R.V. COLLEGE OF ENGINEERING®, BENGALURU

(AUTONOMOUS INSTITUTION AFFILIATED TO VTU, BELAGAVI)
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

Design of Testing and Testability (18EC7F3)

Experimental Learning report on

Complete BIST design for Full Adder

Team Members	USN
Nihar KM	**1RV18EC097**
Sandesh Padiyar	**1RV18EC141**

<u>Under the Guidance of:</u>

Mrs. Namita Palecha

Assistant Professor

Department of Electronics and Communication Engineering

R.V College of Engineering, Bengaluru - 560059

Index

Introduction

The basic idea of BIST, in its most simple form, is to design a circuit so that the circuit can test itself and determine whether it is “good” or “bad” (fault-free or faulty, respectively). This typically requires that additional circuitry and functionality be incorporated into the design of the circuit to facilitate the self-testing feature. This additional functionality must be capable of generating test patterns as well as providing a mechanism to determine if the output responses of the circuit under test (CUT - in our case a 1-bit full adder) to the test patterns correspond to that of a fault-free circuit.

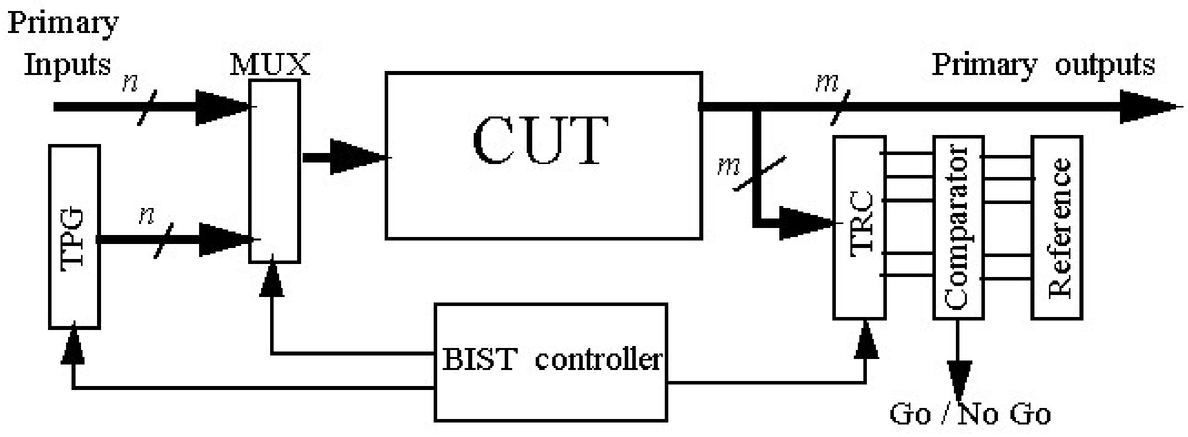


Figure 1.1 BIST Architecture

A representative architecture of the BIST circuitry as it might be incorporated into the CUT is illustrated in the block diagram of Figure 1.1. The two essential functions include the test pattern generator (TPG) and output response analyzer (ORA). While the TPG produces a sequence of patterns for testing the CUT, the ORA compacts the output responses of the CUT into some type of Pass/Fail indication. The BIST controller is the top module generating control signals. When test mode is 1 the golden Signature is compared with the output of the MISR and determine whether it matches with the golden signature stored. Based on this a pass/fail output would be generated.

PROBLEM STATEMENT

Design and simulation of 1 bit Full Adder with complete BIST functionality.

Specifications :

4bit MISR for Output Response Analyzer (ORA).

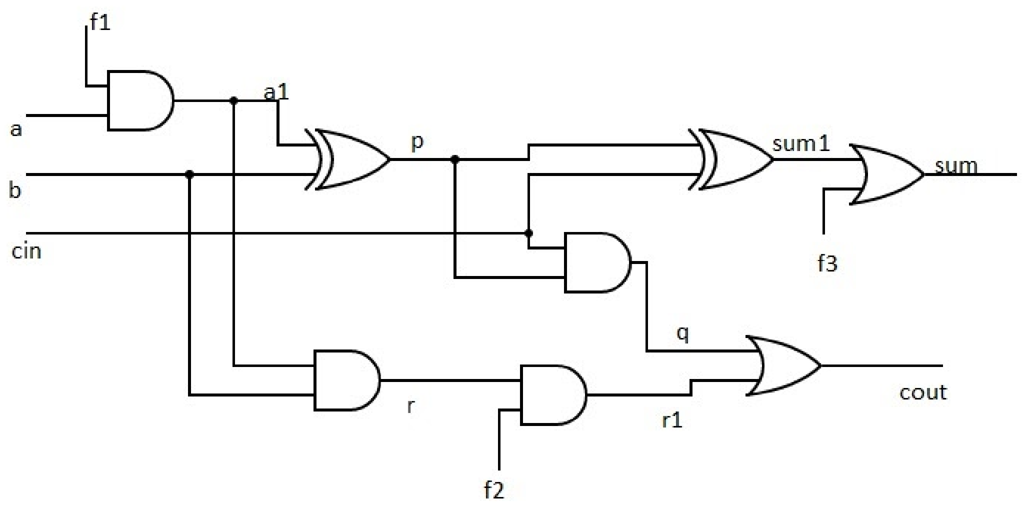
3bit LFSR for Test Pattern Generation.

1 bit Full Adder CUT

Design of BIST controller

CIRCUIT UNDER TEST (CUT)

The CUT is a 1- bit full adder. Faults have been injected at 3 points in the design i.e a@0, sum@1, r@0. The response from the MISR is compared with the golden signature to check whether the fault is detected or goes undetected.



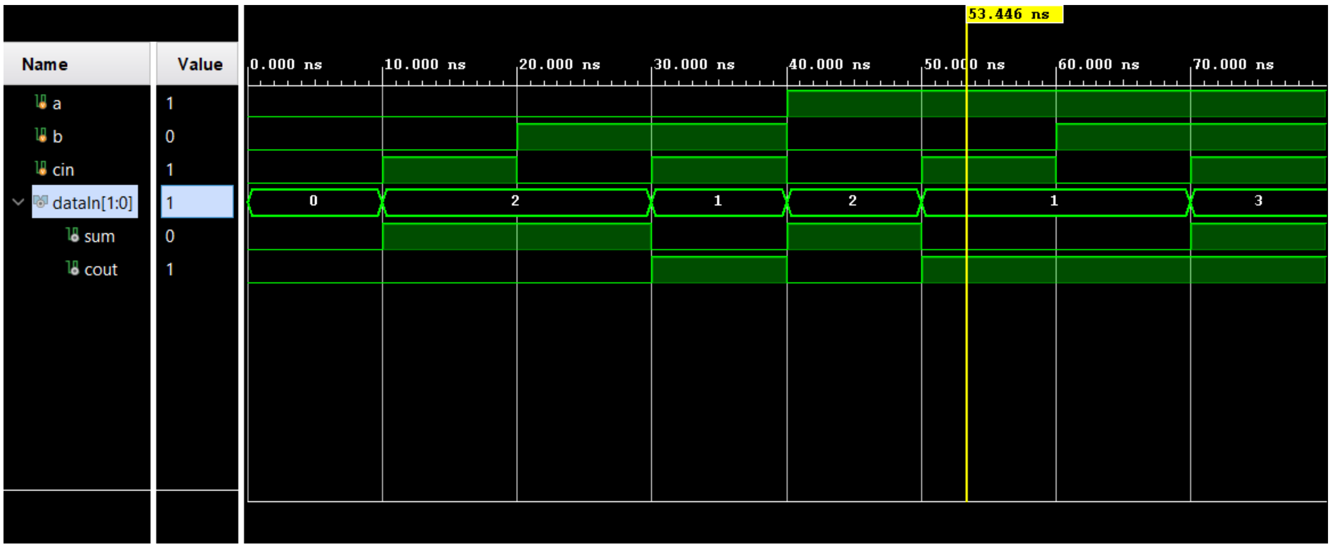
CODE:

```
module full_adder(
   input a,
   input b,
   input cin,
   output[1:0] dataIn );
parameter f_1 = 1'b0;  // a stuck at 0
parameter f_2 = 1'b0;  // r stuck at 0
parameter f_3 = 1'b0;  // sum stuck at 1
wire p,q,r,a11,r1,sum1,sum,cout;
and f1(a11,~f_1,a);
xor x1(p,a11,b);
and a1(r,a11,b);
xor x2(sum1,p,cin);
or f3(sum,sum1,f_3);
and a2(q,p,cin);
and f2(r1,r,~f_2);
or o1(cout,q,r1);
assign dataIn = {sum,cout};
endmodule
```

TEST BENCH:

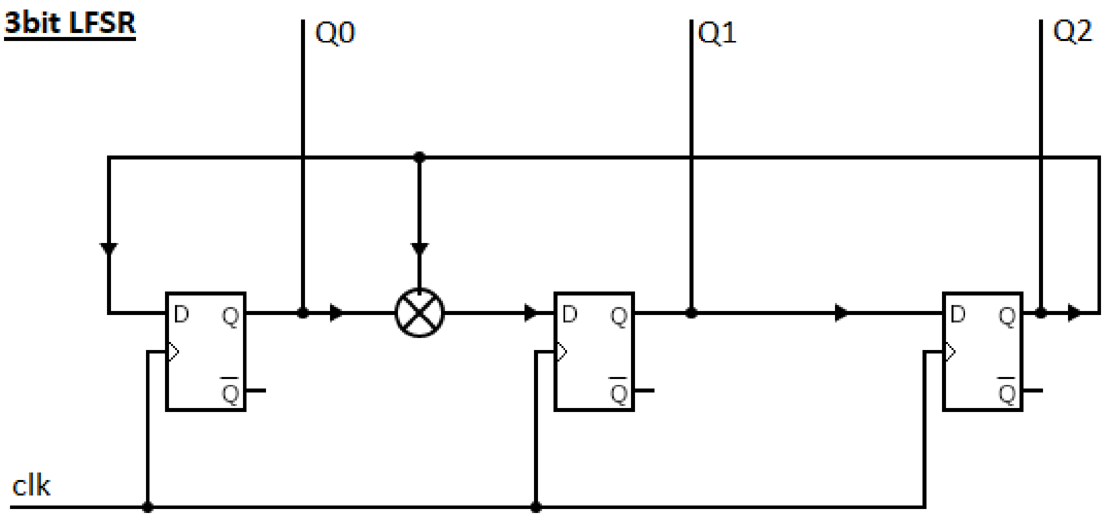
```
module CUT_tb;
reg a,b,cin;
wire sum,cout;

full_adder dut(a,b,cin,sum,cout);

initial begin
a=1'b0;b=1'b0;cin=1'b0;
#10 a=1'b0;b=1'b0;cin=1'b1;
#10 a=1'b0;b=1'b1;cin=1'b0;
#10 a=1'b0;b=1'b1;cin=1'b1;
#10 a=1'b1;b=1'b0;cin=1'b0;
#10 a=1'b1;b=1'b0;cin=1'b1;
#10 a=1'b1;b=1'b1;cin=1'b0;
#10 a=1'b1;b=1'b1;cin=1'b1;
end
initial begin #80 $finish; end
endmodule
```

SIMULATION RESULTS:



TEST PATTERN GENERATOR (TPG) - 3 BIT LFSR USING PRIMITIVE POLYNOMIAL

Polynomial Equation : $1 + x + x^3$



The LFSR has been seeded to 001. The LFSR has period of 7 (= $2^3 - 1$)

Next State Equations:

$Q0^* = Q2$

$Q1^* = Q0 + Q2$

$Q2^* = Q1$

Test Patterns generated

Q0	Q1	Q2
0	0	1
1	1	0
0	1	1
1	1	1
1	0	1
1	0	0
0	1	0

VERILOG CODE:

```
module ML_lfsr(data_out, complete, reset, clock);

input reset;

input clock;

output [0:2] data_out;

output reg  complete;

reg [0:2] lfsr_reg;

reg [2:0] counter;


always@(posedge clock or posedge reset)

begin

  if(reset == 1)

begin

lfsr_reg <= 9'b001;

counter <= 3'b000;

end

  else

begin
```

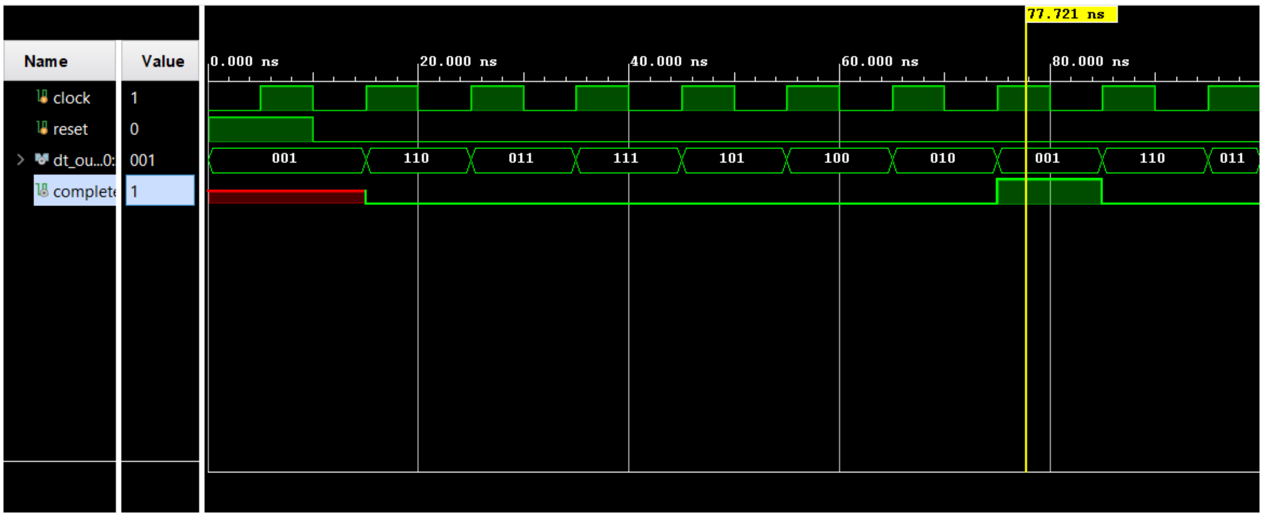
```
    lfsr_reg[0] <= lfsr_reg[2];
    lfsr_reg[1] <= lfsr_reg[0] ^ lfsr_reg[2];
    lfsr_reg[2] <= lfsr_reg[1];
counter = counter + 1;
if(counter <= 3'b110)
     complete = 0;
else
     complete = 1;
end
   end
assign data_out = lfsr_reg;
endmodule
```

TEST BENCH:

```
module tb_ML_lfsr;
reg clock,reset;
wire [0:2] dt_out;
wire [0:2] counter;
ML_lfsr lfsr1(dt_out, complete, reset, clock);
initial
begin
clock = 1'b0;
end
always
#5 clock = ~clock;
initial
begin
reset = 1'b1;
#10 reset = 1'b0;
end
```

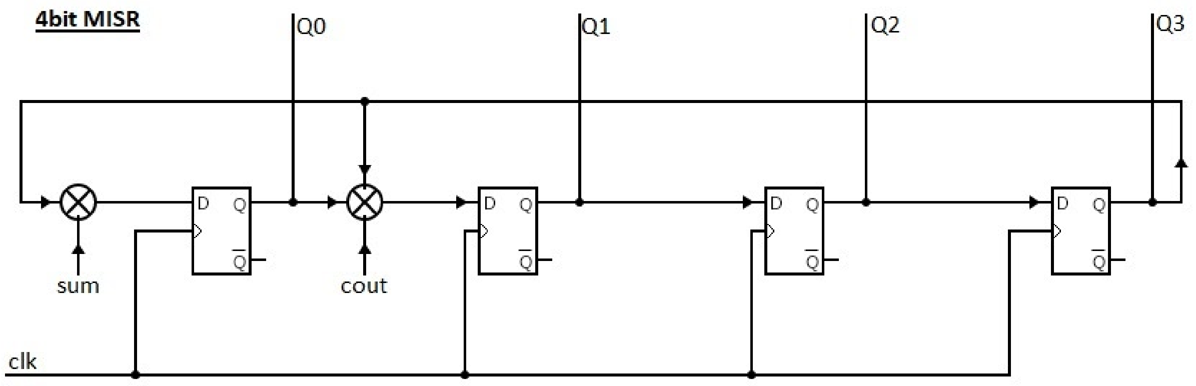
```
always #100 $finish;

endmodule
```

SIMULATION RESULTS:



<u>OUTPUT RESPONSE ANALYZER (ORA)</u> - 4BIT MISR

Polynomial Equation : $1 + x + x^4$



NEXT STATE EQUATIONS

$Q0^* = Sum + Q3$

$Q1^* = Cout + Q1 + Q3$

$Q2^* = Q1$

$Q3^* = Q2$

VERILOG CODE:

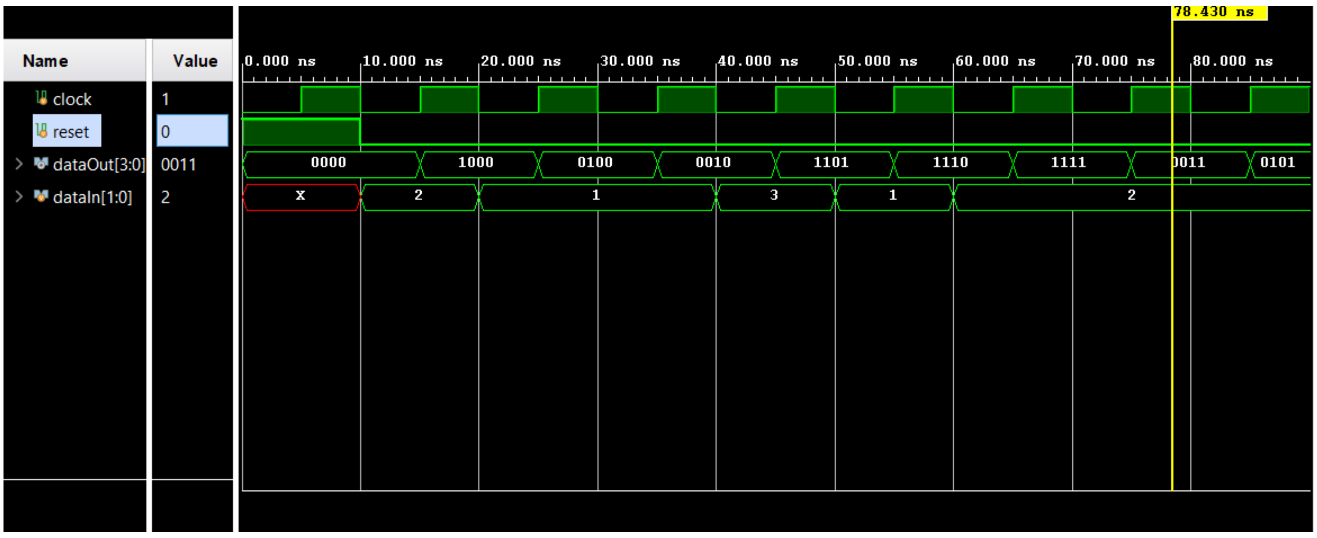
```
module MISR_4bit(dataIn,reset,clock,dataOut);

input [0:1] dataIn;

input reset,clock;

output reg [0:3] dataOut;

always@(posedge clock or posedge reset)

begin

   if(reset == 1)

dataOut <= 4'b0000;

else

  begin

dataOut[0] <= dataOut[3] ^ dataIn[0];

dataOut[1] <= dataOut[3] ^ dataOut[1] ^ dataIn[1];

dataOut[2] <= dataOut[1] ;

dataOut[3] <= dataOut[2] ;

 end

end

endmodule
```

TEST BENCH:

```
module MISR_4bit_tb;

reg clock,reset;

wire [3:0] dataOut;

reg [1:0] dataIn;

MISR_4bit dut(.dataIn(dataIn),.reset(reset),.clock(clock),.dataOut(dataOut));

initial

begin

clock = 1'b0;

end

always
```

```
#5 clock = ~clock;
initial
begin
reset = 1'b1;
#10 reset = 1'b0;
dataIn = 2'b10;
#10 dataIn = 2'b01;
#10 dataIn = 2'b01;
#10 dataIn = 2'b11;
#10 dataIn = 2'b01;
#10 dataIn = 2'b10;
#10 dataIn = 2'b10;
#20 $finish;
end
endmodule
```

SIMULATION RESULTS:



BIST CONTROLLER DESIGN

VERILOG CODE:

```
module controller_new(clock,reset,w,x,y,data_out,dataIn ,dataOut,testmode, fault_detected);


input clock,reset,testmode;

input  w,x,y;            // a,b and cin

output [2:0] data_out;    // lfsr output

output [1:0] dataIn;      // Misr input

output [3:0] dataOut;     // Misr output

output reg fault_detected;

wire  a,b,cin;

wire finish;

parameter golden_signature = 4'b0011;

assign {a,b,cin} = (testmode == 1)? (data_out[2:0]) : ({w,x,y});


ML_lfsr I1(data_out, finish, reset, clock);

full_adder I2(a,b,cin,dataIn);   //dataIn is {sum,cout} of full adder

MISR_4bit I3(dataIn, reset, clock,dataOut);


always @(posedge clock)

begin

        if(testmode == 0)          //when bist mode is not on, faults can't be detected

        fault_detected = 0;

else

begin

        if(finish == 1)

        begin

                if(golden_signature == dataOut)

                fault_detected = 0;

                else
                fault_detected = 1;
end
end
end
endmodule
```

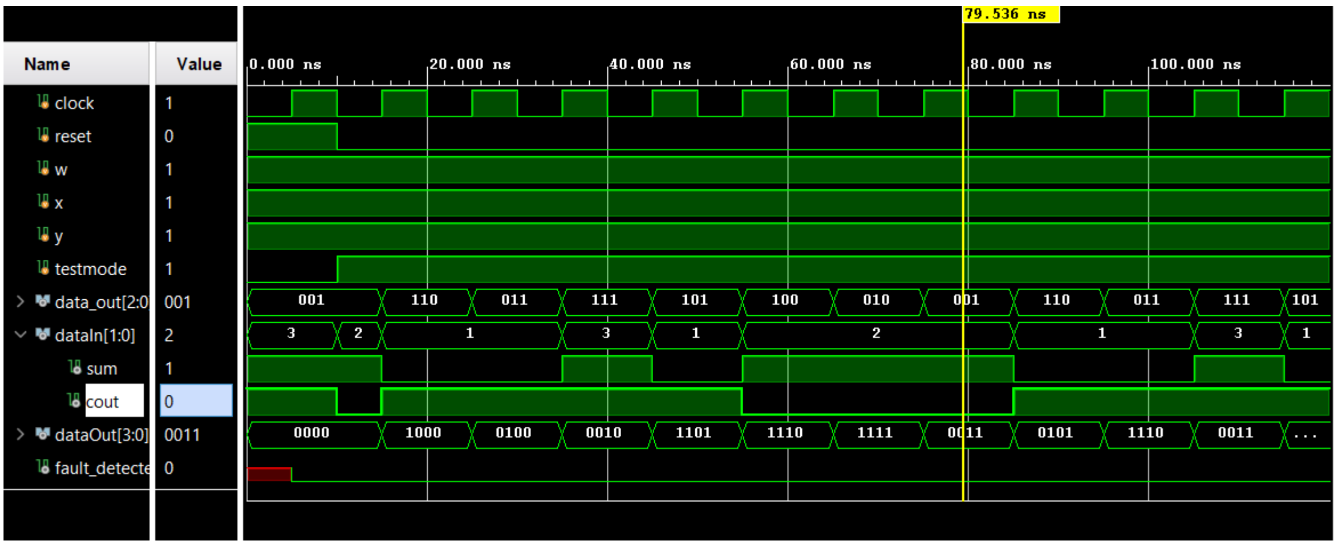
TEST BENCH:

```
module tb_controller_new;
        // Inputs
        reg clock;
        reg reset;
        reg w,x,y;
        reg testmode;

        // Outputs
        wire [2:0] data_out;
        wire [1:0] dataIn;
        wire [3:0] dataOut;



        controller_new uut (
                clock,reset,w,x,y,data_out,dataIn ,dataOut,testmode, fault_detected);
        initial begin
                clock = 0;
                reset = 1;
                w = 1; x=1; y=1;
                testmode = 0;
        end
                always #5 clock = !clock;
                initial begin
```

```
            #10 reset =1'b0;
            testmode =1'b1;
            end
            always #120 $finish;
endmodule
```

OUTPUT WAVEFORM FOR FAULT FREE OPERATION:

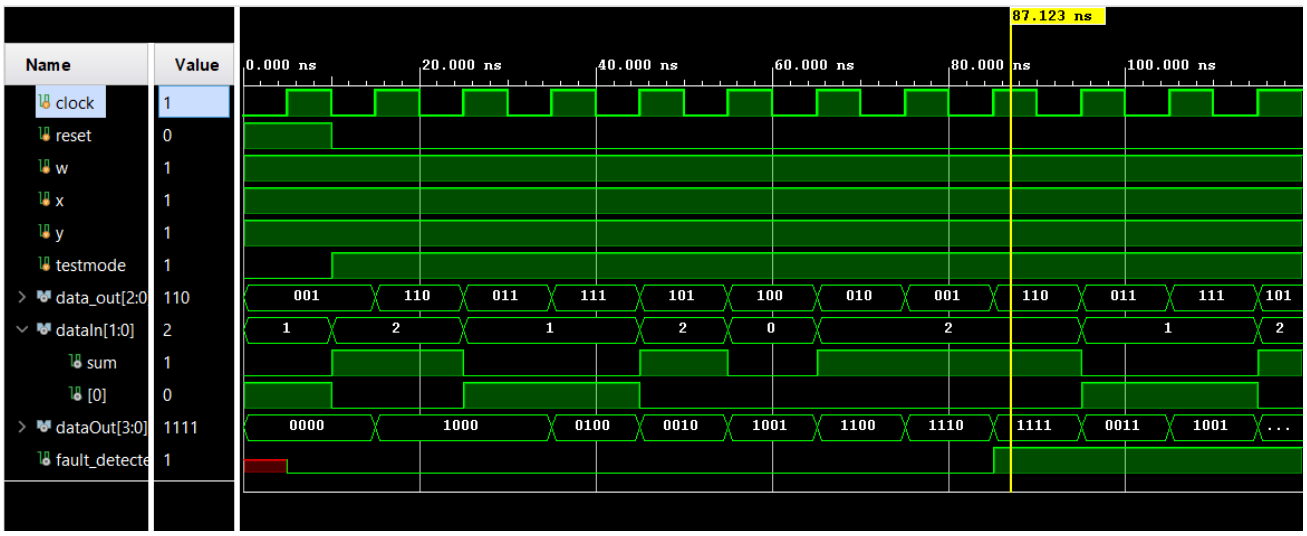
Sum	Cout	Q0	Q1	Q2	Q3
1	0	0	0	0	0
0	1	1	0	0	0
0	1	0	1	0	0
1	1	0	0	1	0
0	1	1	1	0	1
1	0	1	1	1	0
1	0	1	1	1	1
Golden Signature		0	0	1	1



OUTPUT WAVEFORM FOR FAULT a@0:

Sum	Cout	Q0	Q1	Q2	Q3
1	0	0	0	0	0
0	1	1	0	0	0
0	1	1	0	0	0
1	1	0	1	0	0
0	1	0	0	1	0
1	0	1	0	0	1
1	0	1	1	0	0
Faulty Signature (a@0)		1	1	1	0

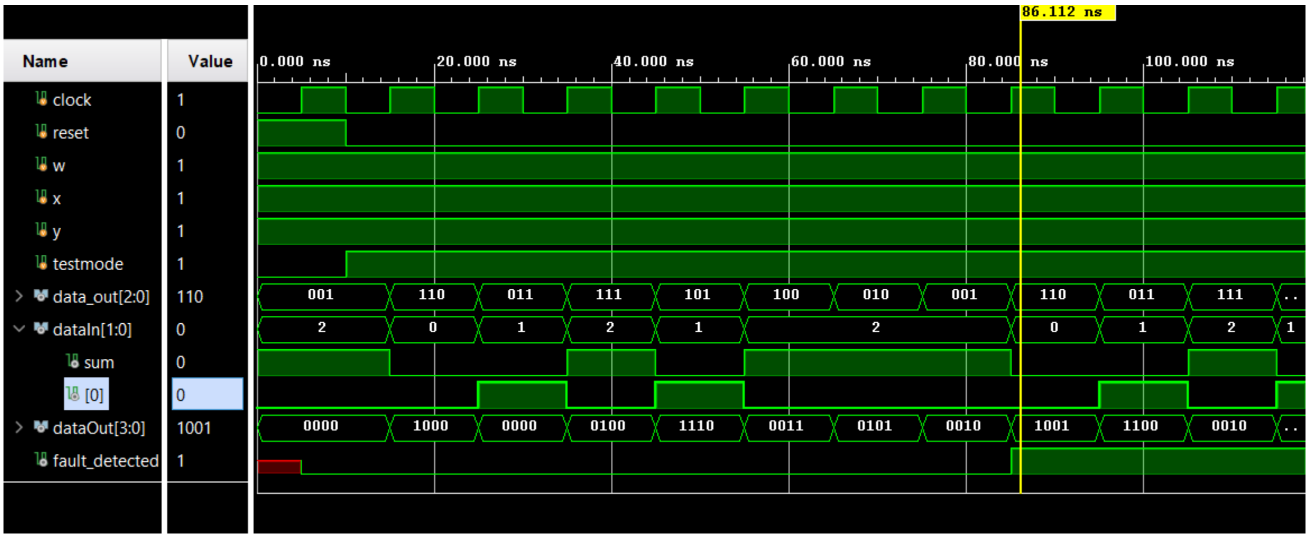
The faulty signature 1110 is different from the golden signature 0011 and hence the fault **a@0 is getting detected.**



OUTPUT WAVEFORM FOR FAULT r@0:

Sum	Cout	Q0	Q1	Q2	Q3
1	0	0	0	0	0
0	1	1	0	0	0
0	1	0	0	0	0
1	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	0	1
Faulty Signature (r@0)		0	0	1	0

The faulty signature 0010 is different from the golden signature 0011 and hence the fault **r@0 is getting detected.**

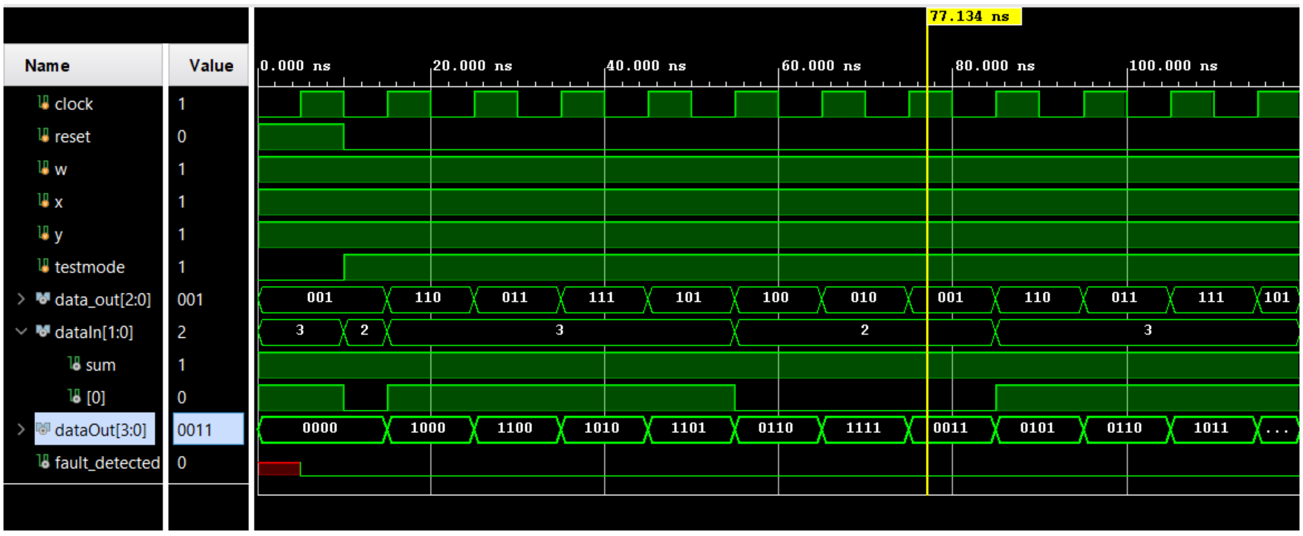


Name	Value
clock	1
reset	0
w	1
x	1
y	1
testmode	1
data_out[2:0]	110
dataIn[1:0]	0
sum	0
[0]	0
dataOut[3:0]	1001
fault_detected	1

OUTPUT WAVEFORM FOR FAULT sum@1:

Sum	Cout	Q0	Q1	Q2	Q3
1	0	0	0	0	0
1	1	1	0	0	0
1	1	1	1	0	0
1	1	1	0	1	0
1	1	1	1	0	1
1	0	0	1	1	0
1	0	1	1	1	1
Faulty Signature (sum@1)		**0**	**0**	**1**	**1**

The **faulty signature is matching with the golden signature 0011** and hence the fault **sum@1 is undetected**.



Name	Value
clock	1
reset	0
w	1
x	1
y	1
testmode	1
data_out[2:0]	001
dataIn[1:0]	2
sum	1
[0]	0
dataOut[3:0]	0011
fault_detected	0

CONCLUSION

BIST verifies all or a portion of the internal functionality of the IC. It is becoming an integral part of many time critical circuits recently. In this experiential learning assignment, a complete BIST design was implemented for full adder which included Output Response Analyzer (ORA) , Test Pattern Generator.

The faults were injected in the Circuit Under Test and analysis was done whether the BIST is able to detect the faults.

The simulation results obtained were verified with hand calculations.