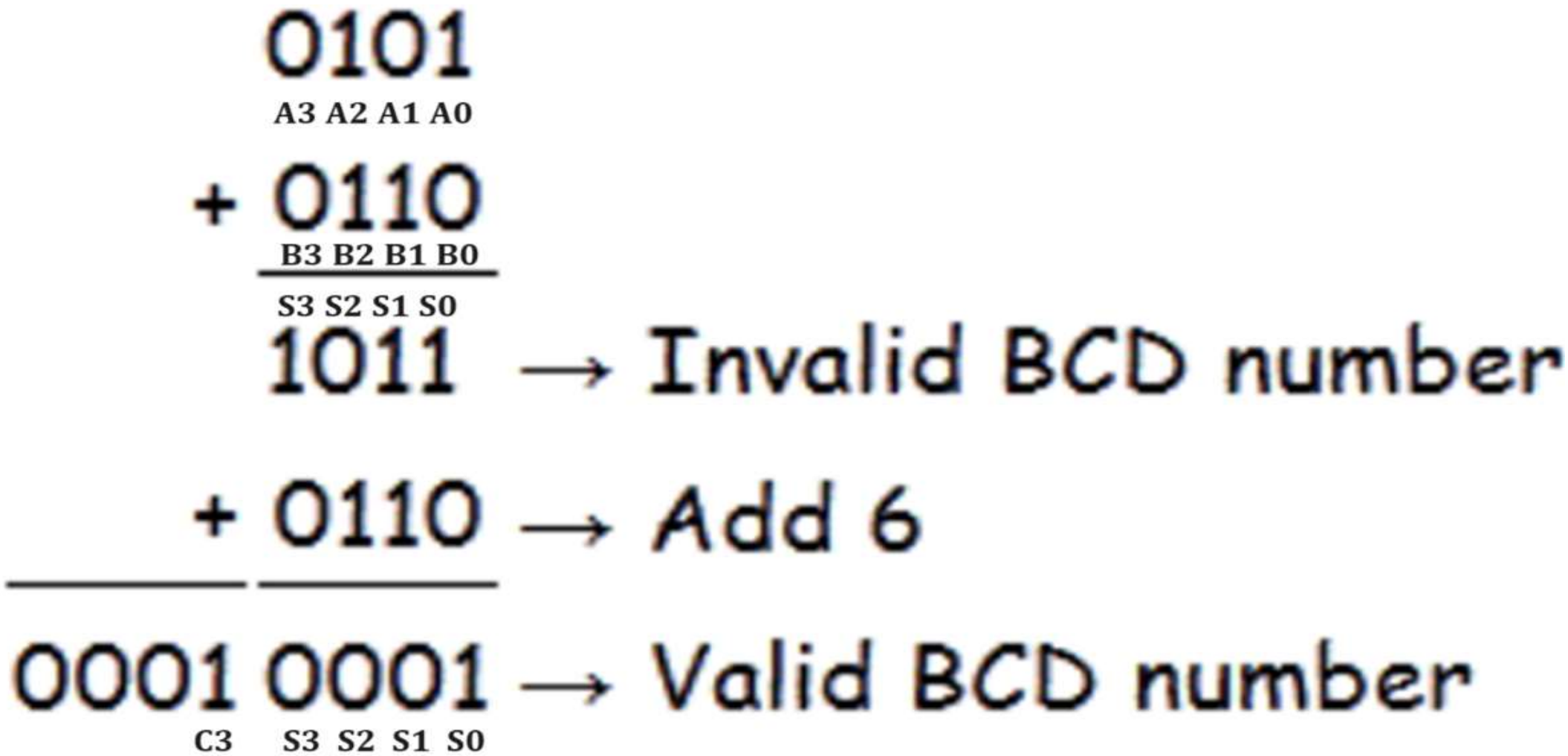
PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Diagram No –(5.1).4: 4-bit BCD CODE EQUIVALENT WITH DECIMAL FORM

DECIMAL EQUIVALENT	BINARY CODED DECIMAL			
	A3	A2	A1	A0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Diagram No –(5.1).5: 4-bit DECIMAL ADDITION/BCD ADDITION E.G.,



PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Circuit description module Verilog program example for implementing decimal/bcd adder (behavioural model)

```
module bcd_adder(a,b,carry_in,sum,carry_out);
input [3:0]a,b;
input carry_in;
output [3:0]sum;
output carry_out;
reg [3:0]sum;
reg carry_out;
reg [4:0]sum_temp;

always@(a,b,carry_in)
    begin
            sum_temp = a + b + carry_in;
            if(sum_temp>9)
            begin
                    sum_temp = sum_temp + 6;
                    carry_out = 1;
```

Circuit description module Verilog program example for implementing decimal/bcd adder (behavioural model)

```
                    sum = sum_temp[3:0];
            end
            else
            begin
                    carry_out = 0;
                    sum = sum_temp[3:0];
            end
    end
endmodule
```

PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Test bench waveform Verilog program example for implementing decimal/bcd adder

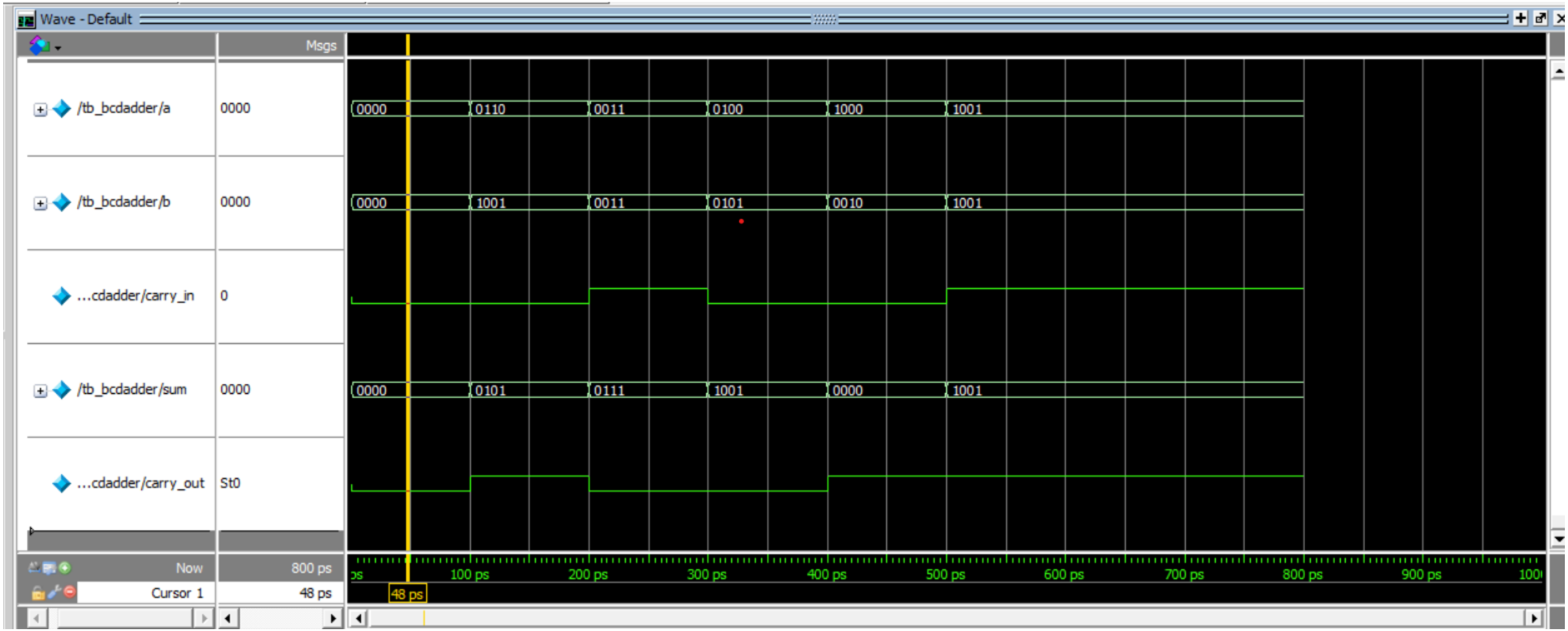
```
module tb_bcdadder();
  reg[3:0]a; reg[3:0]b; reg carry_in;
  wire[3:0]sum; wire carry_out;
  bcd_adder
  uut(.a(a),.b(b),.carry_in(carry_in),
      .sum(sum),.carry_out(carry_out));
```

Test bench waveform Verilog program example for implementing decimal/bcd adder

```
    initial
        begin
            a=0; b=0; carry_in=0; #100
            a=6; b=9; carry_in=0; #100
            a=3; b=3; carry_in=1; #100
            a=4; b=5; carry_in=0; #100
            a=8; b=2; carry_in=0; #100
            a=9; b=9; carry_in=1;
        end
endmodule
```

PROGRAM-5: VERILOG CODE FOR IMPLEMENTATION OF DECIMAL ADDER/BCD ADDER

Diagram No –(5.1).6: output screen shot of decimal adder Verilog code



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 5, 2024

DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-6

DESIGN VERILOG PROGRAM TO IMPLEMENT DIFFERENT TYPES OF MULTIPLEXER LIKE 2:1, 4:1 AND 8:1.

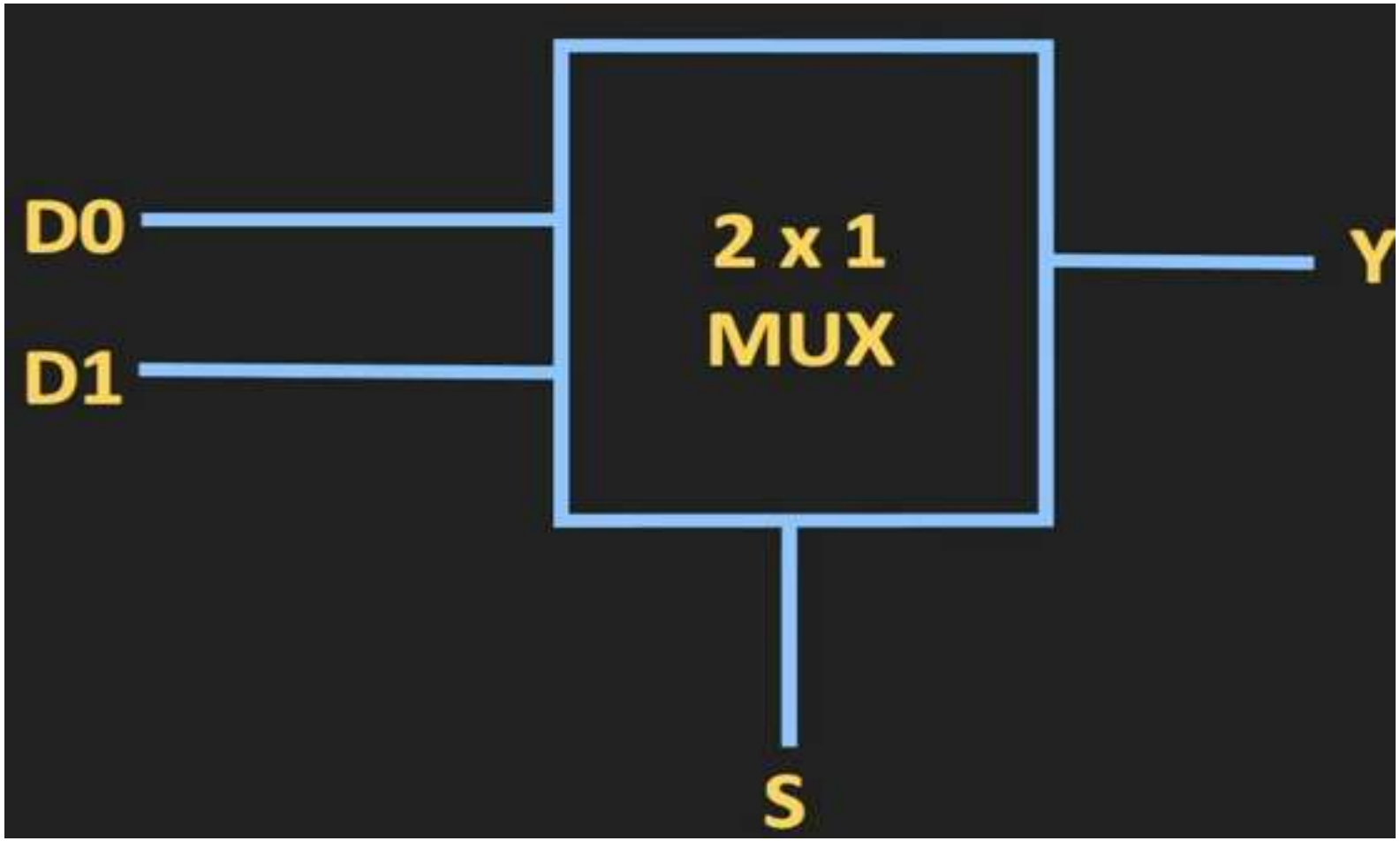
DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-6.1

DESIGN VERILOG PROGRAM TO IMPLEMENT 2:1 MULTIPLEXER

PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 2:1 MULTIPLEXER

Diagram No –(6.1).1: BLOCK DIAGRAM OF 2:1 MUX



PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 2:1 MULTIPLEXER

Diagram No –(6.1).2: TRUTH TABLE OF 2:1 MUX

Truth Table

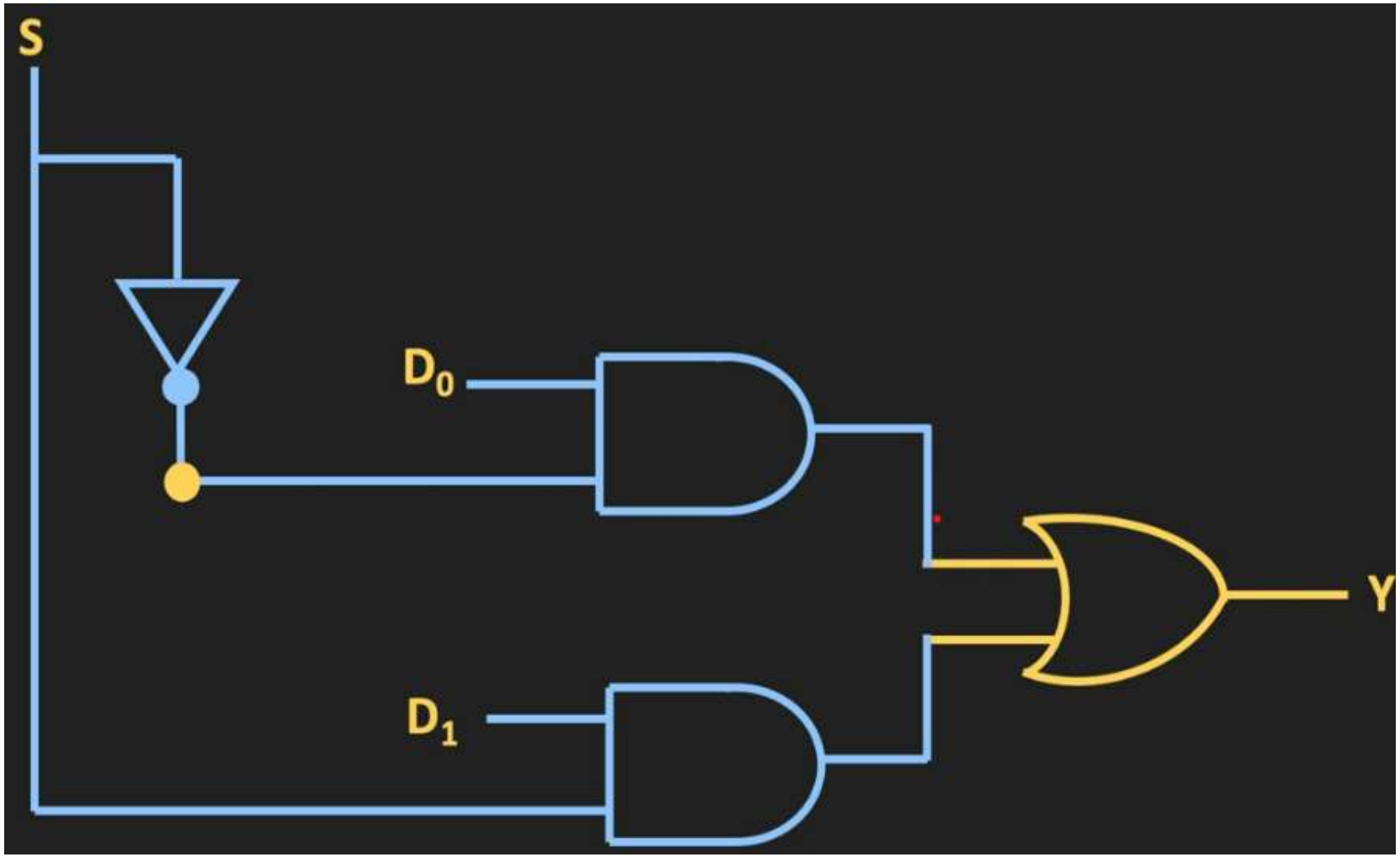
D1	D0	S	Y	Y
0	0	0	0	D0
0	0	1	0	D1
0	1	0	1	D0
0	1	1	0	D1
1	0	0	0	D0
1	0	1	1	D1
1	1	0	1	D0
1	1	1	1	D1

Truth Table

S	Y
0	D0
1	D1

PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 2:1 MULTIPLEXER

Diagram No –(6.1).3: LOGICAL CIRCUIT DIAGRAM OF 2:1 MUX



PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 2:1 MULTIPLEXER

Diagram No –(6.1).4: LOGICAL EXPRESSIONS OF 2:1 MUX

LOGICAL EXPRESSION OF 2:1MUX:

$$Y = S'D0 + SD1$$

PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 2:1 MULTIPLEXER

Circuit description module Verilog Program example for implementing 2:1 MUX(behavioural model)

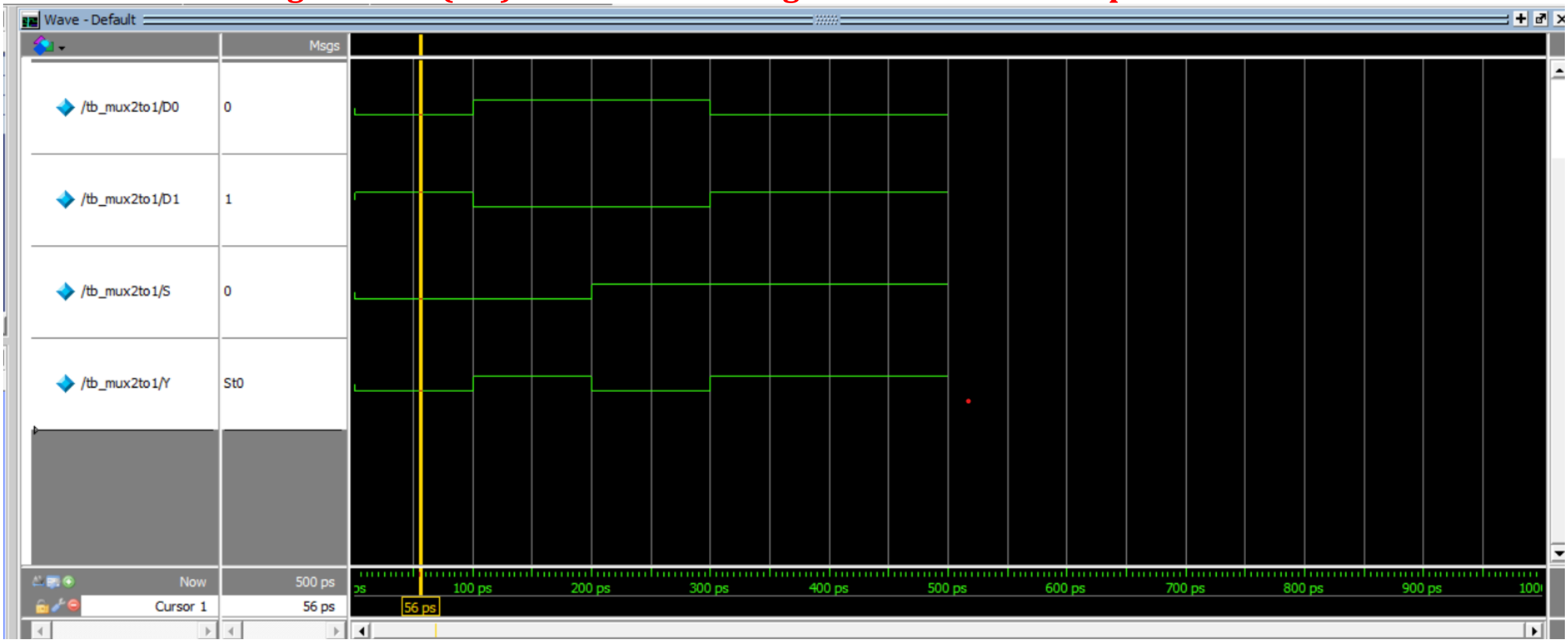
```
module mux2to1(D0,D1,S,Y);
  input wire D0,D1,S;
  output reg  Y;

  always@(D0 or D1 or S)
        begin
              if(S)
              Y = D1;
              else
              Y = D0;
        end
endmodule
```

Test bench waveform module Verilog Program example for implementing 2:1 MUX

```
module  tb_mux2to1( );
        reg D0,D1,S;
        wire Y;
        mux2to1  mu1(D0,D1,S,Y);
        initial
              begin
                    D0 = 1'b0; D1=1'b1; S = 1'b0;
                    #100
                    D0=1'b1; D1 = 1'b0; S = 1'b0;
                    #100
                    D0=1'b1; D1 = 1'b0; S = 1'b1;
                    #100
                    D0=1'b0; D1 = 1'b1; S = 1'b1;
              end
endmodule
```

PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 2:1 MULTIPLEXER

Diagram No –(6.1).5: MUX 2:1 Verilog code simulation output screen shot



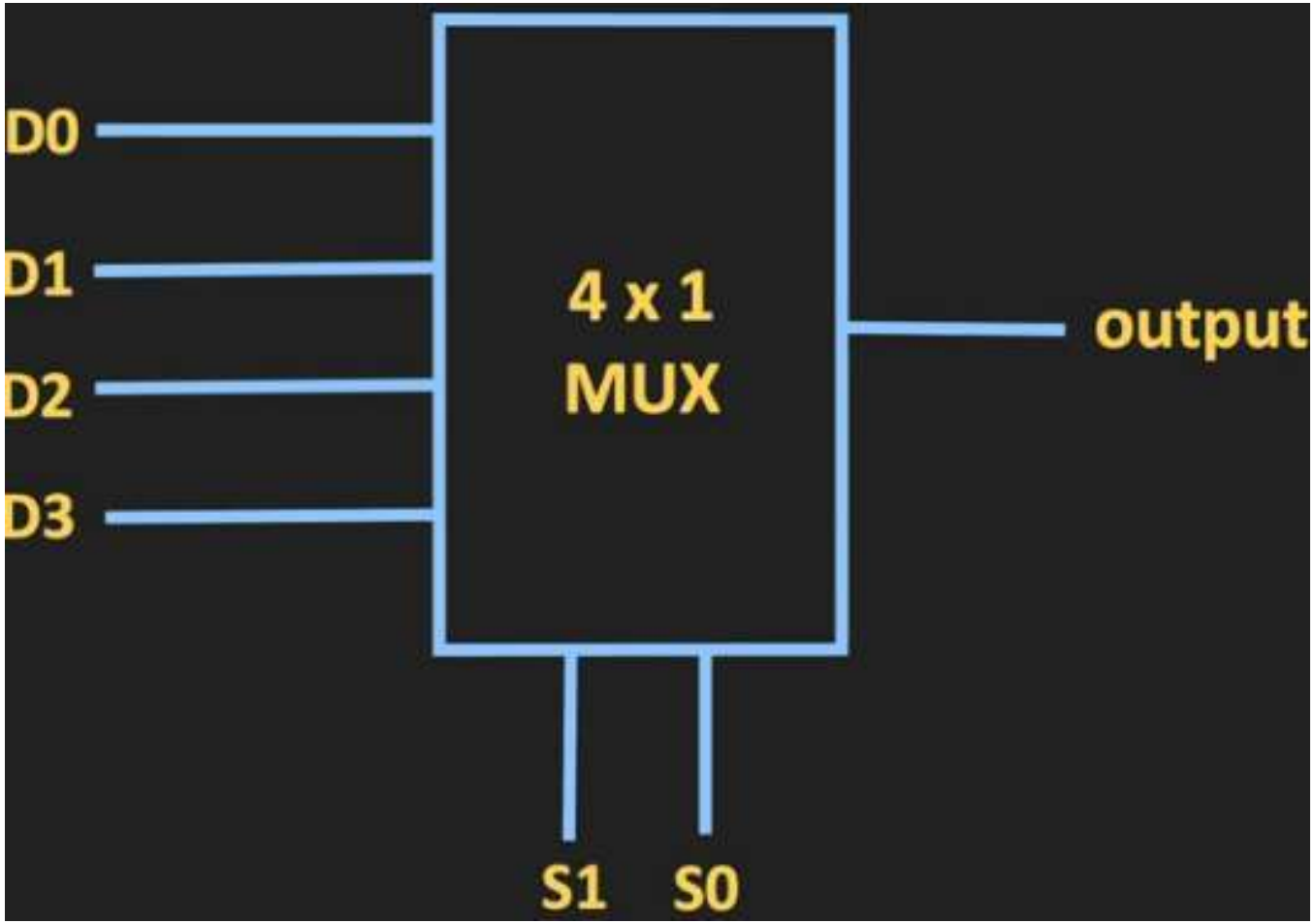
DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-6.2

DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

PROGRAM-6.2: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Diagram No –(6.2).1: BLOCK DIAGRAM OF 4:1 MUX



PROGRAM-6.2: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Diagram No –(6.2).2: TRUTH TABLE OF 4:1 MUX

Truth Table

S1	S0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

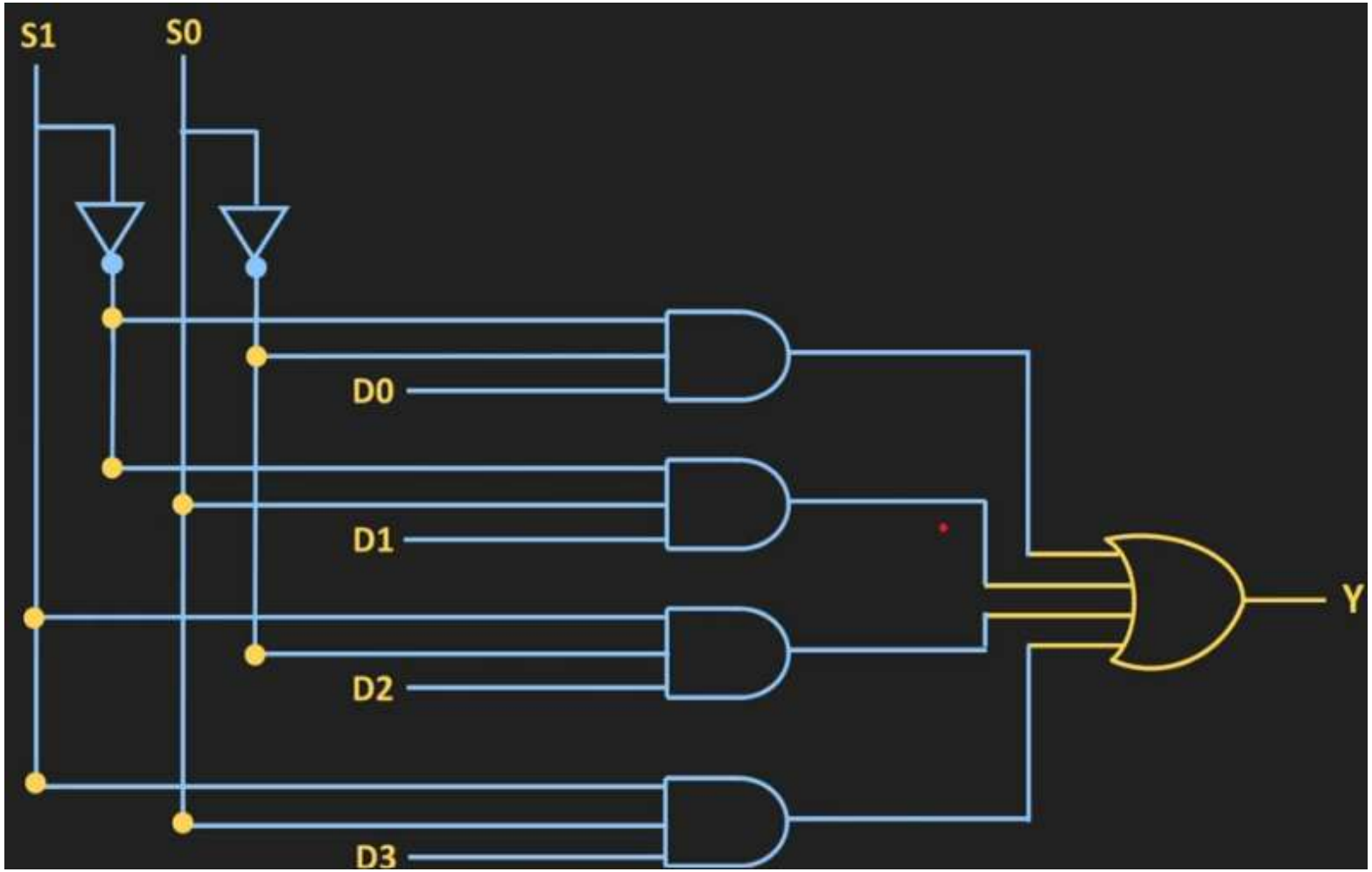
PROGRAM-6.2: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Diagram No –(6.2).3: DETAILED TRUTH TABLE OF 4:1 MUX

S1	S0	D0	D1	D2	D3	Y
0	0	0	1	1	1	0
0	0	1	0	0	0	1
0	1	1	0	1	1	0
0	1	0	1	0	0	1
1	0	1	1	0	1	0
1	0	0	0	1	0	1
1	1	1	1	1	0	0
1	1	0	0	0	1	1

PROGRAM-6.2: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Diagram No –(6.2).4: LOGICAL CIRCUIT DIAGRAM OF 4:1 MUX



PROGRAM-6.2: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Diagram No –(6.2).5: LOGICAL EXPRESSIONS OF 4:1 MUX

$$Y = \overline{S1}\,\overline{S0}\,D0 + \overline{S1}\,S0\,D1 + S1\,\overline{S0}\,D2 + S1\,S0\,D3$$

LOGICAL EXPRESSION OF 4:1MUX:

Y = S1'S0'D0 + S1'S0D1 + S1S0'D2 + S1S0D3

PROGRAM-6.2: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Circuit description module Program example for implementing 4:1 MUX (data flow model)

```
module mux4to1(D0,D1,D2,D3,S1,S0,Y);
  input D0,D1,D2,D3,S1,S0;
  output Y;

  assign Y= ((~S1)&(~S0)&(D0))|
  ((~S1)&(S0)&(D1)) |((S1)&(~S0)&(D2))
  | ((S1)&(S0)&(D3));

endmodule
```

Test bench waveform module Program example for implementing 4:1 MUX (data flow model)

```
module tbmux4_1();
      reg S0,S1;
      reg D0,D1,D2,D3;
      wire Y;
      mux4to1  M1(D0,D1,D2,D3,S1,S0,Y);
```

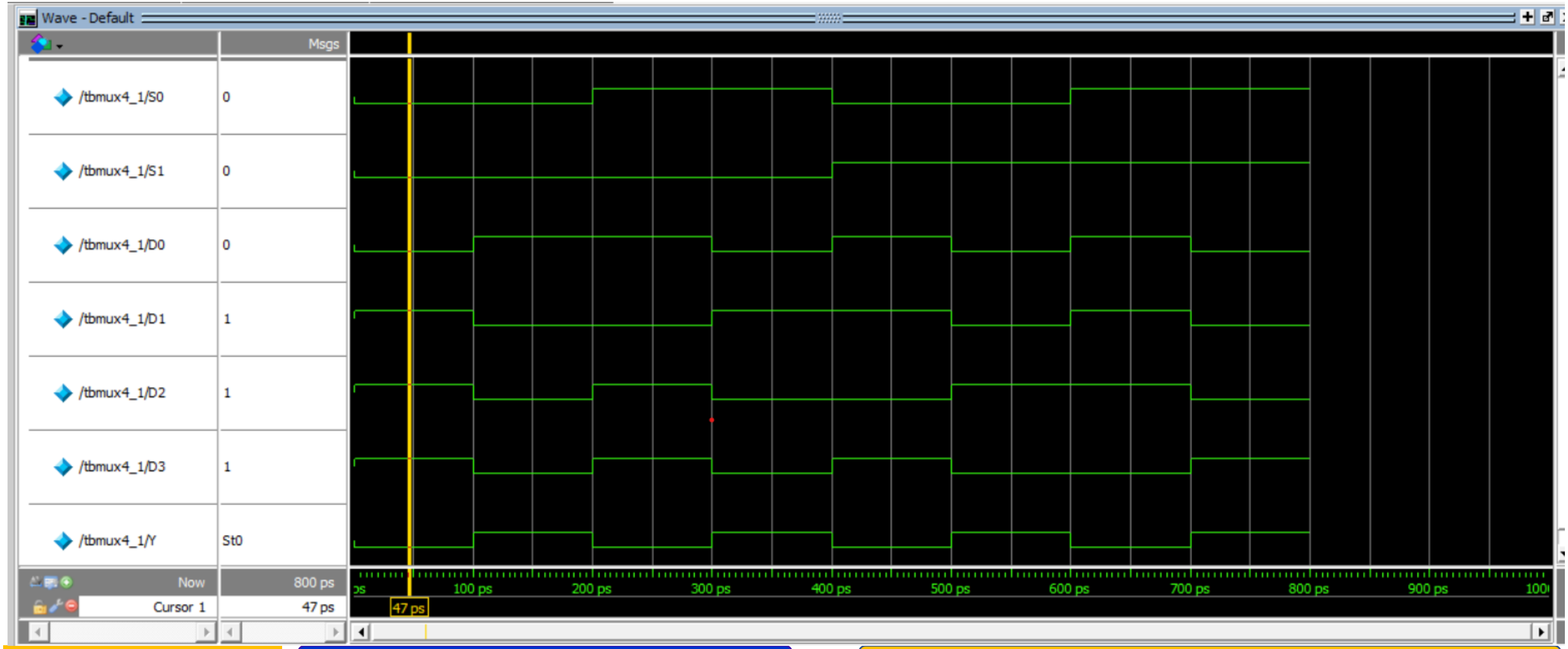
PROGRAM-6.2: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Test bench waveform module Program example for implementing 4:1 MUX (data flow model)

```
initial
        begin
                D0=1'b0;D1=1'b1;D2=1'b1;D3=1'b1;S1=0;S0=0;
                #100
                D0=1'b1;D1=1'b0;D2=1'b0;D3=1'b0;S1=0;S0=0;
                #100
                D0=1'b1;D1=1'b0;D2=1'b1;D3=1'b1;S1=0;S0=1;
                #100
                D0=1'b0;D1=1'b1;D2=1'b0;D3=1'b0;S1=0;S0=1;
                #100
                D0=1'b1;D1=1'b1;D2=1'b0;D3=1'b1;S1=1;S0=0;
                #100
                D0=1'b0;D1=1'b0;D2=1'b1;D3=1'b0;S1=1;S0=0;
                #100
                D0=1'b1;D1=1'b1;D2=1'b1;D3=1'b0;S1=1;S0=1;
                #100
                D0=1'b0;D1=1'b0;D2=1'b0;D3=1'b1;S1=1;S0=1;
        end
endmodule
```

PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 4:1 MULTIPLEXER

Diagram No –(6.2).6: MUX 4:1 Verilog code simulation output screen shot



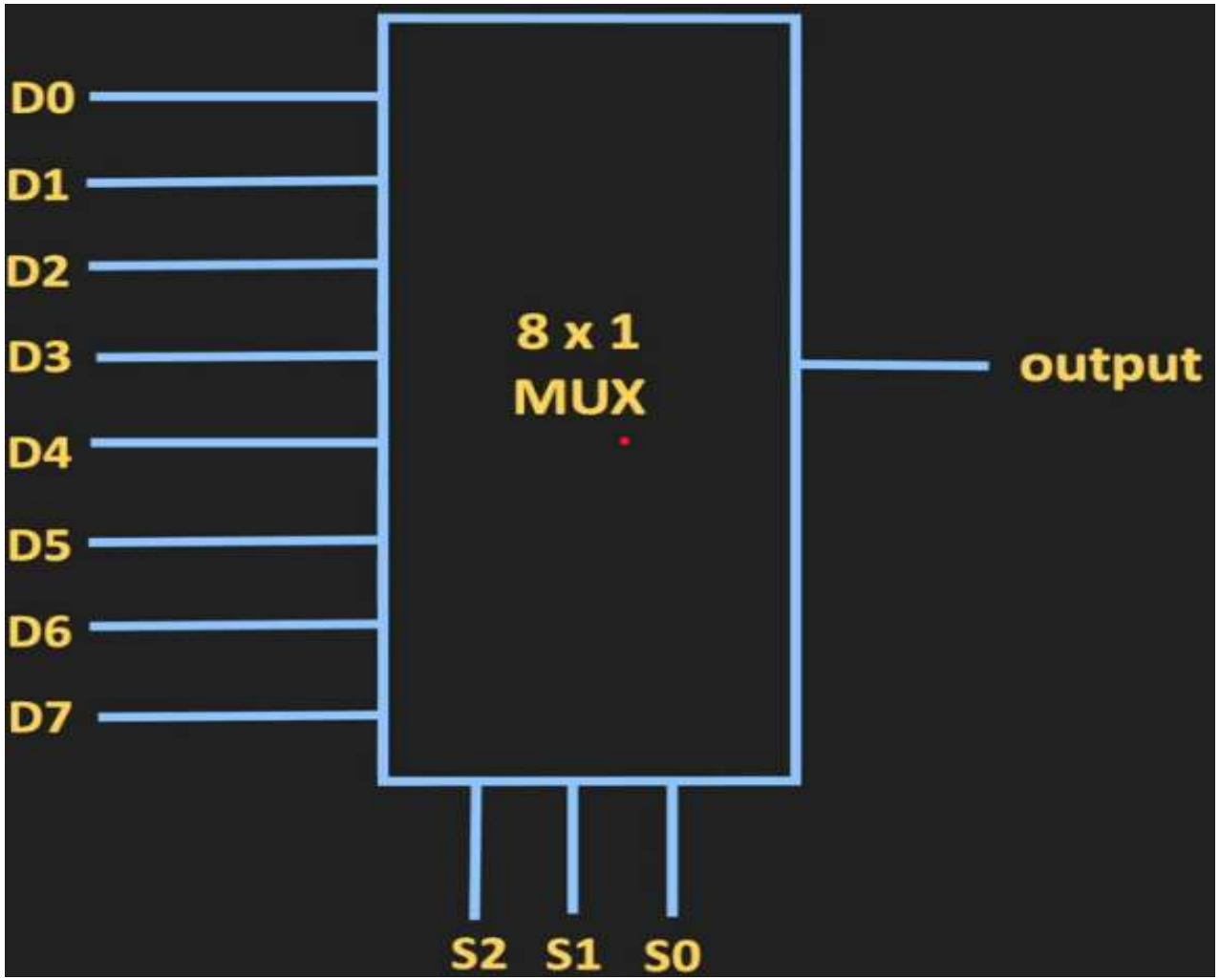
DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-6.3

DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

PROGRAM-6.3: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

Diagram No –(6.3).1: BLOCK DIAGRAM OF 8:1 MUX



PROGRAM-6.3: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

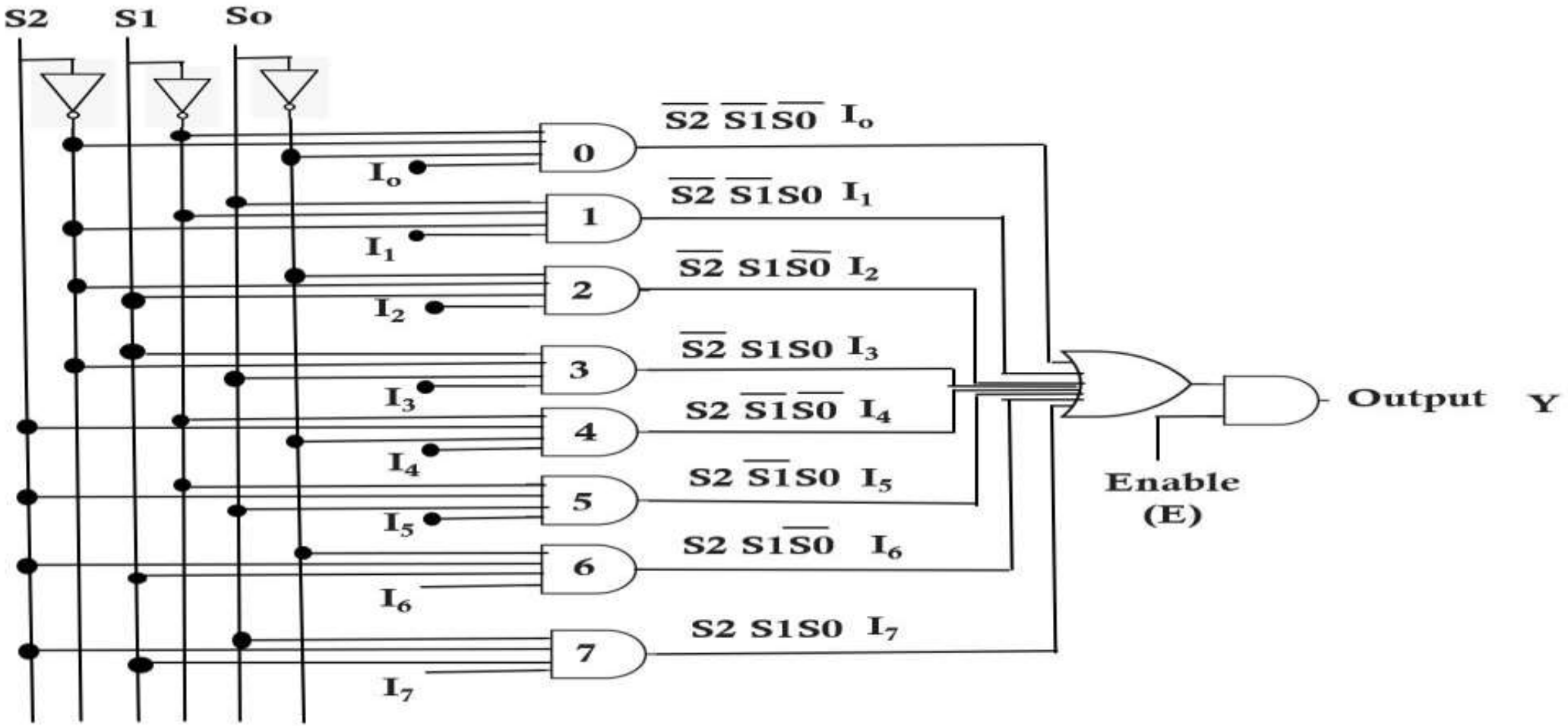
Diagram No –(6.3).2: TRUTH TABLE OF 8:1 MUX

Truth Table

S2	S1	S0	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

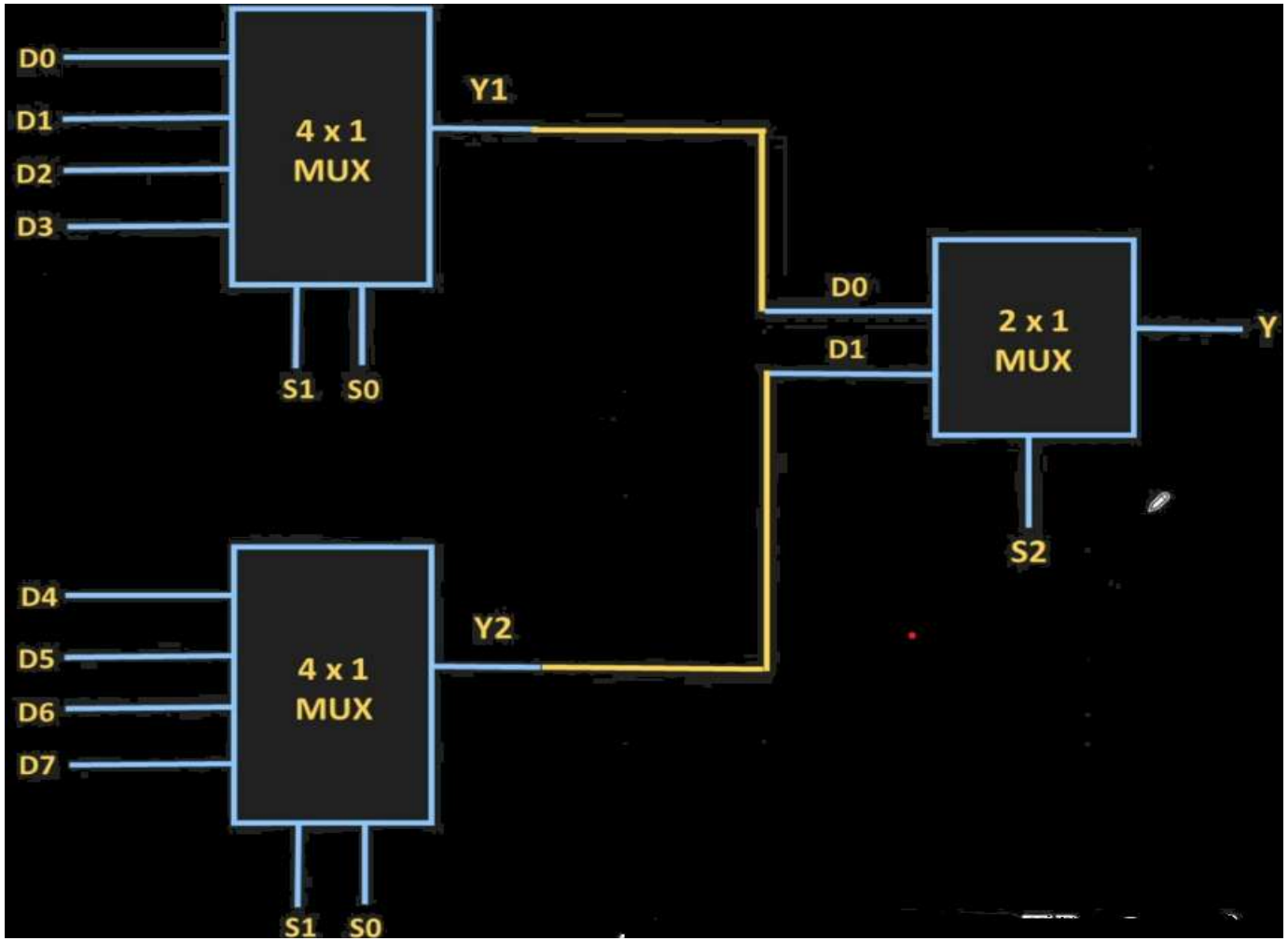
PROGRAM-6.3: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

Diagram No –(6.3).3: LOGICAL CIRCUIT DIAGRAM OF 8:1 MUX



PROGRAM-6.3: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

Diagram No –(6.3).4: LOGICAL CIRCUIT DIAGRAM OF 8:1 MUX USING 4:1 & 2:1 MUX



PROGRAM-6.3: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

Diagram No –(6.3).5: LOGICAL EXPRESSIONS OF 8:1 MUX

LOGICAL EXPRESSION FOR 8:1 MUX:

Y=(S0'.S1'.S2'.D0) + (S0.S1'.S2'.D1) + (S0'.S1.S2'.D2) + (S0.S1.S2'.D3) +
(S0'.S1'.S2 D4) + (S0.S1'.S2 D5) + (S0'.S1.S2 .D6) + (S0.S1.S3.D7)

PROGRAM-6.3: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

Circuit description module Verilog Program example for implementing 8:1 MUX (behavioural model)

```
module mux8(I,s,y);
   input [7:0]I;
   input [2:0]s;
   output y;
   reg y;

   always @(s,I)
   begin
            if (s==000) y = I[0];
            else if (s==001) y=I[1];
            else if (s==010) y=I[2];
            else if (s==011) y=I[3];
            else if (s==100) y=I[4];
            else if (s==101) y=I[5];
            else if (s==110) y=I[6];
            else if (s==111) y=I[7];
```

Circuit description module Verilog Program example for implementing 8:1 MUX (behavioural model)

```
                    end
endmodule
```

PROGRAM-6.3: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

Test bench waveform verilog Program example for implementing 8:1 MUX

```
module mux8_tb;

   // Inputs
   reg [7:0] I;
   reg [2:0] s;

   // Outputs
   wire y;
   mux8 uut(.I(I), .s(s), .y(y));

   initial
           begin
                   s=0;I=1; #100
                   s=1;I=2; #100
                   s=2;I=4; #100
```

Test bench waveform verilog Program example for implementing 8:1 MUX

```
                   s=3;I=8;          #100
                   s=4;I=16;         #100
                   s=5;I=32;         #100
                   s=6;I=64;         #100
                   s=7;I=128;

           end
endmodule
```

PROGRAM-6.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 8:1 MULTIPLEXER

Diagram No –(6.3).6: MUX 8:1 Verilog code simulation output screen shot

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 5, 2024

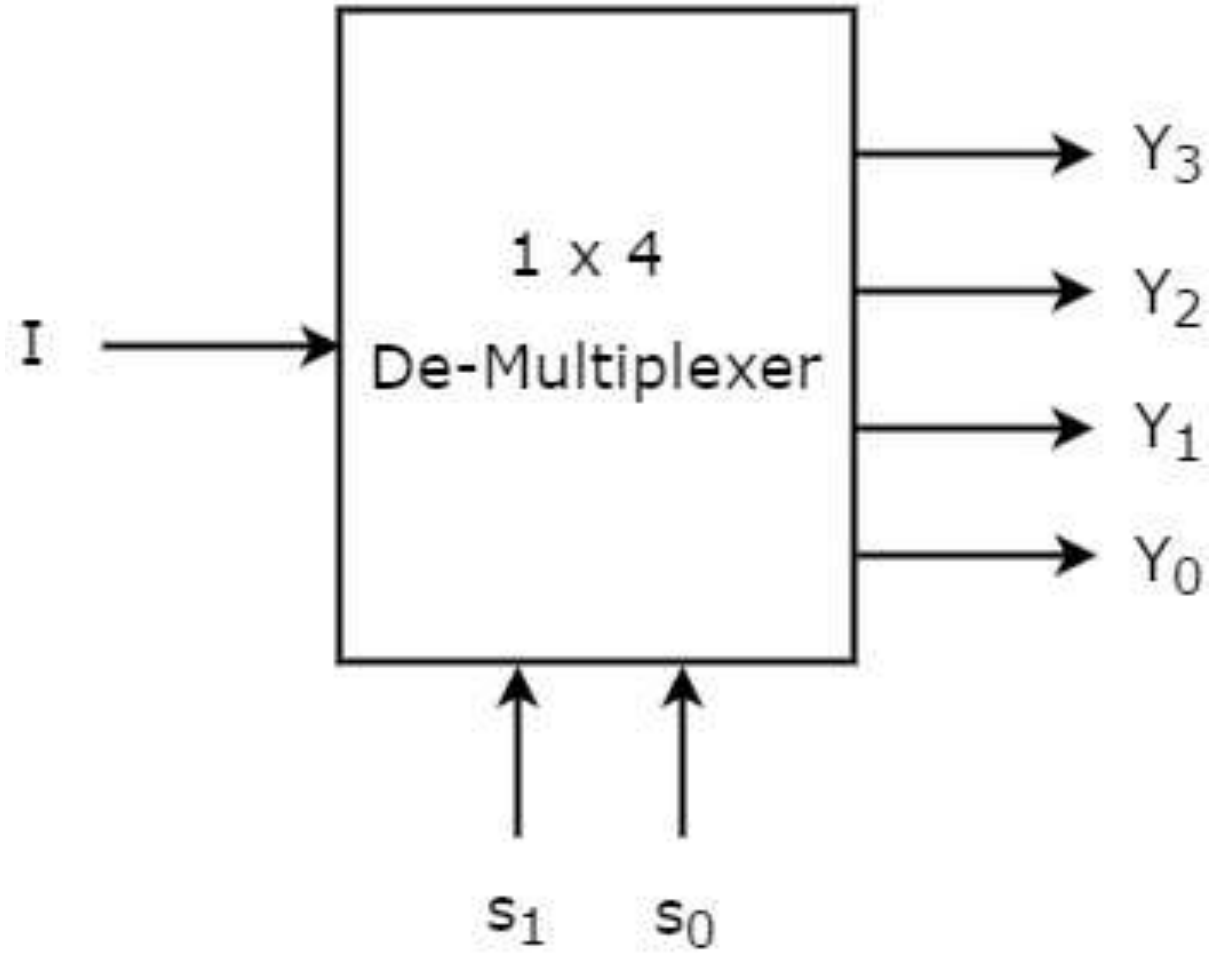
DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-7

DESIGN VERILOG PROGRAM TO IMPLEMENT TYPES OF DE-MULTIPLEXER.

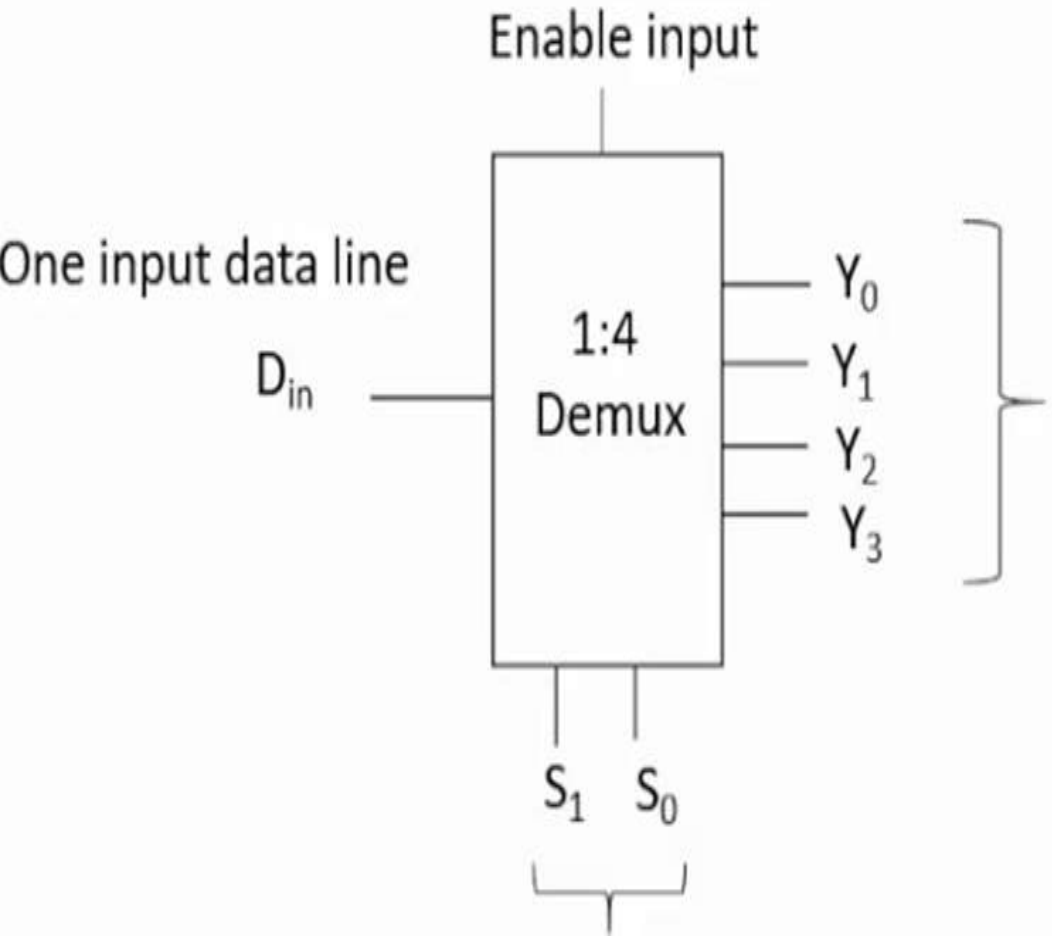
PROGRAM-7.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 1:4 DEMUX

Diagram No –(7.1).1: BLOCK DIAGRAM OF 1:4 DEMUX



PROGRAM-7.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 1:4 DEMUX

Diagram No –(7.1).2: TRUTH TABLE OF 1:4 DEMUX



Truth Table

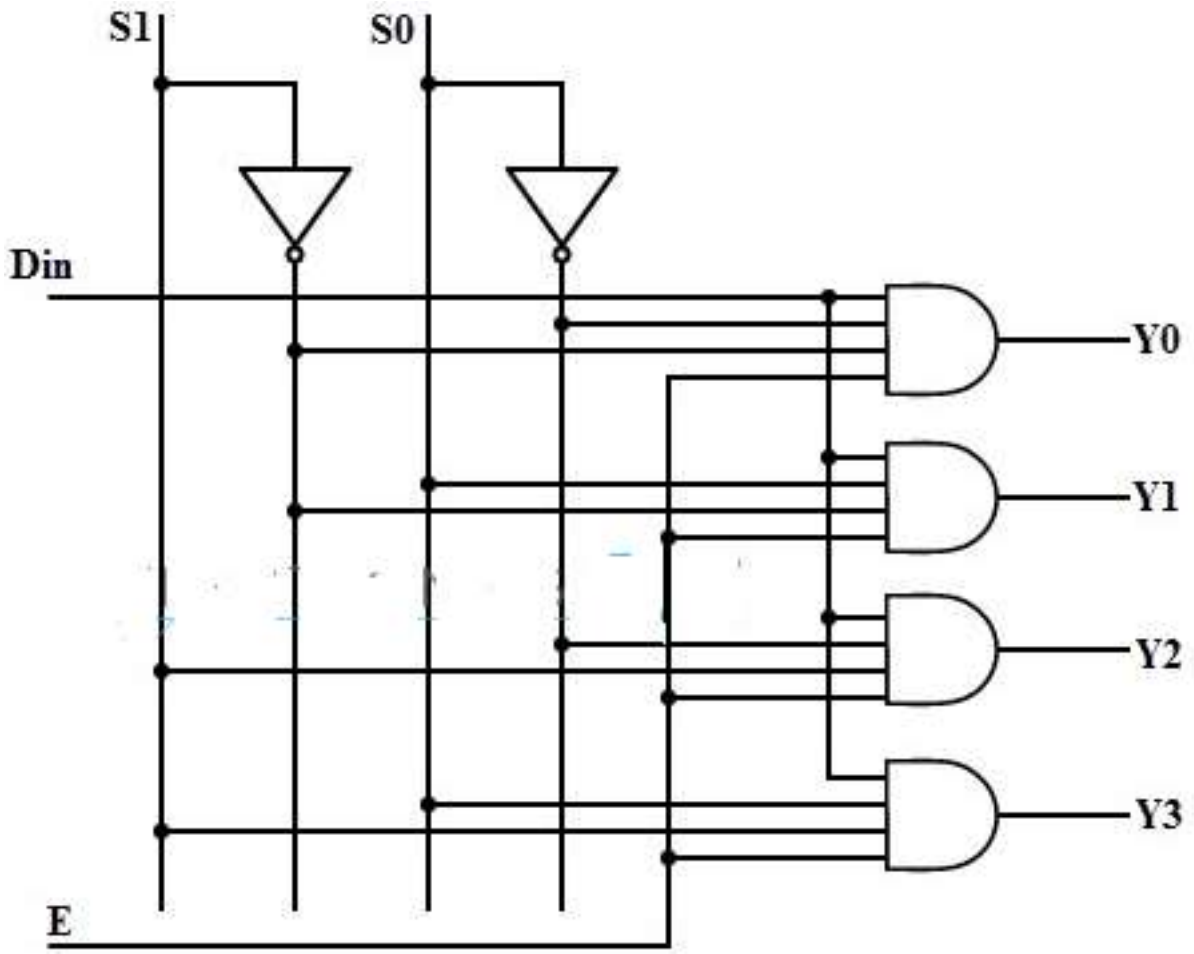
Inputs			Outputs			
E	S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	D_{in}
1	0	1	0	0	D_{in}	0
1	1	0	0	D_{in}	0	0
1	1	1	D_{in}	0	0	0

Disabled

Enabled

PROGRAM-7.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 1:4 DEMUX

Diagram No –(7.1).3: LOGICAL CIRCUIT DIAGRAM OF 1:4 DEMUX



PROGRAM-7.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 1:4 DEMUX

Diagram No –(7.1).4: LOGICAL EXPRESSIONS OF 1:4 DEMUX

LOGICAL EXPRESSION FOR 1:4 DEMUX CONSIDERING +VE ENABLE

Y0 = (~S1) & (~S0) & (I) & (En)
Y1 = (~S1) & (S0) & (I) & (En)
Y2 = (S1) & (~S0) & (I) & (En)
Y3 = (S1) & (S0) & (I) & (En)

LOGICAL EXPRESSION FOR 1:4 DEMUX CONSIDERING -VE ENABLE

Y0 = (~S1) & (~S0) & (I) & (~En)
Y1 = (~S1) & (S0) & (I) & (~En)
Y2 = (S1) & (~S0) & (I) & (~En)
Y3 = (S1) & (S0) & (I) & (~En)

PROGRAM-7.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 1:4 DEMUX

Circuit description Verilog Program example for implementing 1:4 DEMUX (dataflow model)

```
module demux(input s1,s0,I,en, output y3,y2,y1,y0);
        assign y0=(~s1)&(~s0)& (I) & (~en);
        assign y1=(~s1)& (s0) & (I) & (~en);
        assign y2=(s1) & (~s0) & (I) & (~en);
        assign y3=(s1) & (s0) & (I) & (~en);
 endmodule
```

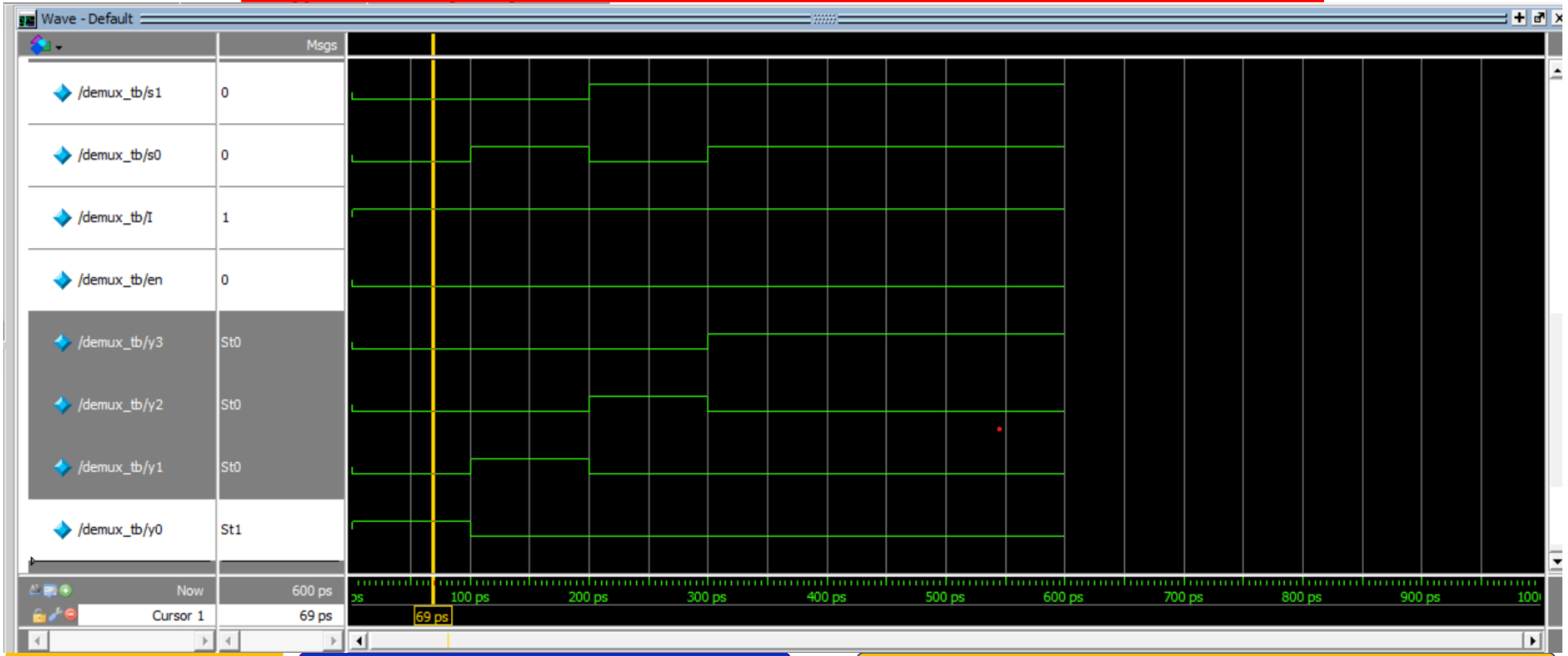
PROGRAM-7.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 1:4 DEMUX

Test bench waveform Verilog Program example for implementing 1:4 DEMUX (dataflow model)

```
module demux_tb;
        reg s1;reg s0; reg I;reg en;
        wire y3;wire y2; wire y1;wire y0;
        demux uut ( .s1(s1), .s0(s0), .I(I), .en(en), .y3(y3), .y2(y2),  .y1(y1), .y0(y0));
    initial
        begin
                s1 = 0;s0 = 0;I = 1;en = 0;      #100
                s1 = 0;s0 = 1;I = 1;en = 0;      #100
                s1 = 1;s0 = 0;I = 1;en = 0;      #100
                s1 = 1;s0 = 1;I = 1;en = 0;      #100
        end
endmodule
```

PROGRAM-7.1: DESIGN VERILOG PROGRAM TO IMPLEMENT 1:4 DEMUX

Diagram No –(7.1).5: DEMUX 1:4 VERILOG CODE OUTPUT SCREEN SHOT



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE SUBJECT: DIGITAL DESIGN & COMPUTER ORGANISATION LAB

COURSE CODE: BCS302

DETAILS OF COURSE COORDINATOR

Mr. PRASHANTH KUMAR. S. P

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

#541, BLOCK-3, SHAGYA(VILLAGE & POST), KOLLEGAL TALUK
CHAMARAJANAGAR DISTRICT, KARNATAKA STATE, INDIA - 571439
Email-ID: prashantheshwar2010@gmail.com Contact no: 9008929445

February 8, 2024

DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-8

DESIGN VERILOG PROGRAM FOR IMPLEMENTING VARIOUS TYPES OF FLIP-FLOPS SUCH AS SR, JK AND D.

DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-8.1

DESIGN VERILOG PROGRAM FOR IMPLEMENTING VARIOUS TYPES OF SR FLIP-FLOPS

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

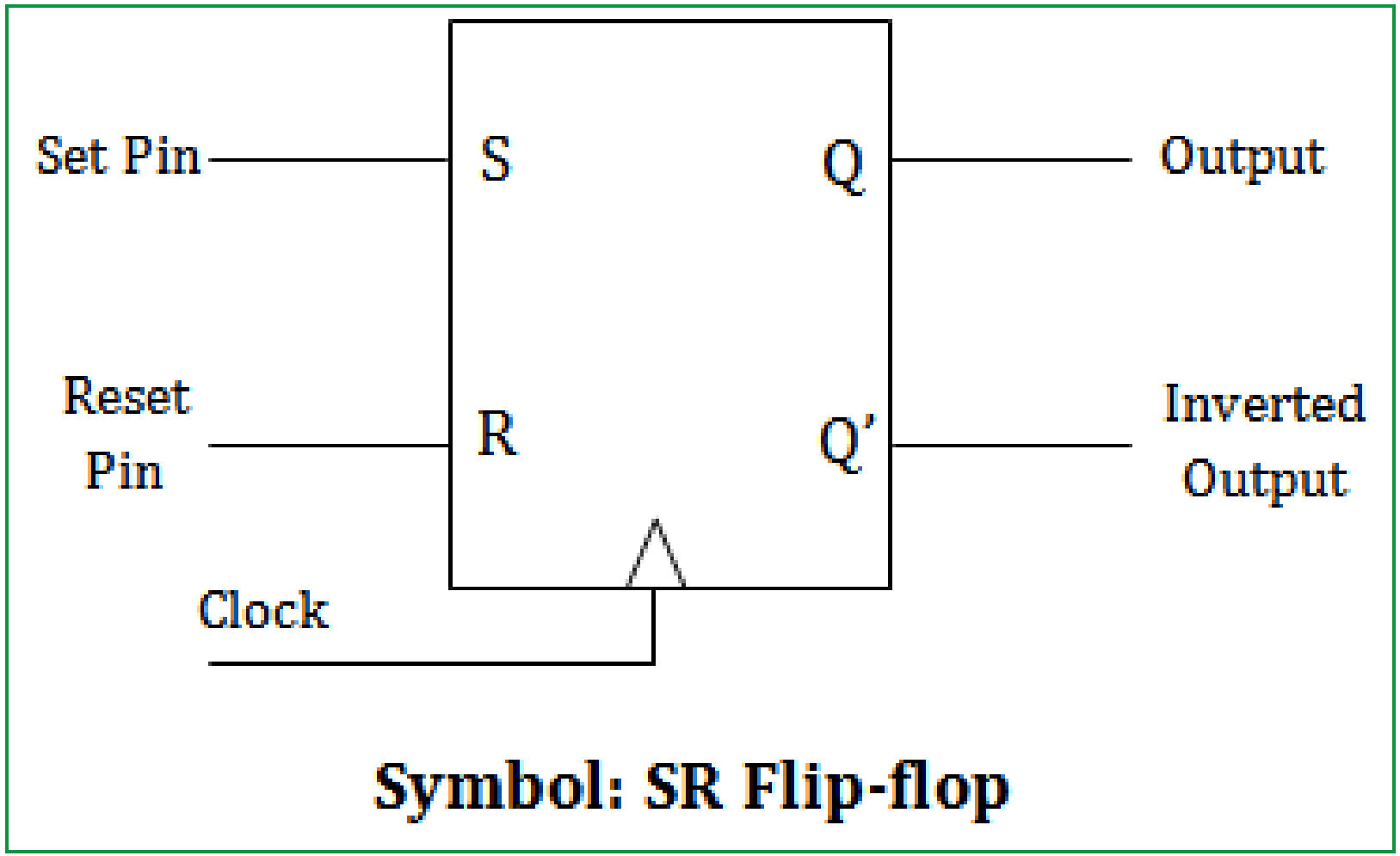
- **Truth Table:** A Truth table shows how a logic circuits output responds to various combination of inputs of it.

- **Characteristics Table:** A characteristic table defines the next state(Qn+1) of the flip-flop in terms of flip-flop input and current state (Qn).

- **Excitation Table:** Excitation table defines the flip-flop input variables as function of the current state(Qn) and next state(Qn+1).

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Diagram No –(8.1).1: SYMBOLIC REPRESENTATION/BLOCK DIAGRAM OF SR FLIP FLOP



Symbol: SR Flip-flop

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Diagram No –(8.1).2: TRUTH TABLE OF SR FLIP FLOP

CLK	S	R	Q	Q'
0	X	X	No Change	
1	0	0	No Change	
1	0	1	0	1
1	1	0	1	0
1	1	1	Invalid	

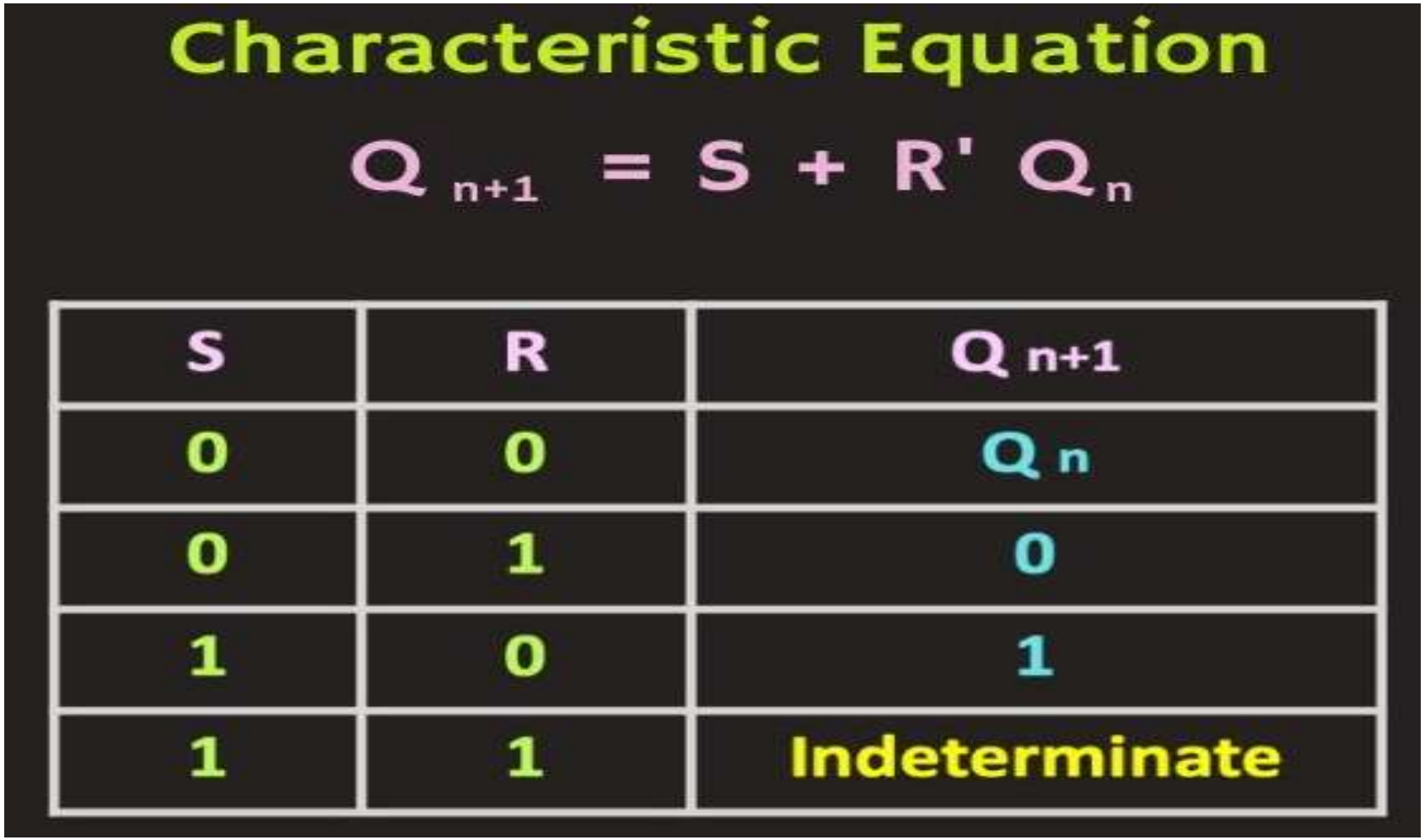
PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Diagram No –(8.1).3: TRUTH TABLE OF SR FLIP FLOP

Sno	S	R	Q	Q'	State
1	1	0	1	0	Q is set to 1
2	1	1	1	0	No change
3	0	1	0	1	Q' is set to 1
4	1	1	0	1	No change
5	0	0	1	1	Invalid

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Diagram No –(8.1).4: SR FLIP-FLOP CHARACTERISTIC EQUATION AND TABLE



S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Indeterminate

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Diagram No –(8.1).5: SR FLIP-FLOP CHARACTERISTIC EQUATION & TABLE AND EXCITATION TABLE

Characteristics Table

Qn	S	R	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	invalid
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	invalid

Excitation Table

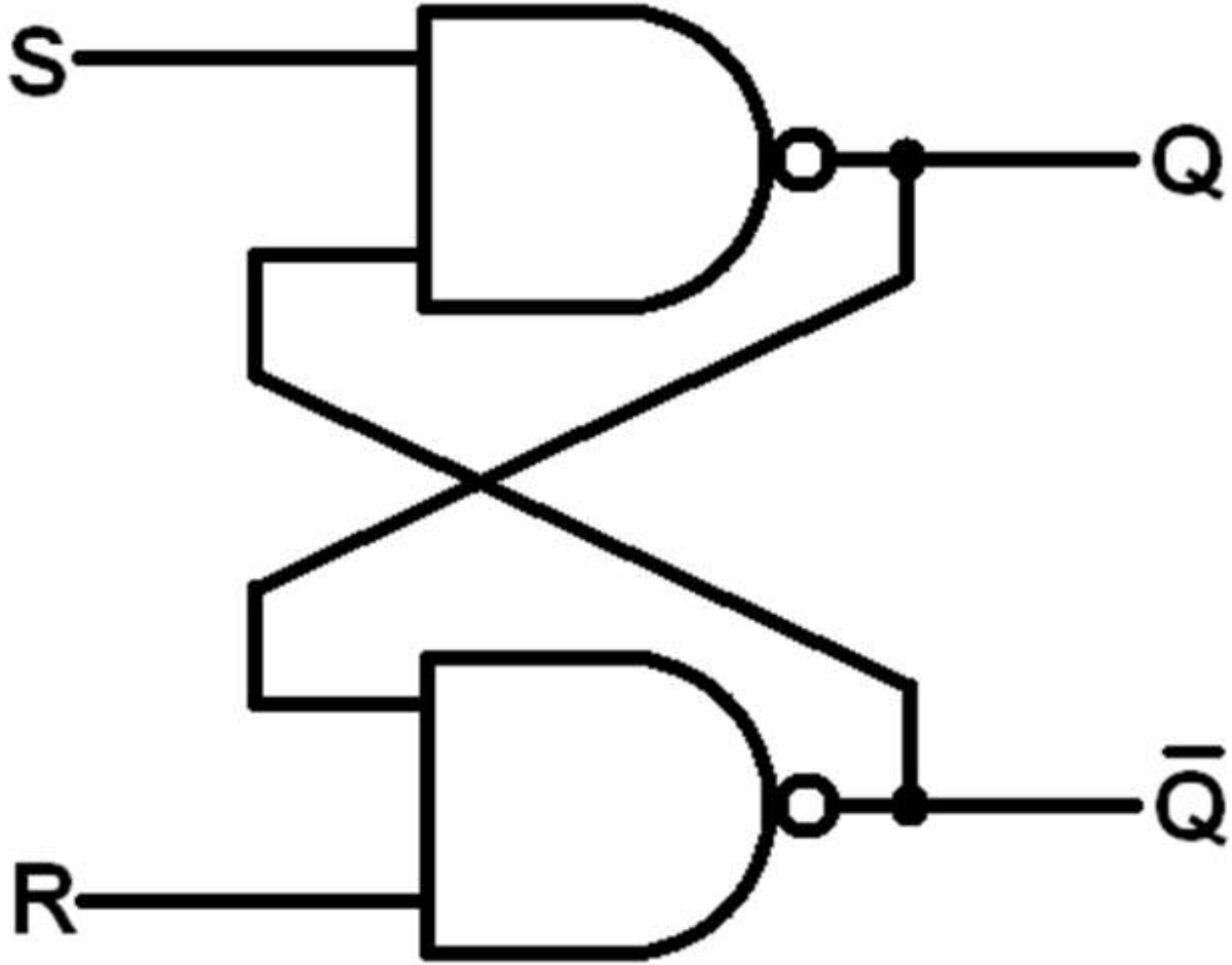
Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Qn+1 \ SR	00	01	11	10
0	0	1	X	1
1	1	0	X	1

$$Q_{n+1} = S + Q_n \overline{R}$$

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Diagram No –(8.1).6: SR FLIP-FLOP LOGICAL CIRCUIT DIAGRAM USING NAND GATE



PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Circuit description module Verilog Program example for implementing SR-FLIP FLOP

```
module srff(input clk,rst,input [1:0]SR,output reg  q, qb);
   always@(posedge clk or posedge rst)
          begin
                 if(rst==1)
                        begin
                               q=1'b0; qb=~q;
                        end
                 else
                        case (SR)
                        2'b01:
                               begin
                                      q = 1'b0;qb = 1'b1;
                               end
```

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Circuit description module Verilog Program example for implementing SR-FLIP FLOP

```
                2'b10:
                        begin
                                q = 1'b1;qb = 1'b0;
                        end
                2'b11:
                        begin
                                q = 1'b1;qb = 1'b1;
                        end
                default:
                        begin
                        end
                endcase
        end

endmodule
```

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

TEST BENCH module Verilog Program example for implementing SR-FLIP FLOP

```
module tb_SR;
  reg clk;reg rst;reg [1:0] SR;
  wire q;wire qb;

  srff uut (.clk(clk),.rst(rst),.SR(SR),.q(q),.qb(qb));

  initial
        begin
              clk = 0;
              rst=0;
              SR=2'b00;
        end
  always #5 clk=~clk;
```

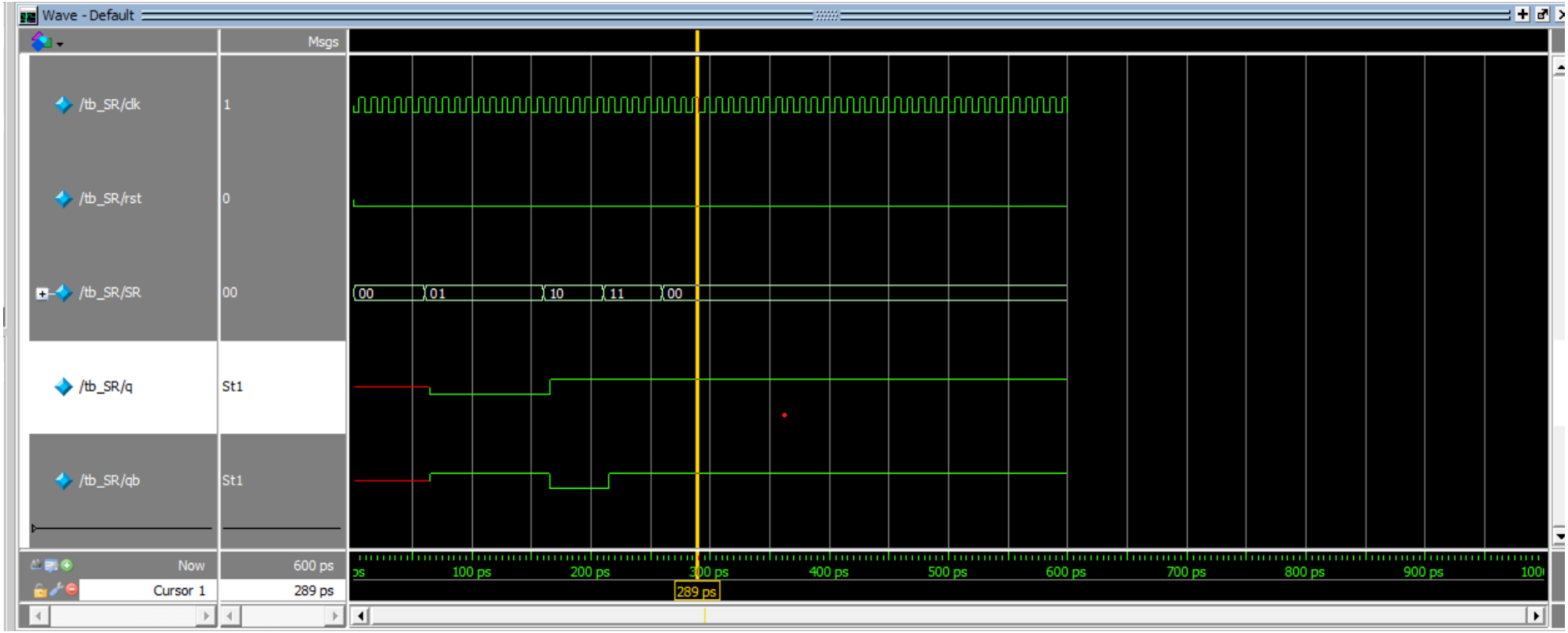
PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

TEST BENCH module Verilog Program example for implementing SR-FLIP FLOP

```
initial
        begin
                #10;
                #50 SR=2'b01;
                #50 SR=2'b01;
                #50 SR=2'b10;
                #50 SR=2'b11;
                #50 SR=2'b00;
        #50;
  end
endmodule
```

PROGRAM-8.1: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF SR-FLIP FLOP

Diagram No –(8.1).7: output screen shot of Verilog code for SR-FLIP FLOP



DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-8.2

DESIGN VERILOG PROGRAM FOR IMPLEMENTING VARIOUS TYPES OF JK FLIP-FLOPS

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

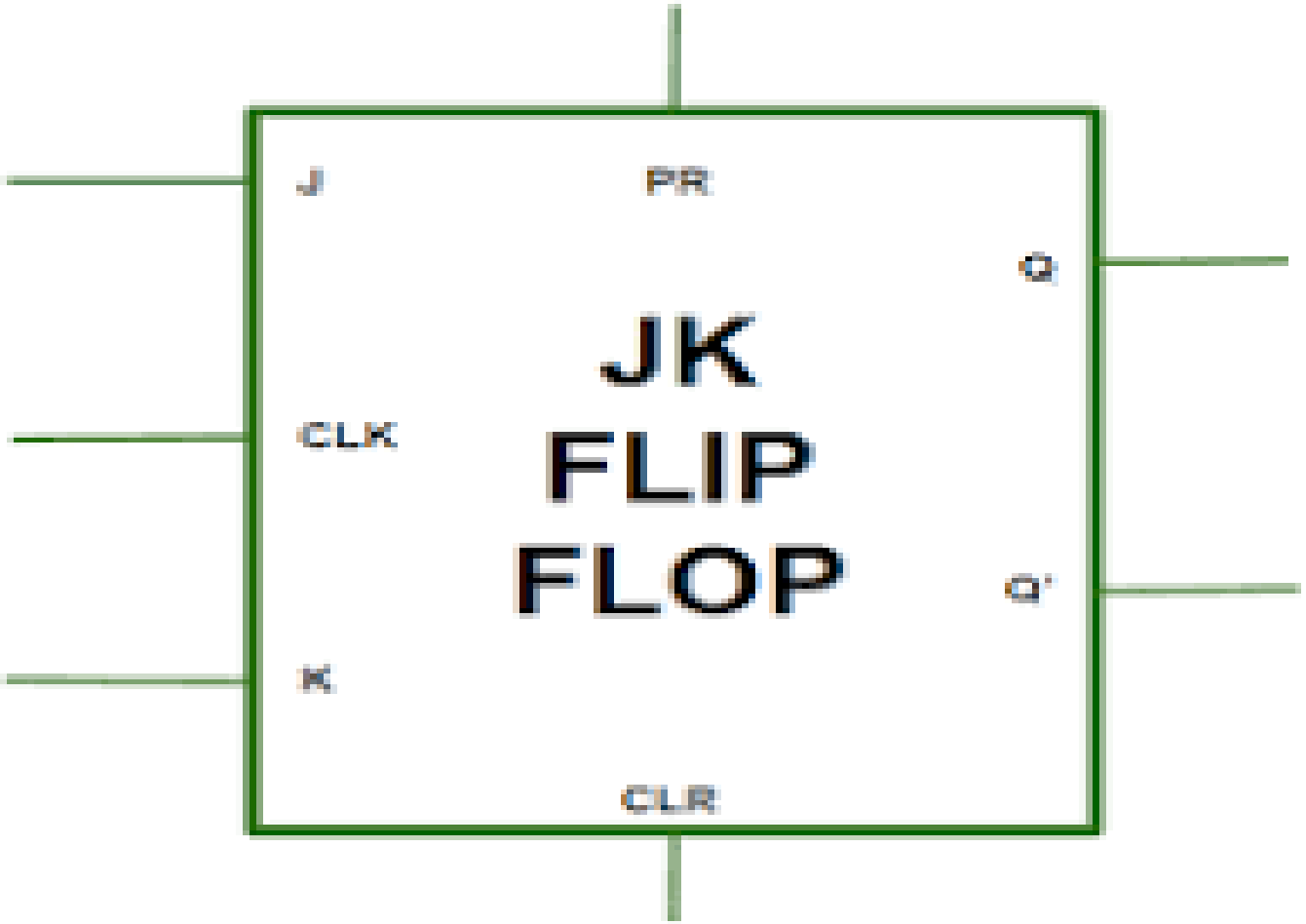
- **Truth Table:** A Truth table shows how a logic circuits output responds to various combination of inputs of it.

- **Characteristics Table:** A characteristic table defines the next state(Qn+1) of the flip-flop in terms of flip-flop input and current state (Qn).

- **Excitation Table:** Excitation table defines the flip-flop input variables as function of the current state(Qn) and next state(Qn+1).

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).1: BLOCK DIAGRAM/SYMBOLIC REPRESENTATION OF JK FLIP FLOP



PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).2: TRUTH TABLE OF JK FLIP FLOP

Truth Table

CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	Q_n'

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).3: DETAILED CHARACTERISTIC TABLE OF JK FLIP-FLOP

CLK	J	K	Q_n	Q_{n+1}	Q_{n+1}
0	X	X	0/1	0/1	Q_n
↑	0 0	0 0	0 1	0 1	Q_n
↑	0 0	1 1	0 1	0 0	0
↑	1 1	0 0	0 1	1 1	1
↑	1 1	1 1	0 1	1 0	Q_n'

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).4: CHARACTERSTIC TABLE OF JK-FLIP FLOP

CLK	J	K	Qn	Qn+1	Qn+1
1	0	0	0	0	Qn
1			1	1	
1	0	1	0	0	0
1			1	0	
1	1	0	0	1	1
1			1	1	
1	1	1	0	1	Qn'
1			1	0	

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).5: CHARACTERSTIC EQUATION OF JK-FLIP FLOP

CHARACTERISTIC EQUATION OF JK FLIP FLOP:

$$Qn+1 = (JQn') + (K'Qn)$$

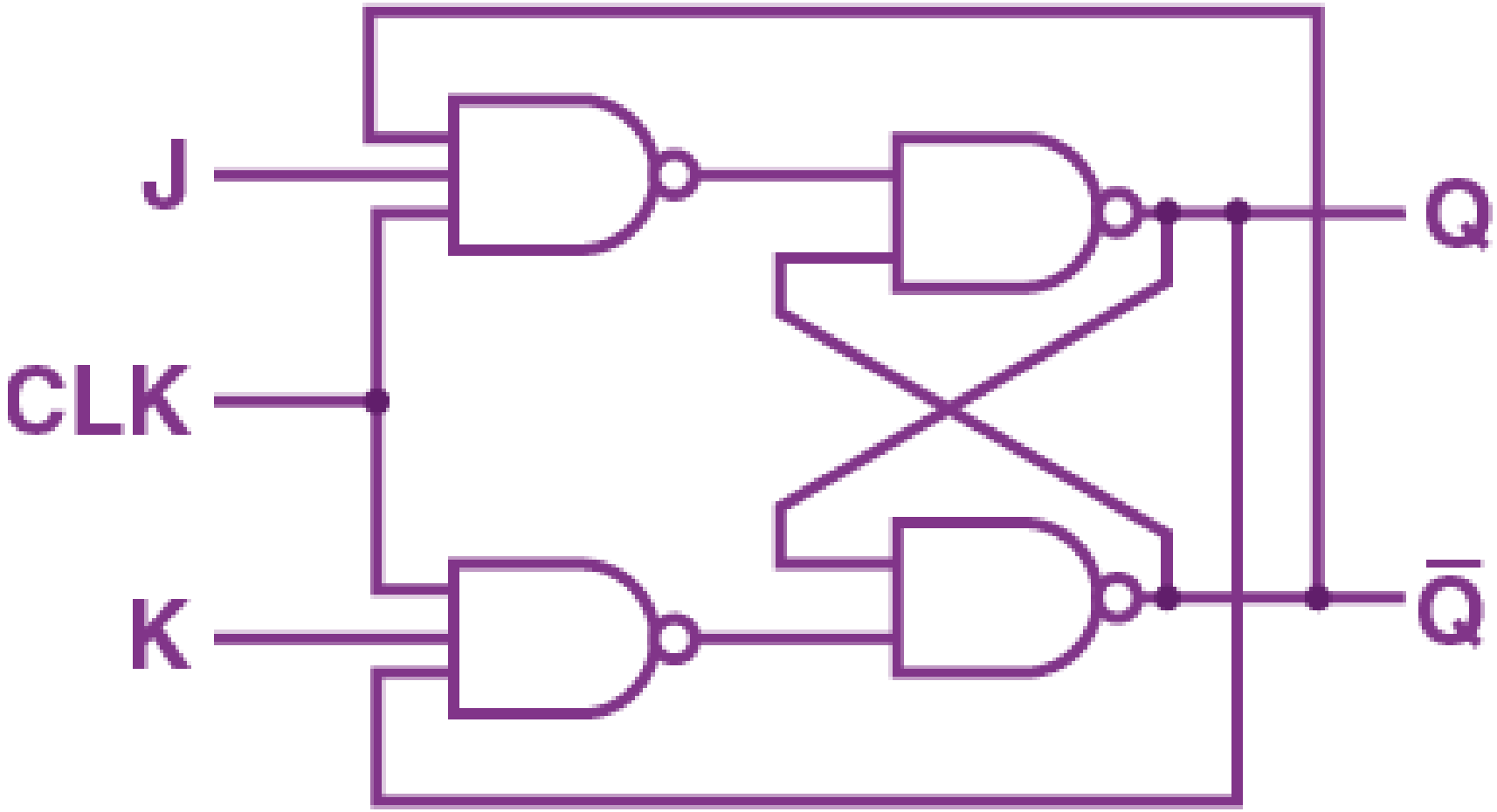
PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).6: EXCITATION TABLE OF JK-FLIP FLOP

Q_n	Q_{n+1}	J	K
0	0	0	X
1	0	X	1
0	1	1	X
1	1	X	0

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).7: LOGICAL CIRCUIT DIAGRAM OF JK FLIP-FLOP USING NAND GATE



PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Circuit description module Verilog Program example for implementing JK-FLIP FLOP

```
module jk_ff(input [1:0] jk,input rst,clk,output reg q,qb);
   always@(posedge clk,posedge rst)
          begin

                 if(rst==1)
                        q=1'b0;
                 else
                        case (jk)
                               2'b01:q=1'b0;
                               2'b10:q=1'b1;
                               2'b11:q=~q;
                               default: begin end
                        endcase
                        qb = ~q;
          end
endmodule
```

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

TEST BENCH module Verilog Program example for implementing JK-FLIP FLOP

```
module tb_jk;
  reg [1:0] jk;reg rst;reg clk;
  wire q;wire qb;
  jk_ff uut (.jk(jk), .rst(rst), .clk(clk),.q(q), .qb(qb));

  initial
      begin
          jk =2'b00;
          rst = 0;
          clk = 0;
      end
  always #5clk=~clk;
```

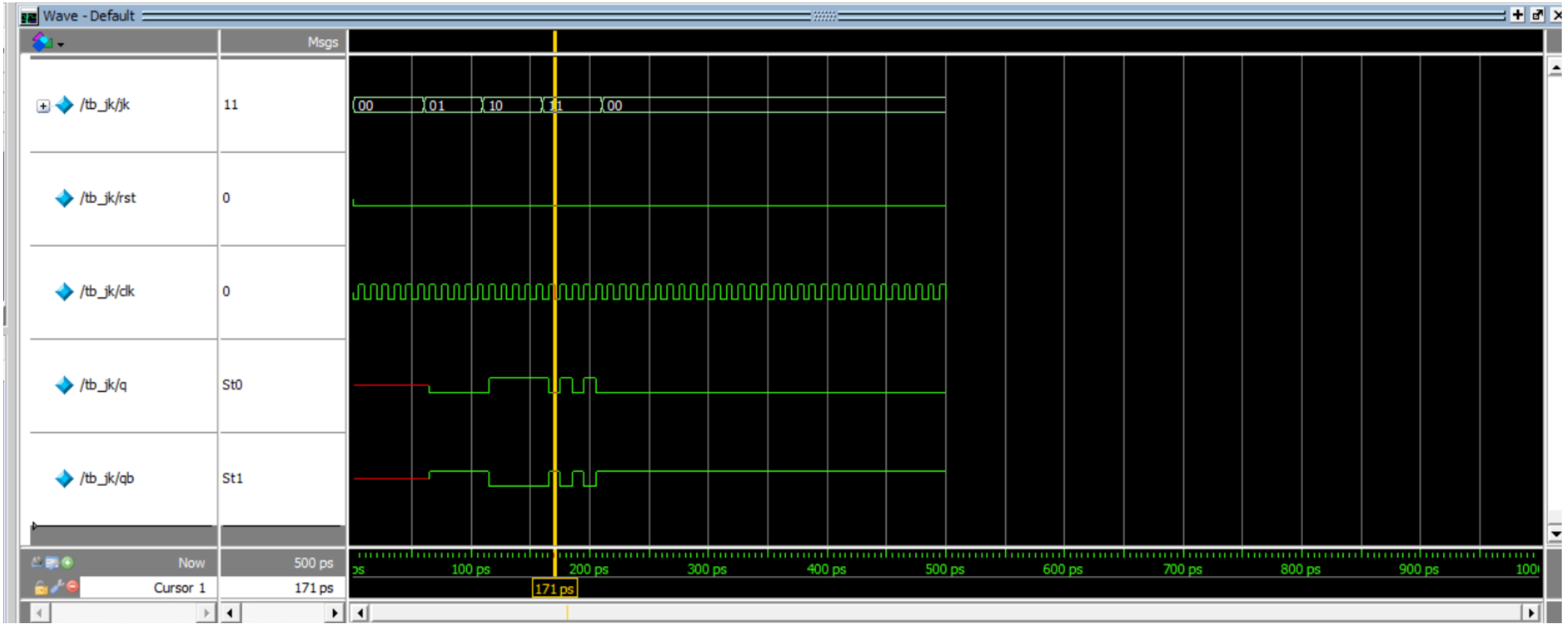
PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

TEST BENCH module Verilog Program example for implementing JK-FLIP FLOP

```
initial
        begin
                #10;
                #50 jk=2'b01;
                #50 jk=2'b10;
                #50 jk=2'b11;
                #50 jk=2'b00;
                #50;
        end
endmodule
```

PROGRAM-8.2: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF JK-FLIP FLOP

Diagram No –(8.2).8: output screen shot of the Verilog code for JK FLIP-FLOP



DIGITAL DESIGN AND COMPUTER ORGANISATION LAB(BCS302)

PROGRAM-8.3

DESIGN VERILOG PROGRAM FOR IMPLEMENTING VARIOUS TYPES OF D FLIP-FLOPS

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

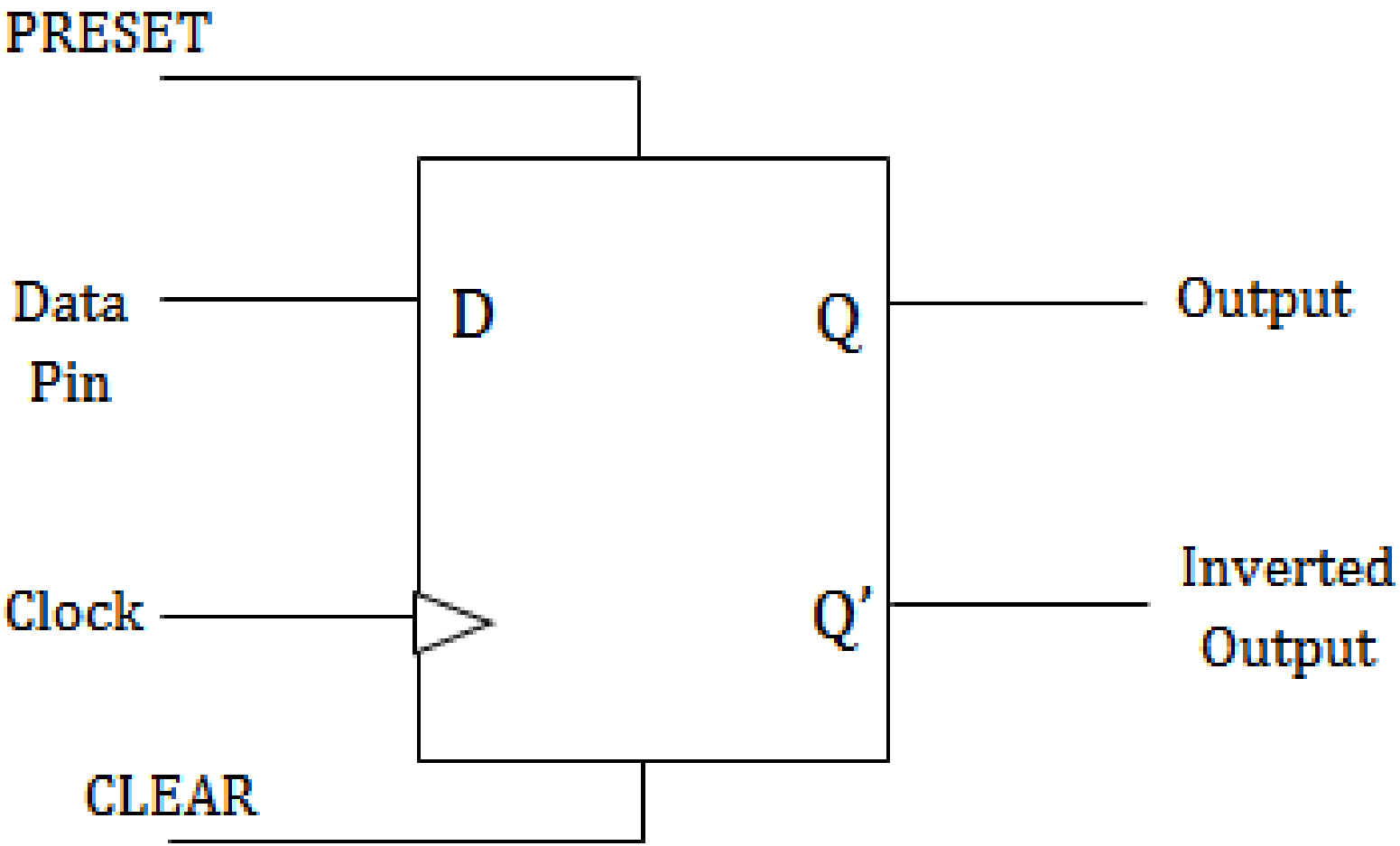
- **Truth Table:** A Truth table shows how a logic circuits output responds to various combination of inputs of it.

- **Characteristics Table:** A characteristic table defines the next state(Qn+1) of the flip-flop in terms of flip-flop input and current state (Qn).

- **Excitation Table:** Excitation table defines the flip-flop input variables as function of the current state(Qn) and next state(Qn+1).

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Diagram No –(8.3).1: BLOCK DIAGRAM/SYMBOLIC REPRESENTATION OF D FLIP-FLOP



PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Diagram No –(8.3).2: TRUTH TABLE OF D FLIP FLOP

D	Present state Q_n	Next state Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Truth table of D flip flop

D	CLK	$\bar{Q}$
0	1 (Raising Edge)	0
1	1 (Raising Edge)	1

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Diagram No –(8.3).3: DETAILED CHARACTERISTIC TABLE OF D-FLIP FLOP

Clk	Q(t–1)	D	Q(t)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Diagram No –(8.3).4: EXCITATION TABLE OF D-FLIP FLOP

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table of D flip flop

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

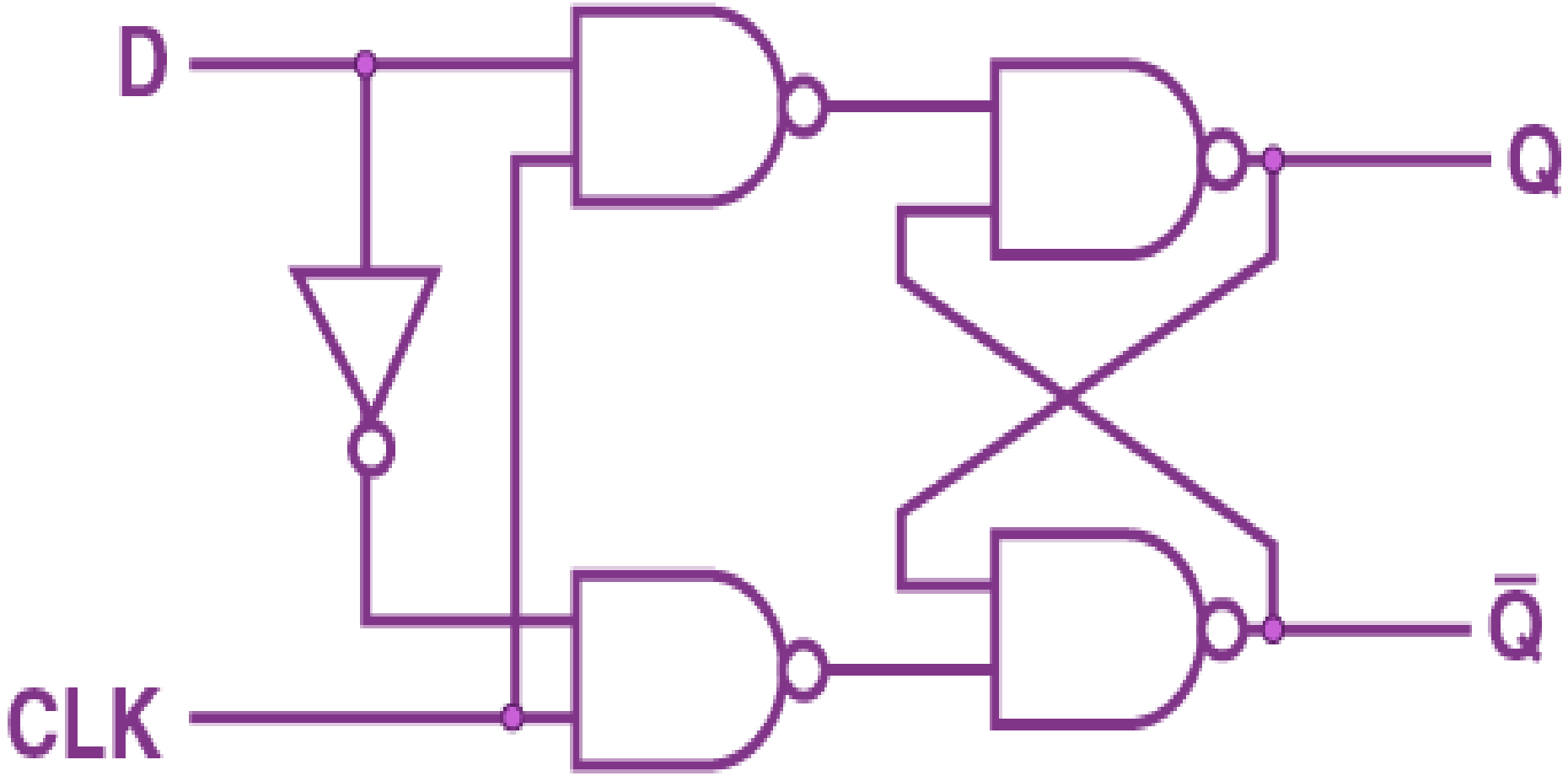
Diagram No –(8.3).5: CHARACTERISTIC EQUATION OF D-FLIP FLOP

CHARACTERSTIC EQUATION OF D-FLIP FLOP:

$$Qn+1 = D$$

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Diagram No –(8.3).6: LOGICAL CIRCUIT DIAGRAM OF D-FLIP FLOP USING NAND GATES



PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Circuit description module Program example for implementing D-flip flop

```
module dff1(input d,clk,rst,output reg q,qb);

  reg temp=0;

  always@(posedge clk,posedge rst)
        begin
              if (rst==0)
                    temp=d;

              else
                    temp=0;
```

Circuit description module Program example for implementing D-flip flop

```
              q = temp;
              qb = ~temp ;
        end
endmodule
```

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Test bench module Program example for implementing D-flip flop

```
module tb_dff;
   reg d;reg clk;reg rst;
   wire q;          wire qb;

   dff1 uut (.d(d), .clk(clk), .rst(rst), .q(q), .qb(qb));

   initial
         begin
               rst = 0; clk = 0;d=0;
         end

         always #25 clk = ~clk;
         always #500 rst=~rst;
         always #75 d=~d;
endmodule
```

PROGRAM-8.3: DESIGN OF VERILOG CODE FOR IMPLMENTATION OF D-FLIP FLOP

Diagram No –(8.3).7: output screen shot of the Verilog code for D-FLIP FLOP