• Efficient State Encoding

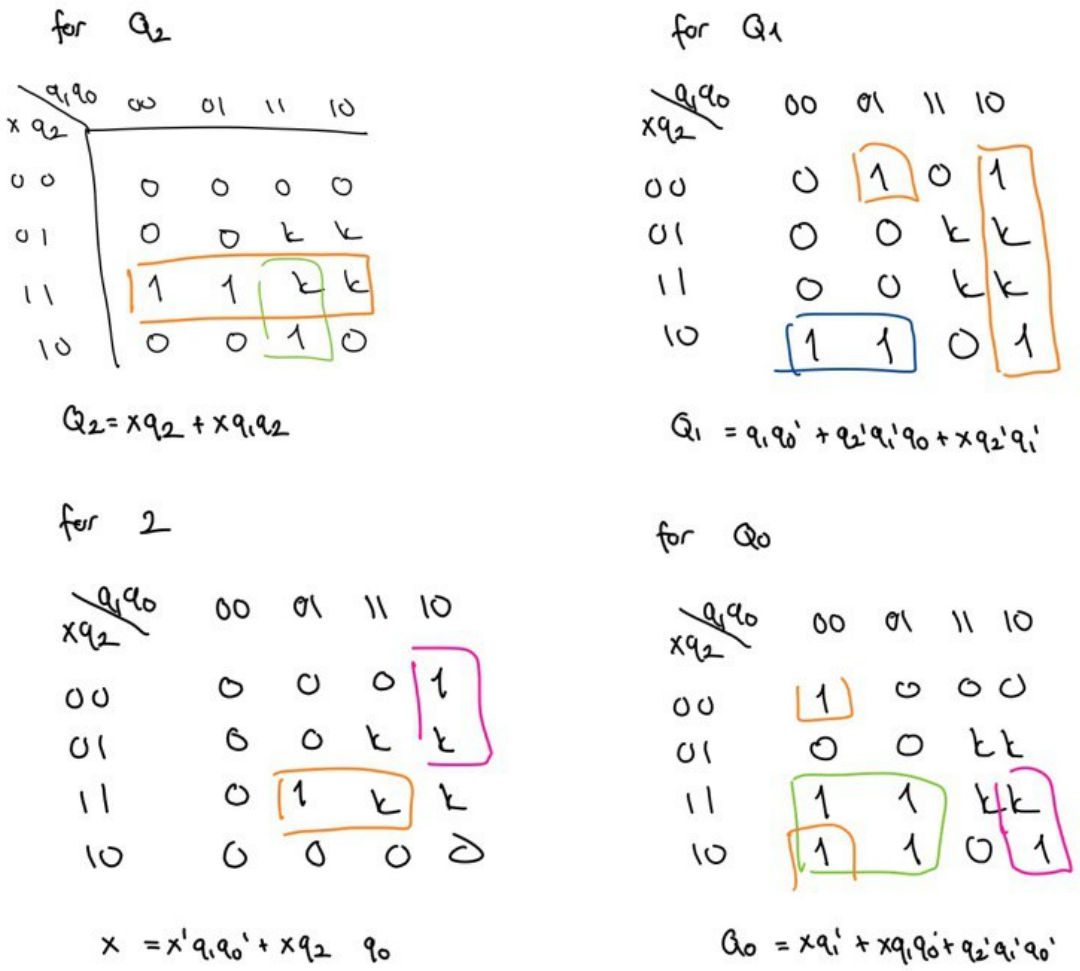
There exist various state encoding techniques aimed at minimizing power consumption and enhancing circuit speed. Additionally, optimizing efficiency often involves reducing states that do not impact the outcome. Binary encoding is commonly employed for Finite State Machines (FSMs) due to its simplicity and efficiency in representing states using binary notation.

• State Encoding for The Experiment

The state encoding used in the experiments is provided below in binary format:

state	binary code
A	000
B	001
C	010
D	011
E	100
F	101

Figure 3: State Encoding Example



LUT4 Usage:
To design the circuit using LUT-4, it is advisable to opt for the one-hot encoding technique. In this approach, states are coded as 1000, 0100, 0010, and 0001. Utilizing a LUT-4 with four entries allows us to select the state efficiently.

Verilog Code:

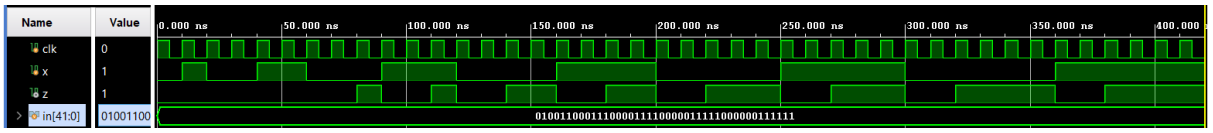
```
module fsm1(
input x,clk,
output z
);
    wire [2:0]Q;
    reg [2:0]q;
    initial begin
    q=3'b000;
    end
    assign z=(x & q[2] & q[0]) | (~x & ~q[0] & q[1] );
    always @(posedge clk)begin
        q[2]<=Q[2];
        q[1]<=Q[1];
        q[0]<=Q[0];
    end
    assign Q[0]=(~(q[2]) & ~(q[1]) & ~(q[0])) | (x & ~(q[1])) | (x &
~(q[0]));
    assign Q[1]=(~(q[2]) & ~(q[1]) & x) | (q[1] & ~q[0]) | (~(q[2]) &
~(q[1]) & (q[0]));
    assign Q[2]=(x & q[2]) | (x & q[1] & q[0]);
endmodule
```

Testbench Code:

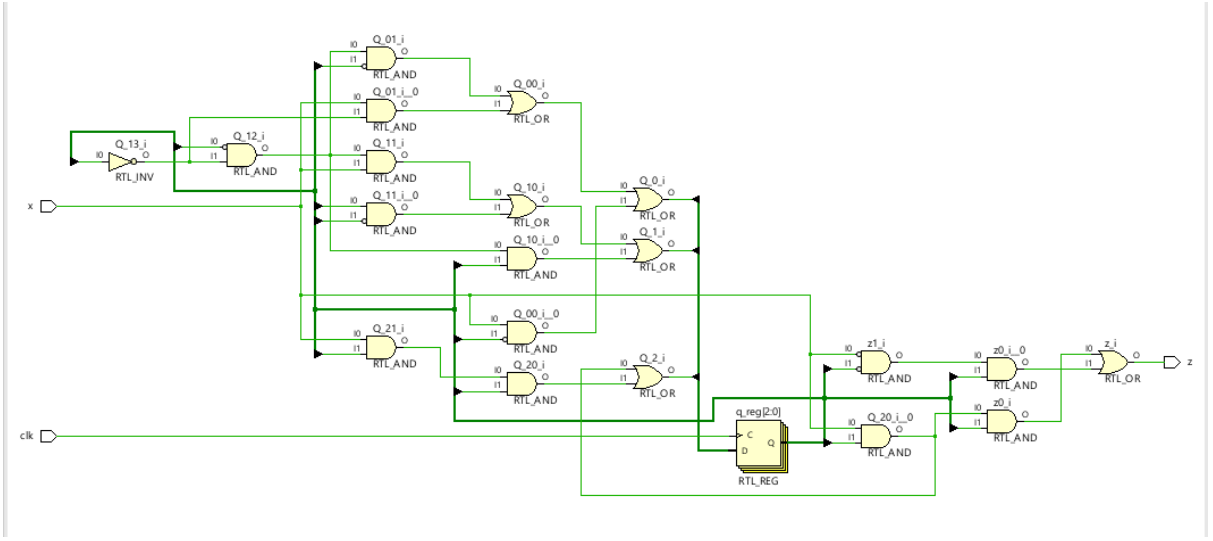
```
module tb_fsm();
    reg clk,x;
    reg [41:0] in;
    wire z;
    fsm1 uut(.x(x), .clk(clk), .z(z));
    integer i;
    initial begin
        clk=1'b0;
        in=42'b 010011000111000011110000011111000000111111;
        i=41;
```

```
    while(i>=0)
        begin
        clk<=~clk;
        x=in[i];
        #5
        clk<=~clk;
        i=i-1;
        #5;
    end
    $finish;
    end
endmodule
```

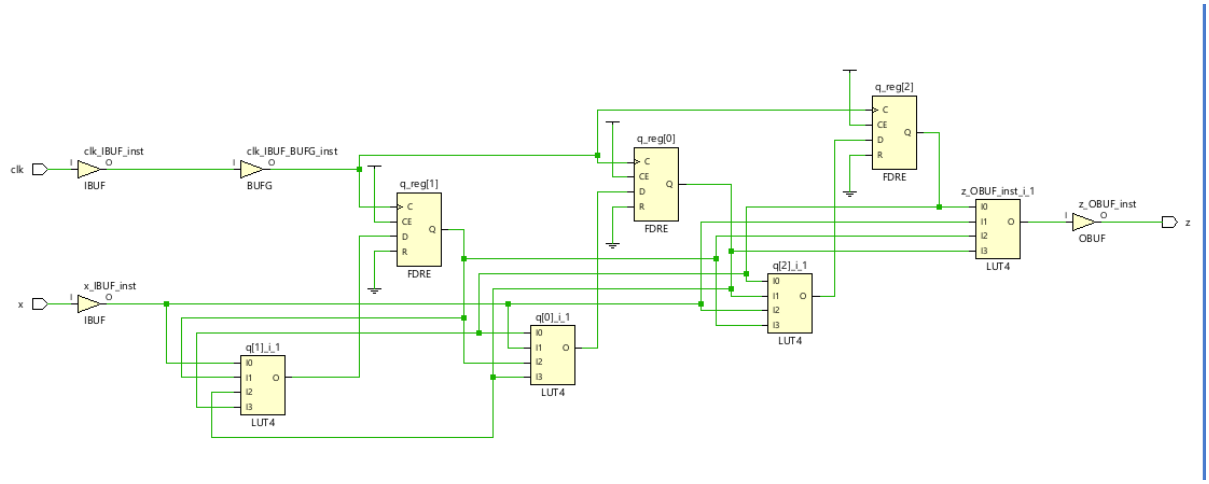
Simulation Results:



RTL Schematic:



Technology Schematic:

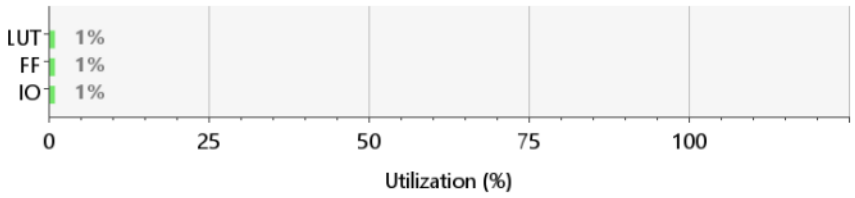


TİMİNG and Utilization Reports:

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
x	z	8.617	SLOW	2.551	FAST

Resource	Utilization	Available	Utilization %
LUT	2	63400	0.00
FF	3	126800	0.00
IO	3	210	1.43



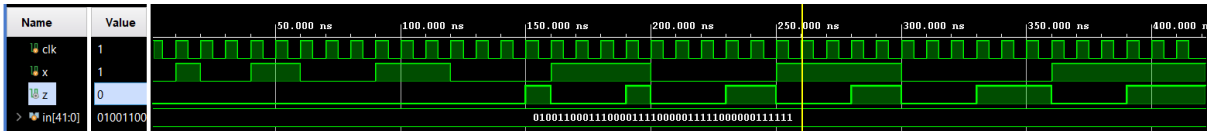
Faulty Outputs:
In this particular design, undesired 1 outputs occur due to the nature of the FSM being a Mealy machine. This is because the design generates a faulty 1 output at the occurrence of the third consecutive 1, considering input 1 from x as well. When the change is 101 or 010 it means the output is wrong harmful

To address this issue and transition to a Moore type machine, where the x input is not considered through the stages, you can integrate a D-type flip-flop (DFF) at the end of the circuit. This modification ensures that the output arises at the occurrence of the fourth consecutive 1, aligning with the characteristics of a Moore-type machine.

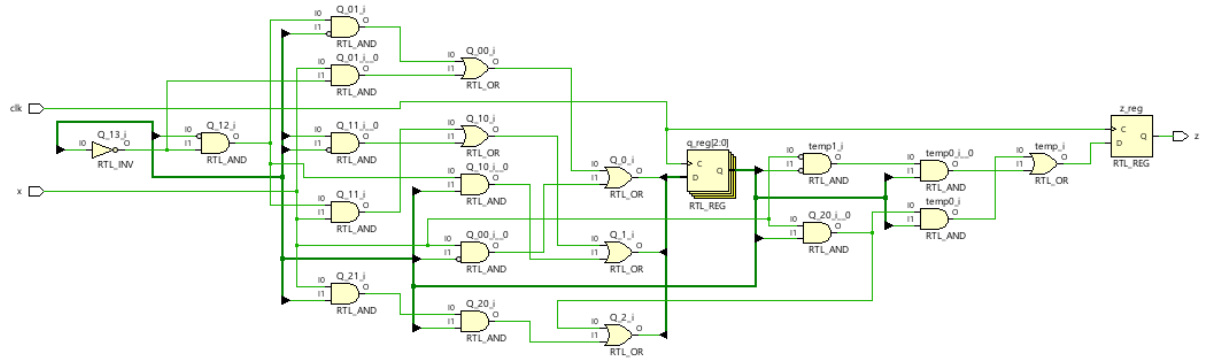
Moore Machine

```
module fsm1(
    input x,clk,
    output reg z
);
     wire temp;
    wire [2:0]Q;
    reg [2:0]q;
    initial begin
        q=3'b000;
    end
    assign temp=(x & q[2] & q[0]) | (~x & ~q[0] & q[1] );
    always @(posedge clk)begin
        q[2]<=Q[2];
        q[1]<=Q[1];
        q[0]<=Q[0];
        z<=temp;
    end
    assign Q[0]=(~(q[2]) & ~(q[1]) & ~(q[0])) | (x & ~(q[1])) | (x &
~(q[0]));
    assign Q[1]=(~(q[2]) & ~(q[1]) & x) | (q[1] & ~q[0]) | (~(q[2]) &
~(q[1]) & (q[0]));
    assign Q[2]=(x & q[2]) | (x & q[1] & q[0]);
endmodule
```

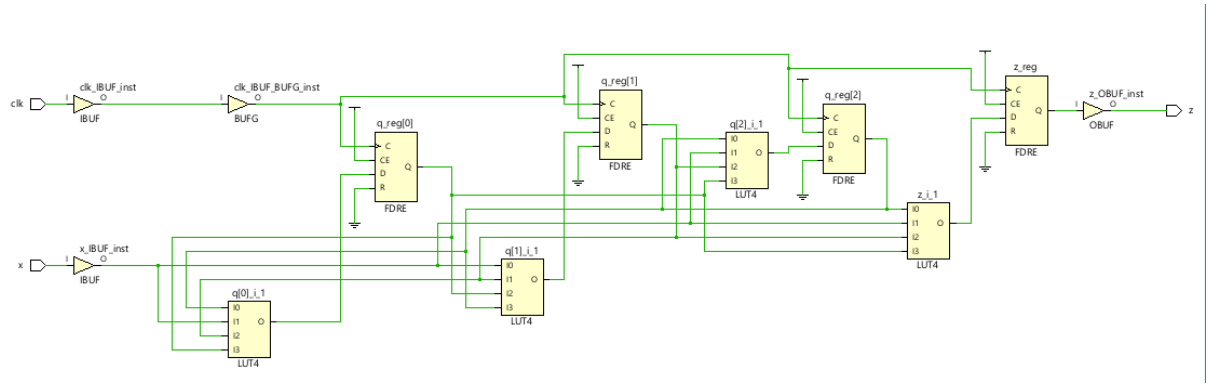
Simulation Result:



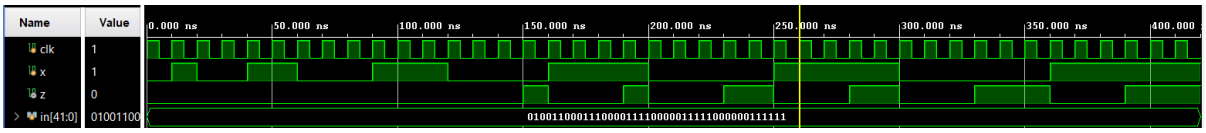
RTL Schematic:



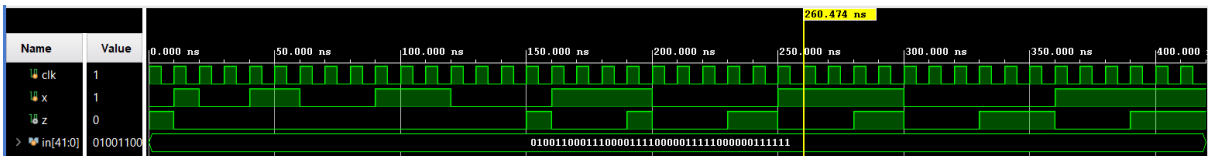
Technology Schematic:



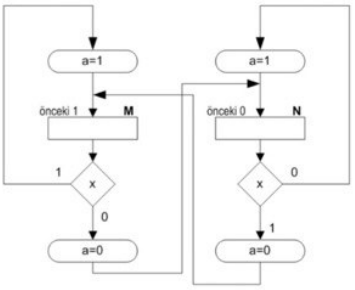
for initial State=111



for initial State=110



The circuit wont stuck if we start from invalid states.



Q	Q' x=0	Q' x=1
M	N,0	M,1
N	N,1	M,0

State	Codes
M	0
N	1

x	q_2	Q_2	a
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	0

for Q

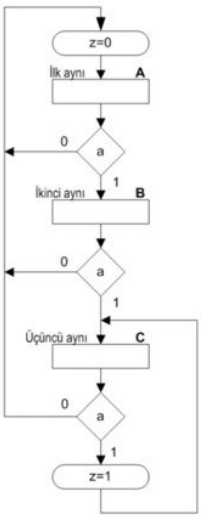
x|q	0	1
0	1	1
1	0	0

for a

x|q	0	1
0	0	1
0	1	0

$a = xq_2' + x'q_2$

$Q_2 = x'$



Q	Q' x=0	Q' x=1
A	A,0	B,0
B	A,0	C,0
C	A,0	C,1

State	Codes
A	00
B	01
C	10

a	q_1	q_0	Q_1	Q_0	2
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	k	k	k
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	k	k	k

for Q_1

a|q_1q_0	00	01	11	10
0	0	0	k	0
1	0	1	k	1

$Q_1 = aq_0 + aq_1$

for Q_2

a|q_1q_0	00	01	11	10
0	0	0	k	0
1	1	0	k	0

$Q_2 = aq_1'q_0'$

for 2

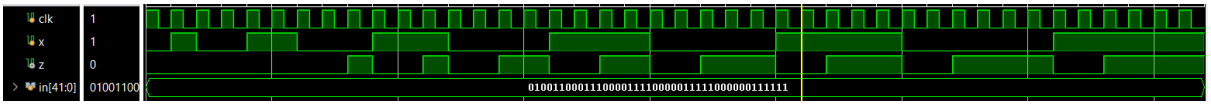
a|q_1q_0	00	01	11	10
0	0	0	k	0
1	0	0	k	1

$2 = aq_1$

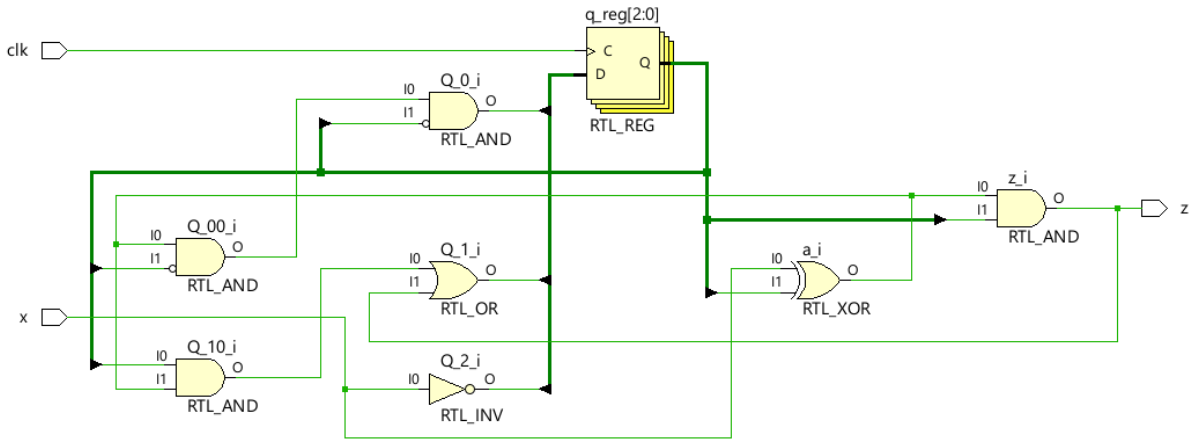
Verilog Code:

```
module fsm2(
input x,clk,
output z
    );
    wire [2:0]Q;
    reg [2:0] q;
    wire a;
        initial begin
    q=3'b000;
    end
    assign a=x ^ q[2];
    assign Q[2]=~x;
    assign Q[1] = (q[0] & a) | (a & q[1]);
    assign Q[0] = a & (~q[1]) & (~q[0]);
    assign z= a & q[1];
    always @(posedge clk)begin
        q[2]=Q[2];
        q[1]=Q[1];
        q[0]=Q[0];
    end
endmodule
```

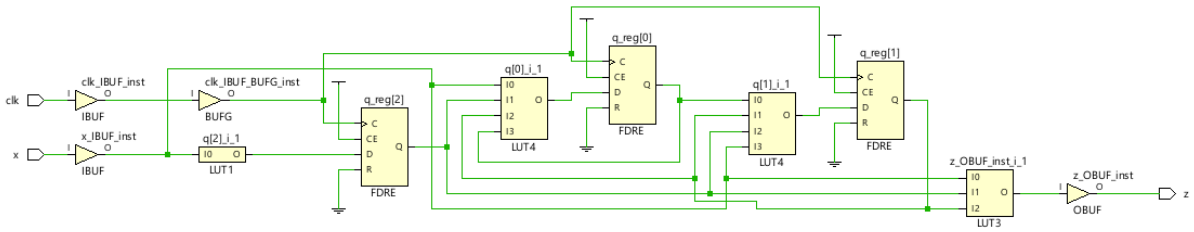
Simulation results:



RTL Schematic:

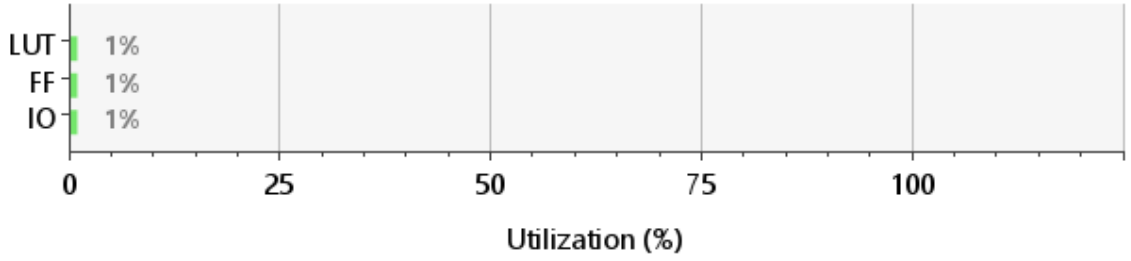


Technology Schematic:



Utilization and Timing Reports:

Resource	Utilization	Available	Utilization %
LUT	2	63400	0.00
FF	3	126800	0.00
IO	3	210	1.43



From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
x	z	8.617	SLOW	2.551	FAST

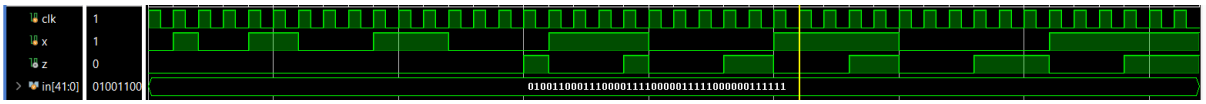
Moore Type:

Verilog Code:

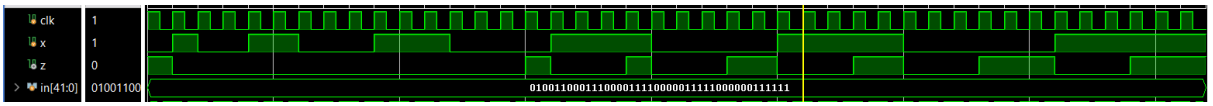
```
module fsm2(
    input x,clk,
    output reg z
);
    wire temp;
    wire [2:0]Q;
    reg [2:0] q;
    wire a;
    initial begin
```

```
    q=3'b011;
    end
    assign a=x ^ q[2];
    assign Q[2]=~x;
    assign Q[1] = (q[0] & a) | (a & q[1]);
    assign Q[0] = a & (~q[1]) & (~q[0]);
    assign temp= a & q[1];
    always @(posedge clk)begin
        q[2]=Q[2];
        q[1]=Q[1];
        q[0]=Q[0];
        z<=temp;
    end
endmodule
```

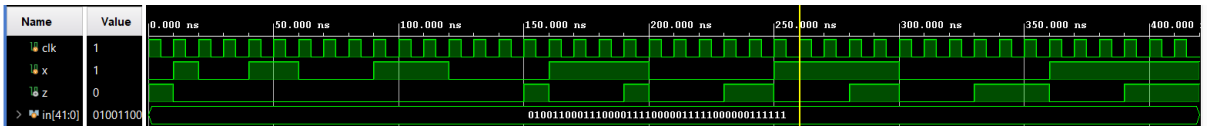
Simulation Results:



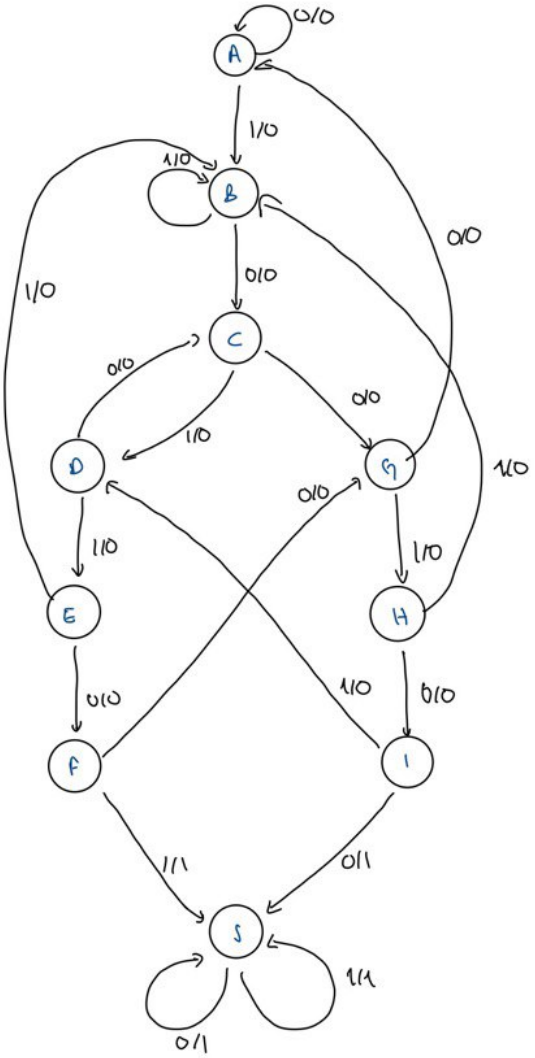
From state 111



From state 110:



Circuit that detects ’101’ and ’100’



	x=0	x=1
A	A/0	B/0
B	C/0	B/0
C	G/0	D/0
D	C/0	E/0
E	F/0	B/0
F	G/0	S/1
G	A/0	H/0
H	I/0	B/0
I	S/1	D/0
S	S/1	S/1

State	encoding
A	0000
B	0001
C	0011
D	0010
E	0110
F	0111
G	0101
H	0100
I	1100
S	1000

Verilog Code:

```
module FSM3(
input reset,
input x,clk,
output z
);
reg [3:0] state;
parameter
    A = 4'b0000,
    B = 4'b0001,
    C = 4'b0011,
```

```
D = 4'b0010,
E = 4'b0110,
F = 4'b0111,
G = 4'b0101,
H = 4'b0100,
I = 4'b1100,
S = 4'b1000;

always@(posedge reset  or posedge clk)
begin
if(reset==1) state <= A;
else begin
    case (state)
        A:if(x) state <= B;
          else state <= A;

        B:if(x) state <= B;
          else state <= C;

        C:if(x) state <= D;
          else state <= G;

        D:if(x) state <= E;
          else state <= C;

        D: if(x) state<= E;
           else state <= C;

        E: if(x) state<= B;
           else state <= F;

        F: if(x) state<= S;
           else state <= G;

        G: if(x) state<= H;
           else state <= A;

        H: if(x) state<= B;
           else state <= I;

        I: if(x) state<= D;
           else state <= S;
```

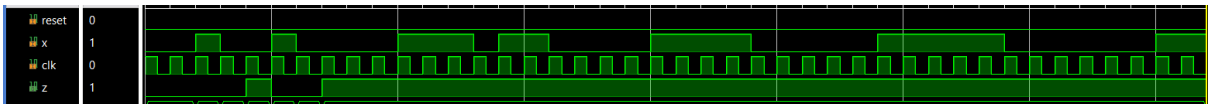
```
            S: if(x) state<= S;
               else state <= S;

            default : state<= A;

            endcase
            end
            end
            assign z = (state == S)|(~(x)&(state==G)) | (x&(state==D));
endmodule
```

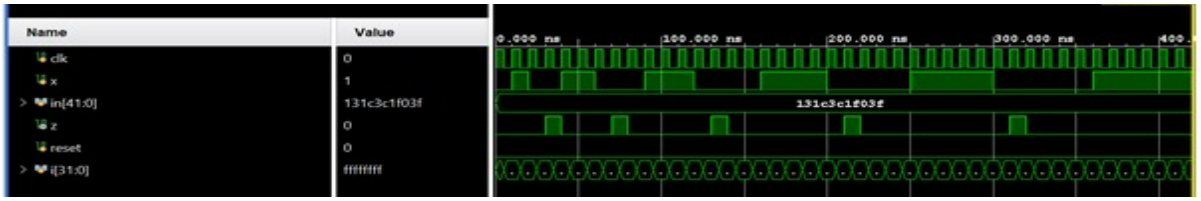
TB Code:

```
`timescale 1ns / 1ps


module TB_FSM();
  reg clk,x;
  reg [41:0] in;
  wire z;
  reg reset;

  FSM3 uut(.x(x), .clk(clk), .z(z), .reset(reset));

  integer i;
  initial begin
  reset=1'b0;
  clk=1'b0;
  in=42'b 010010000111011000011110000011111100000011;
  i=41;
  while(i>=0)
  begin
  clk<=~clk;
  x=in[i];
  #5
  clk<=~clk;
  i=i-1;
  #5;
  end
  $finish;

  end
```

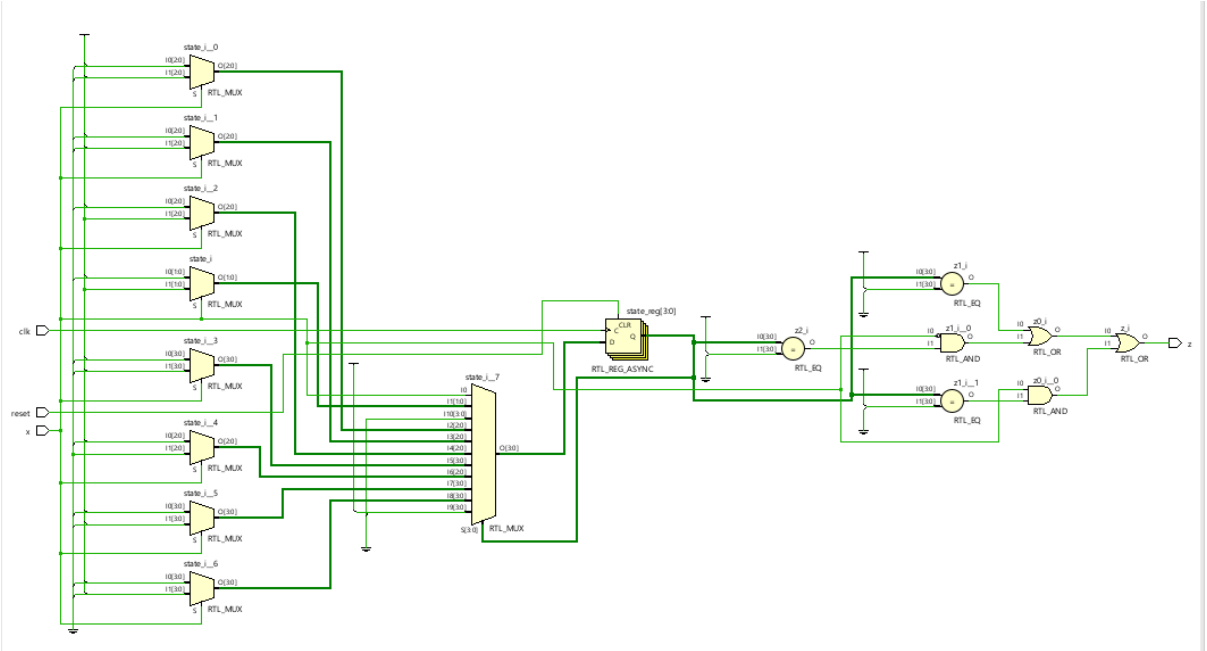
```
endmodule
```

Simulation result:

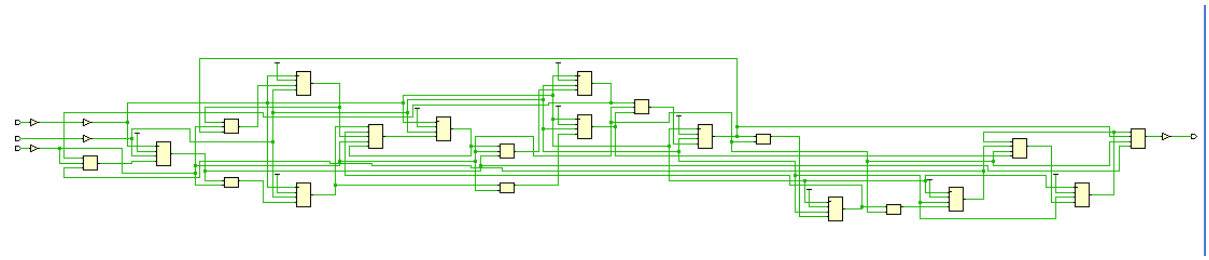




RTL Schematic:

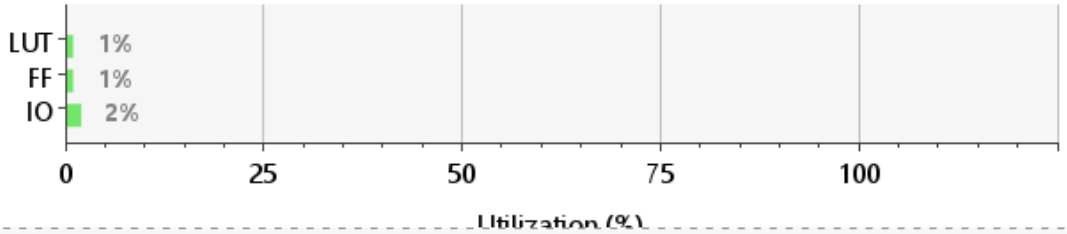


Technology Schematic:



Timing and Utilization Reports:
6 LUTs are used and delay is 8.459ns

Resource	Utilization	Available	Utilization %
LUT	6	63400	0.01
FF	10	126800	0.01
IO	4	210	1.90



From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
x	z	8.459	SLOW	2.537	FAST