RV College of Engineering®

Go, change the world

Autonomous institution affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE, New Delhi, Accredited by NAAC, Bengaluru.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Design for Testing and Testability (18EC7F3)

Implementation of SISR, MISR and Cellular Automata

Submitted by,

Student name	USN
Lalith Kumar E	**1RV18EC079**
Neha Daoo	**1RV18EC096**

Guided by
Mrs. Namita Palecha
Assistant Professor
Electronics and Communication Engineering

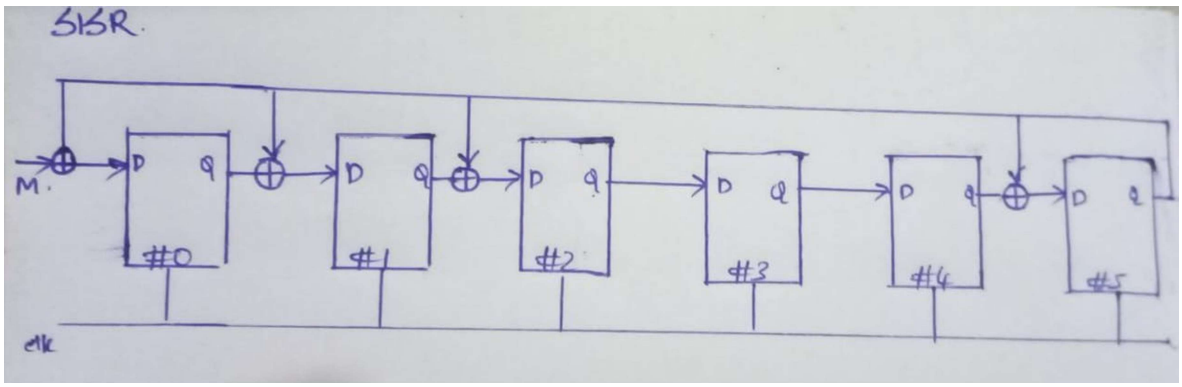
Electronics and Communication Engineering

2021-22

1. SISR Implementation

Single Input Signature Analyzer's are used to reduce the length of the test sequence for comparison when a long stream of inputs are given to circuit after testing. They help in analyzing circuits when billions of test vectors are provided to a circuit.

Polynomial chosen for SISR is $f(x) = x^6 + x^5 + x^2 + x + 1$



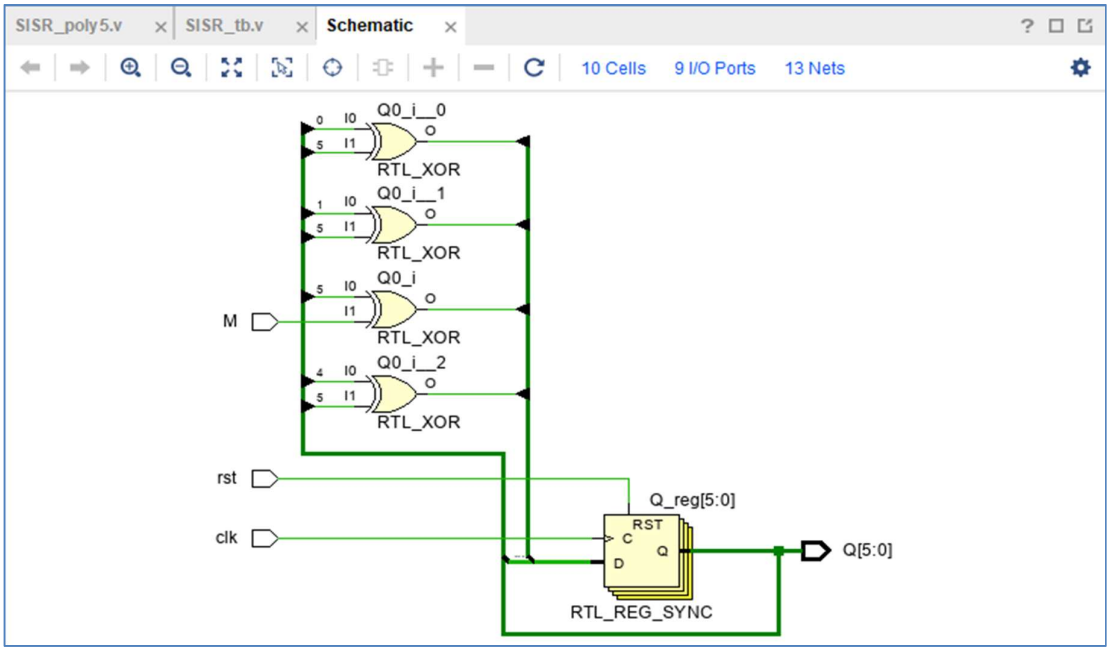
Fault Free Output Response is M0 =1111_0010_1101

Fault Output Response is M1=1111_0000_1101

SISR Equations:-

```
Q[0]<=Q[5]^M;
Q[1]<=Q[0]^Q[5];
Q[2]<=Q[1]^Q[5];
Q[3]<=Q[2];
Q[4]<=Q[3];
Q[5]<=Q[4]^Q[5];
```

Schematic:-



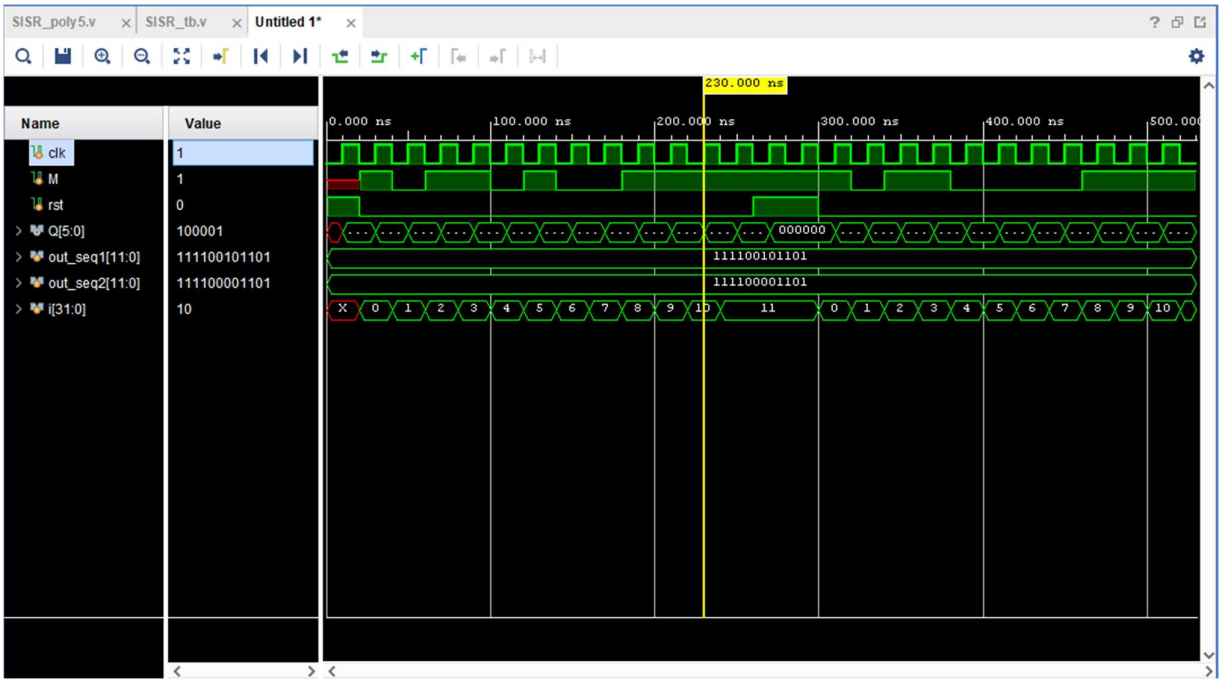
Verilog Code:-

```
23  module SISR_poly6(input clk,input rst,input M,output reg[5:0]Q);
24  // polynmial is x^6+x^5+x^2+x+1
25  always@(posedge clk)
26  begin
27      if(rst)
28          begin
29              Q<=5'd0;
30          end
31      else
32          begin
33              Q[0]<=Q[5]^M;
34              Q[1]<=Q[0]^Q[5];
35              Q[2]<=Q[1]^Q[5];
36              Q[3]<=Q[2];
37              Q[4]<=Q[3];
38              Q[5]<=Q[4]^Q[5];
39          end
40  end
41
42  endmodule
```

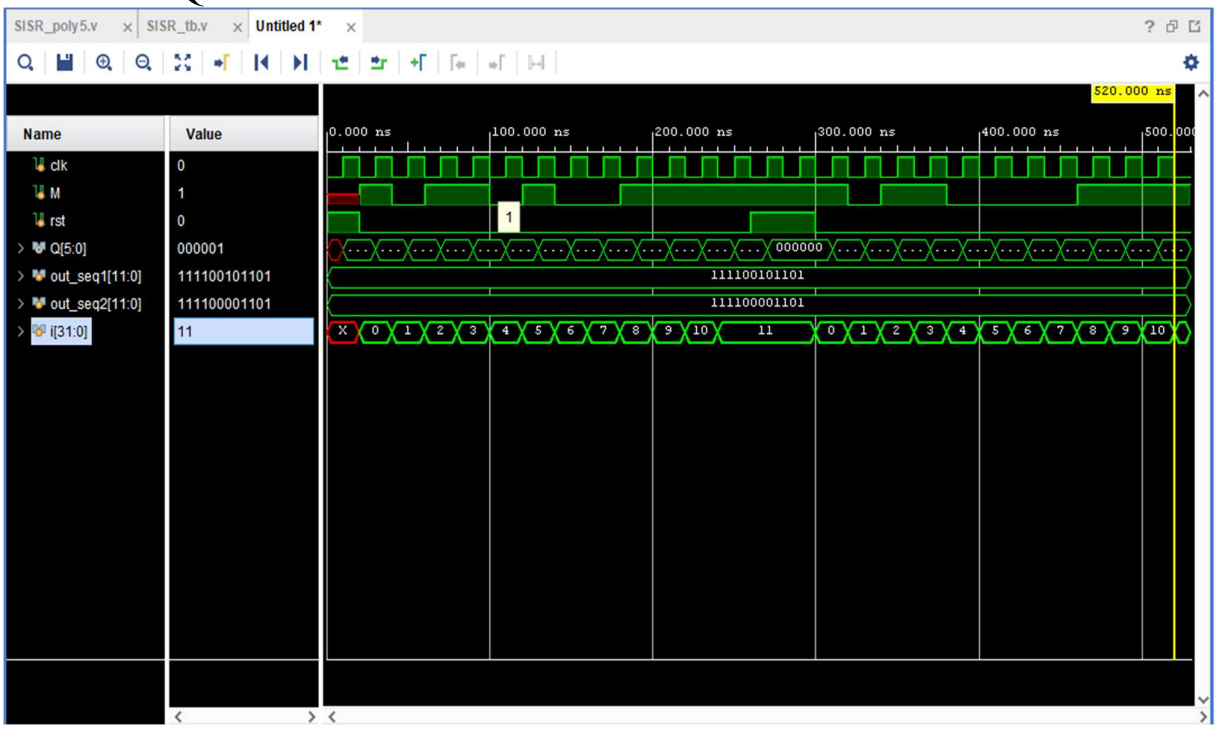
Test bench:-

```
module SISR_tb();

reg clk,M,rst;
wire[5:0]Q;

reg[11:0] out_seq1,out_seq2;//Output sequence of 12 bits is reduced to 6 bits by SISR.
integer i;
 SISR_poly6 uut(clk,rst,M,Q);

initial
    begin
        clk=0;
        out_seq1=12'b1111_0010_1101;  //Output sequence response Fault Free  -----> this way of propagation
        out_seq2=12'b1111_0000_1101;  //Output sequence faulty
    end

always
#10clk=~clk;

initial
    begin
    rst=1;
    #20
        rst=0;
        for(i=0;i<4'd11;i=i+1)
            begin
                M=out_seq1[i];
                #20;
            end
    #20
    rst=1;
    #40
    rst=0;
    for(i=0;i<4'd11;i=i+1)
            begin
                M=out_seq2[i];
                #20;
            end

    #10$finish;
    end
endmodule
```

Simulation Output:-



Fault Free Q=100001

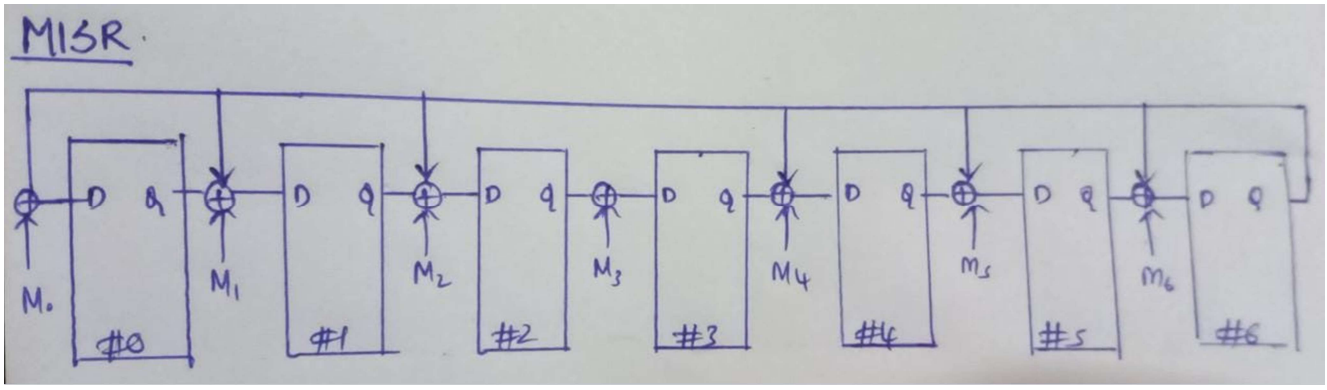


Faulty output Q=000001

Hence for the following sequence of patterns, fault is detected.

MISR Implementation:-

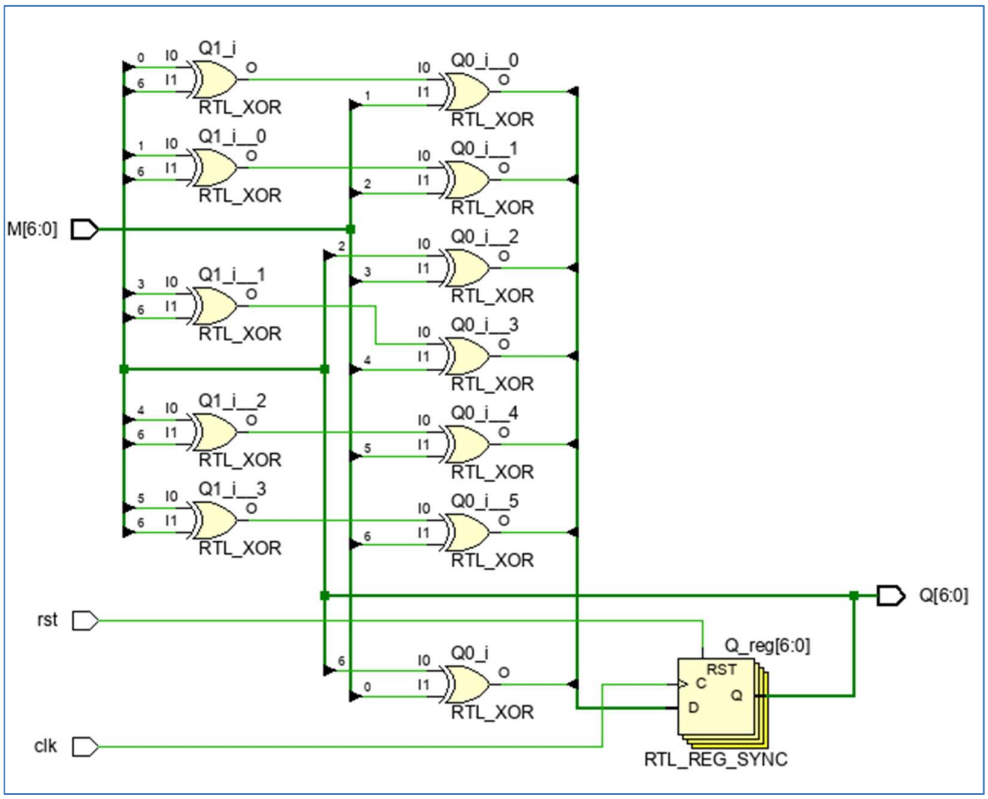
Polynomial chosen for MISR is $f(x) = x^7 + x^6 + x^5 + x^4 + x^2 + x + 1$



MISR Equations:-

Q[0]=Q[6]^M[0];
Q[1]=Q[0]^Q[6]^M[1];
Q[2]=Q[1]^Q[6]^M[2];
Q[3]=Q[2]^M[3];
Q[4]=Q[3]^Q[6]^M[4];
Q[5]=Q[4]^Q[6]^M[5];
Q[6<=Q[5]^Q[6]^M[6];

Schematic:-



Verilog Code:-

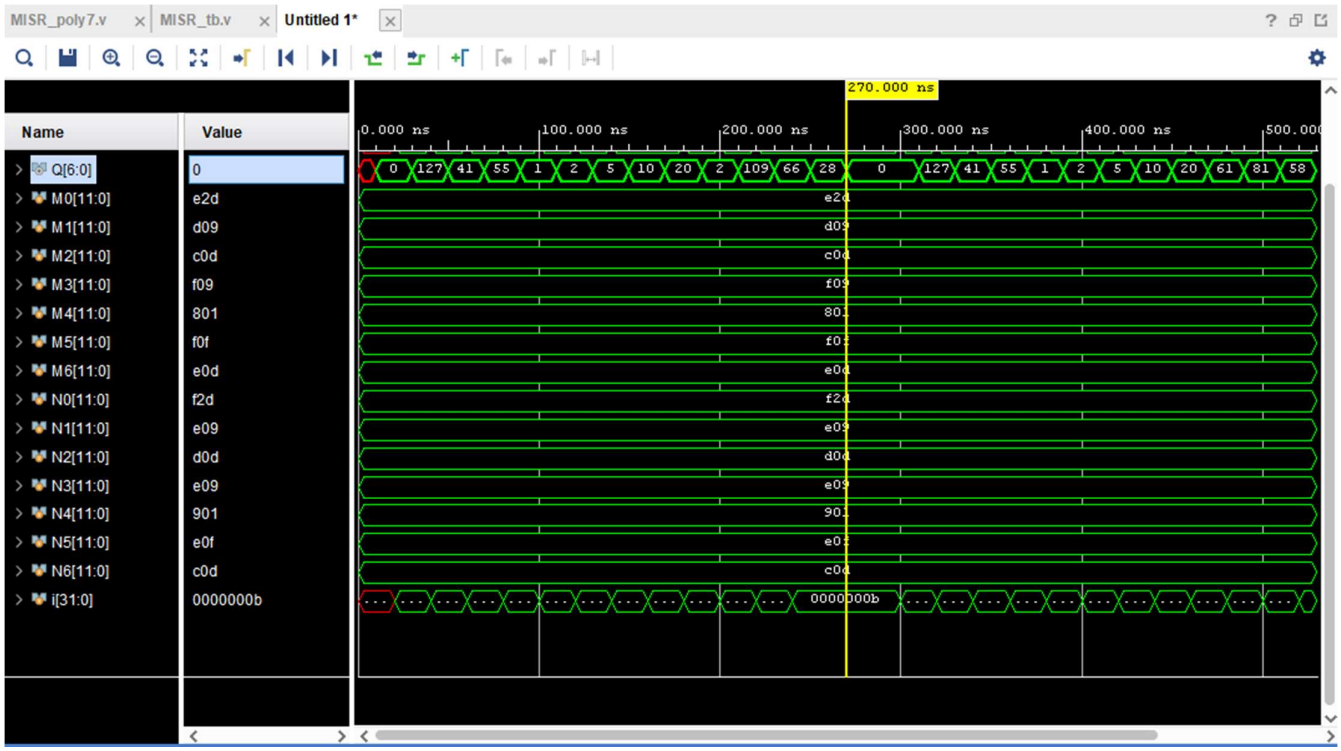
```
module MISR_poly7(input clk,input rst,input[6:0]M,output reg[6:0]Q);
// polynmial is x^7 + x^6 + x^5 + x^4 + x^2 + x^1 + 1
always@(posedge clk)
begin
    if(rst)
        begin
            Q<=5'd0;
        end
    else
        begin
            Q[0]<=Q[6]^M[0];
            Q[1]<=Q[0]^Q[6]^M[1];
            Q[2]<=Q[1]^Q[6]^M[2];
            Q[3]<=Q[2]^M[3];
            Q[4]<=Q[3]^Q[6]^M[4];
            Q[5]<=Q[4]^Q[6]^M[5];
            Q[6]<=Q[5]^Q[6]^M[6];
        end
end

endmodule
```

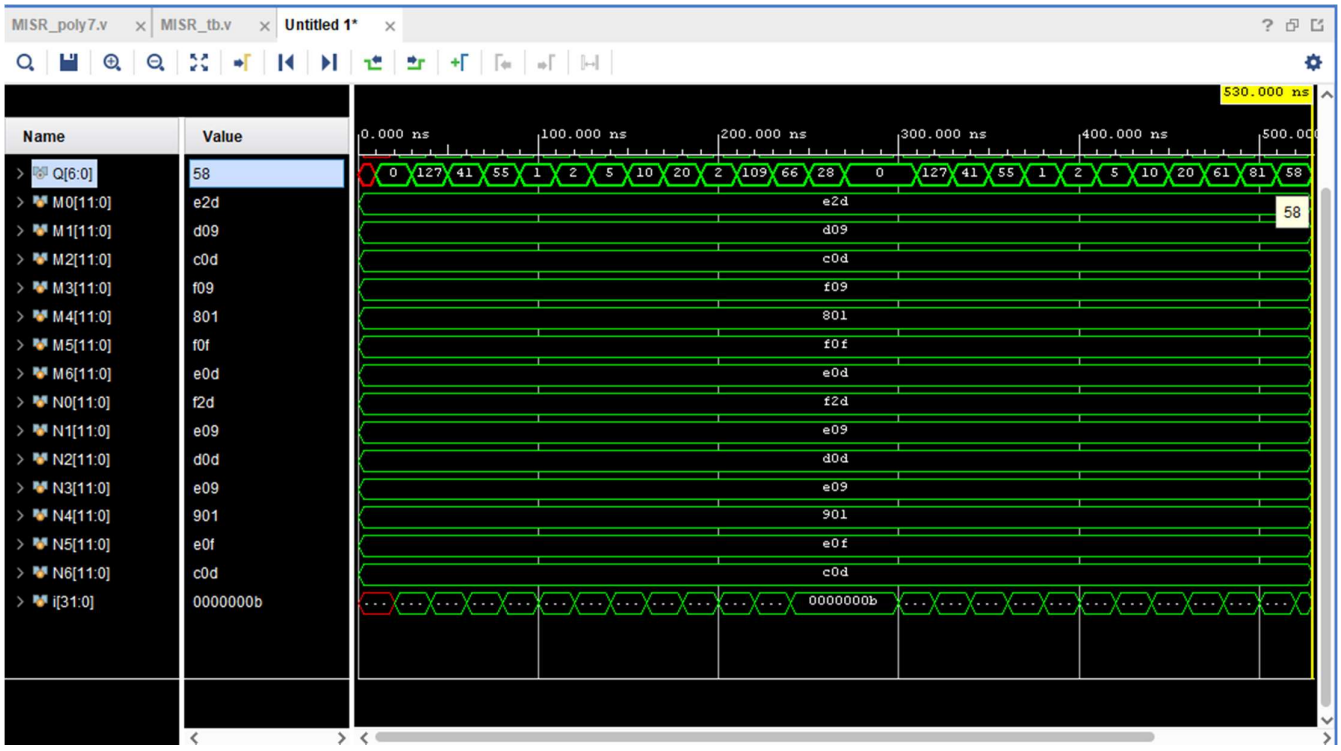
MISR_poly7.v | MISR_tb.v | Untitled 1*

L:/LALITH/3.College_Files/Semesters/7th SEMESTER/5.Design for Testability/4.Internal Assessment/DFT_assignmer

```
56      always
57      #10clk=~clk;
58
59      initial
60          begin
61          rst=1;
62          #20
63              rst=0;
64              for(i=0;i<4'd11;i=i+1)
65                  begin
66                      M[6:0]={M6[i],M5[i],M4[i],M3[i],M2[i],M1[i],M0[i]};
67                      #20;
68                  end
69          #20
70          rst=1;
71          #40
72          rst=0;
73          for(i=0;i<4'd11;i=i+1)
74                  begin
75                      M[6:0]={N6[i],N5[i],N4[i],N3[i],N2[i],N1[i],N0[i]};
76                      #20;
77                  end
78
79          #10$finish;
80          end
81      endmodule
82
83
```

Simulation Output:-



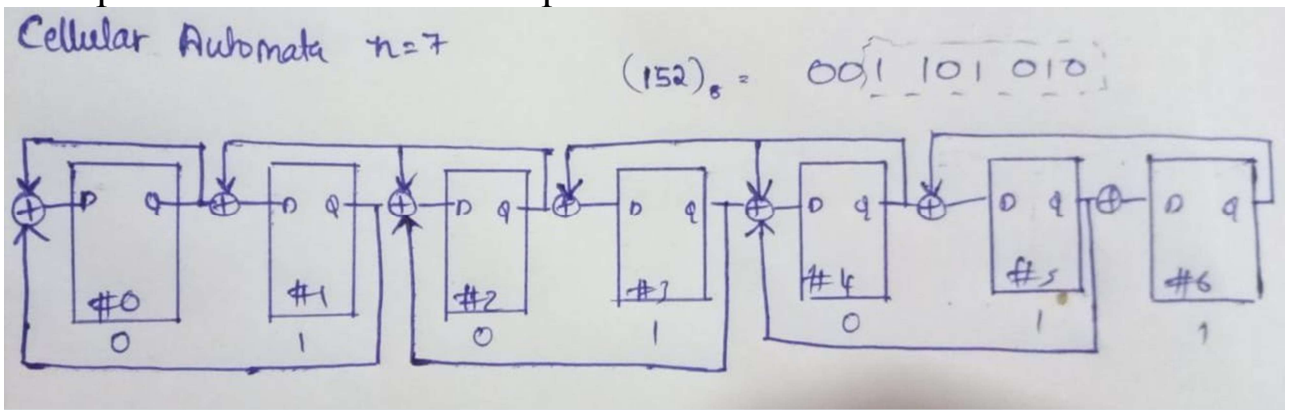


Cellular Automata:-

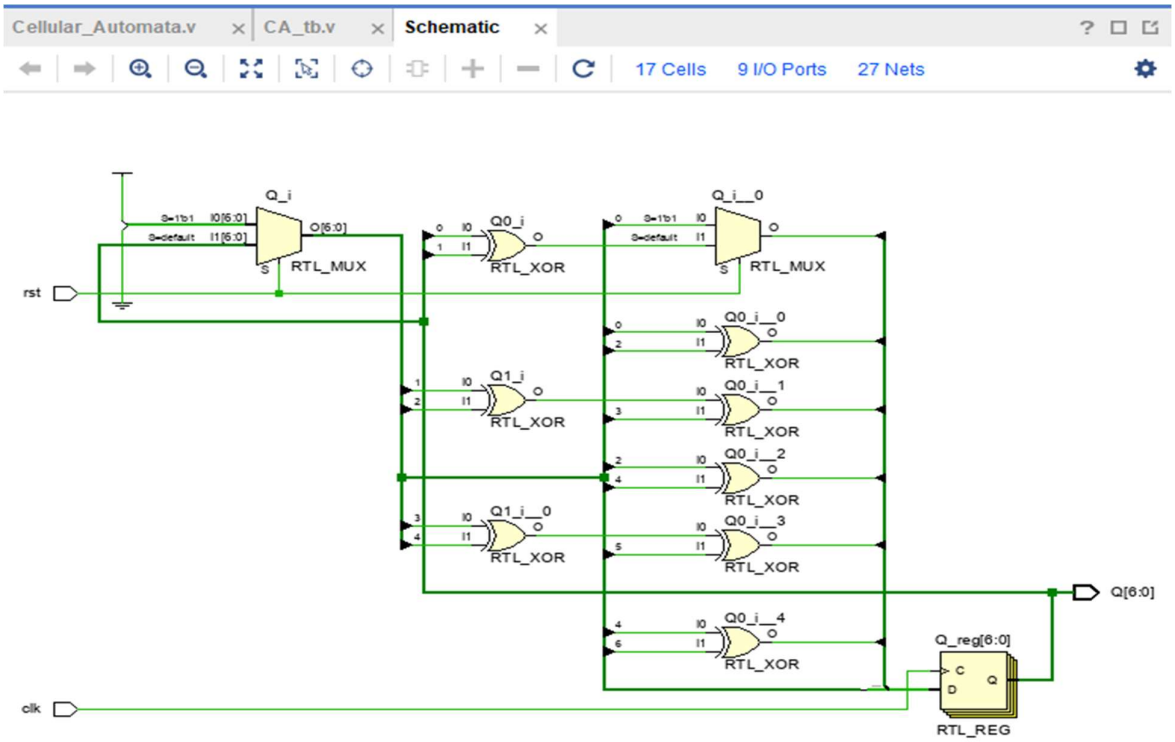
Cellular Automata is used for generating random patterns and has many applications in the fields of physics, theoretical biology and microstructure modeling. Quantum Dot Cellular Automata (QCA) is one of the emerging trends in the field of nanotechnology which help to overcome the limitations of CMOS technology.

For n=7 , Rule $(152)_8$ is applied as $(001\ 101\ 010)_2$

'1' represents rule 90 and '0' represents rule 150



Schematic:-



Verilog Code

```
module Cellular_Automata(input clk,input rst,output reg[6:0]Q);
//n=7 celular automata
always@(posedge clk)
    begin
        if(rst)
            Q=7'b1;
        else
            Q[0]<=Q[0]^Q[1];
            Q[1]<=Q[0]^Q[2];
            Q[2]<=Q[1]^Q[2]^Q[3];
            Q[3]<=Q[2]^Q[4];
            Q[4]<=Q[3]^Q[4]^Q[5];
            Q[5]<=Q[4]^Q[6];
            Q[6]<=Q[5];
    end
endmodule
```

Test bench

```
module CA_tb();

reg clk,reset;
wire[6:0]Q;

Cellular_Automata uut(clk,reset,Q);

always
#2clk=~clk;

initial
    begin
        clk=0;
        reset=1;
        #4
        reset=0;
    end
endmodule
```

Simulation:-

The cellular automata repeats the patterns after 512 ns as shown.