DIGITAL SYSTEM DESIGN APPLICATIONS

(CRN: 11275)

THE REPORT OF EXPERIMENT – 7

Faculty of Electrical and Electronics Engineering

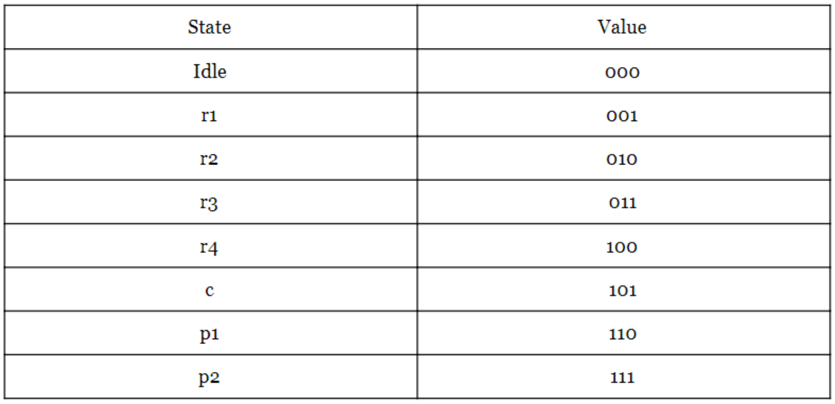
Electronics and Communication Engineering

Yusuf Tekin – 040200043

1. Circuit That Detects Four Consecutive 1 or 0

Finite State Machine (FSM) Encoding Methodes:

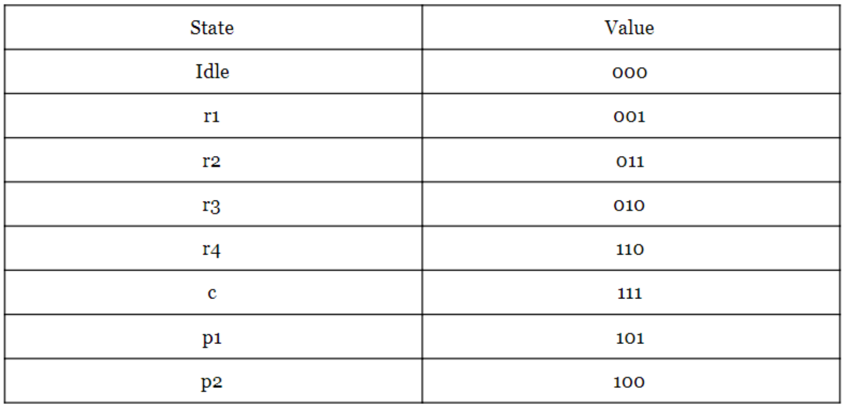
- **Binary Encoding:** In state machines, there could be a variety of states. In HDL coding these states are represented by an encoding number. Binary encoding method requires assigning these numbers one by one to minimize the length of the state vector, which is good for FPGA and PLA designs.



State	Value
Idle	000
r1	001
r2	010
r3	011
r4	100
c	101
p1	110
p2	111

Binary Encoded State Values Table[1]

- **Gray Encoding:** In gray encoding, only one-bit changes when moving between various states. As a result of this behavior, gray encoding consumes less power than binary coding. To explain furthermore, between each state jump, there are transistors on and off to represent 1s and 0s. This change in transistors causes the parasitic capacitors of the transistors to charge or discharge which requires power. With gray encoding, all the state jumps done by changing only one-bit that means the unchanged bits have not consumed any energy. Also, it can reduce glitches in the implementation.



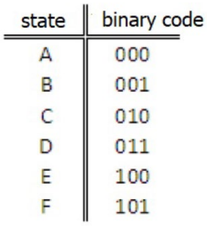
State	Value
Idle	000
r1	001
r2	011
r3	010
r4	110
c	111
p1	101
p2	100

Gray Encoded State Values Table[1]

- **One-Hot Encoding:** One-Hot Encoding simplifies the circuits above via using one bit for each state. This simplifies the circuit and reduces propagation delays which results in making the circuit compatible with higher frequency clocks. However, the trade-off is that one-hot encoding increases the number of flip-flops used to store the state of the system.

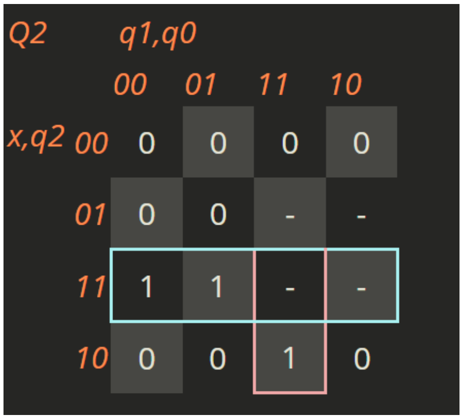
State	Value
Idle	00000001
r1	00000010
r2	00000100
r3	00001000
r4	00010000
c	00100000
p1	01000000
p2	10000000

One-Hot Encoded State Values Table[1]



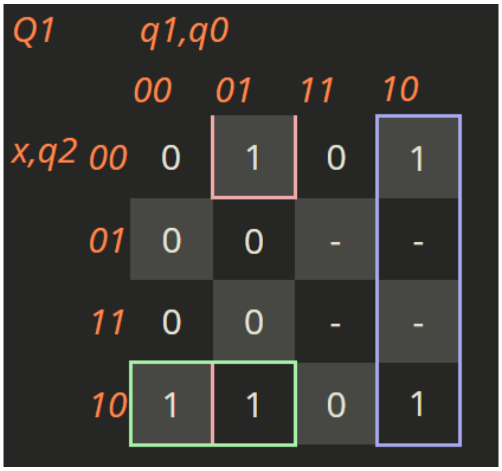
state	binary code
A	000
B	001
C	010
D	011
E	100
F	101

Binary Encoding



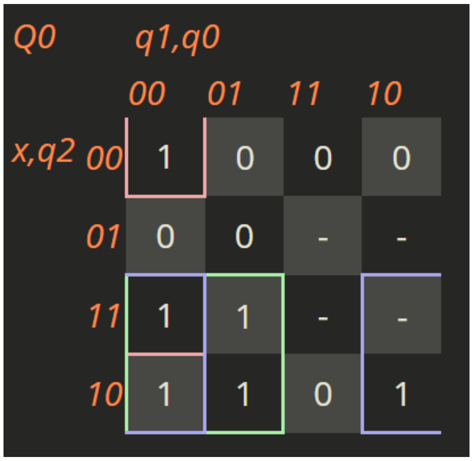
Q2		q1,q0			
		00	01	11	10
x,q2	00	0	0	0	0
	01	0	0	-	-
	11	1	1	-	-
	10	0	0	1	0

Karnaugh Map of Q2



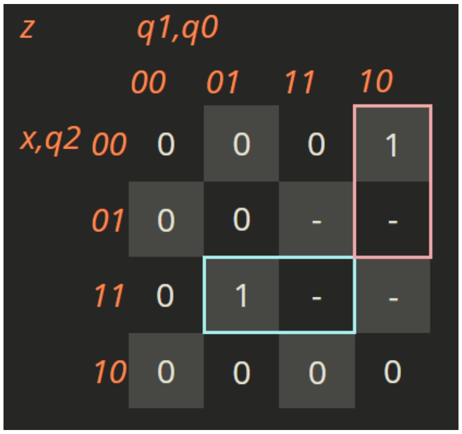
Q1	q1,q0			
x,q2	00	01	11	10
00	0	1	0	1
01	0	0	-	-
11	0	0	-	-
10	1	1	0	1

Karnaugh Map of Q1



Q0	q1,q0			
x,q2	00	01	11	10
00	1	0	0	0
01	0	0	-	-
11	1	1	-	-
10	1	1	0	1

Karnaugh Map of Q0



z	q1,q0			
x,q2	00	01	11	10
00	0	0	0	1
01	0	0	-	-
11	0	1	-	-
10	0	0	0	0

Karnaugh Map of Output z

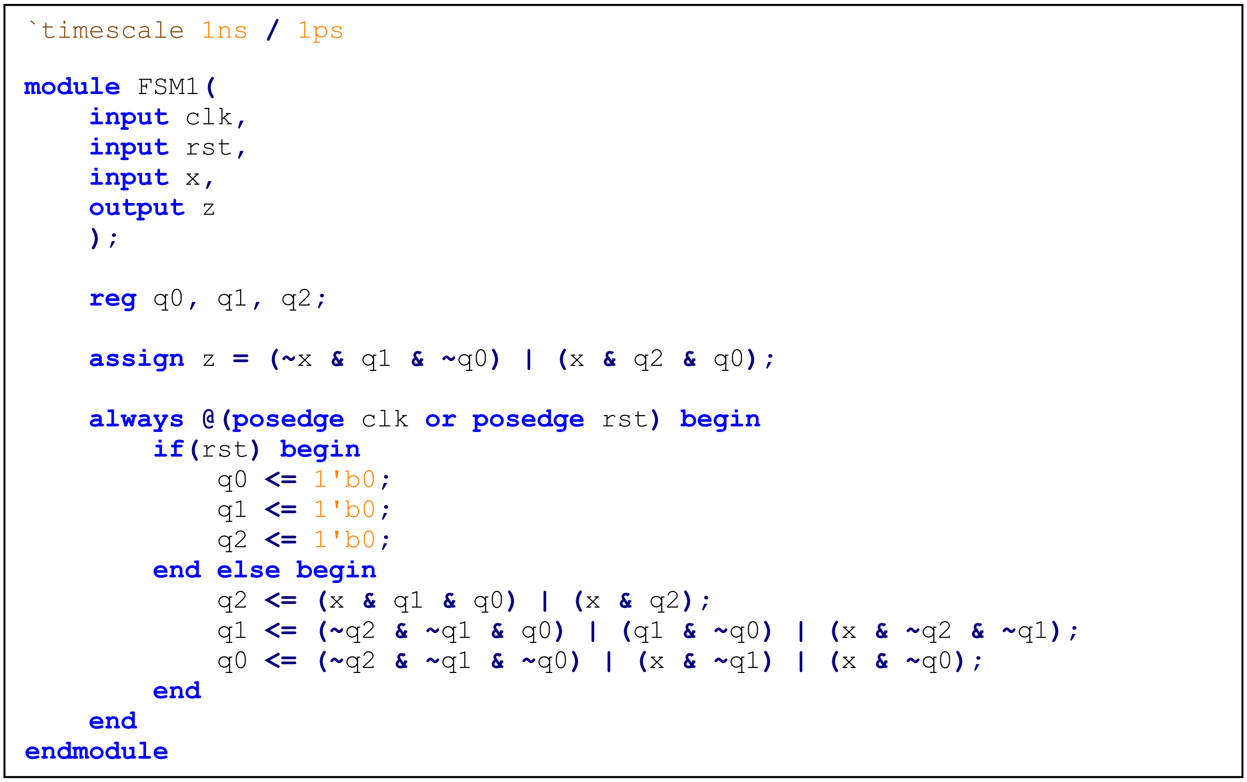
REDUCED EQUATIONS

$$z(x, q2, q1, q0) = x'q1q0' + xq2q0$$

$$Q2(x, q2, q1, q0) = xq1q0 + xq2$$

$$Q1(x, q2, q1, q0) = q2'q1'q0 + q1q0' + xq2'q1'$$

$$Q0(x, q2, q1, q0) = q2'q1'q0' + xq1' + xq0'$$



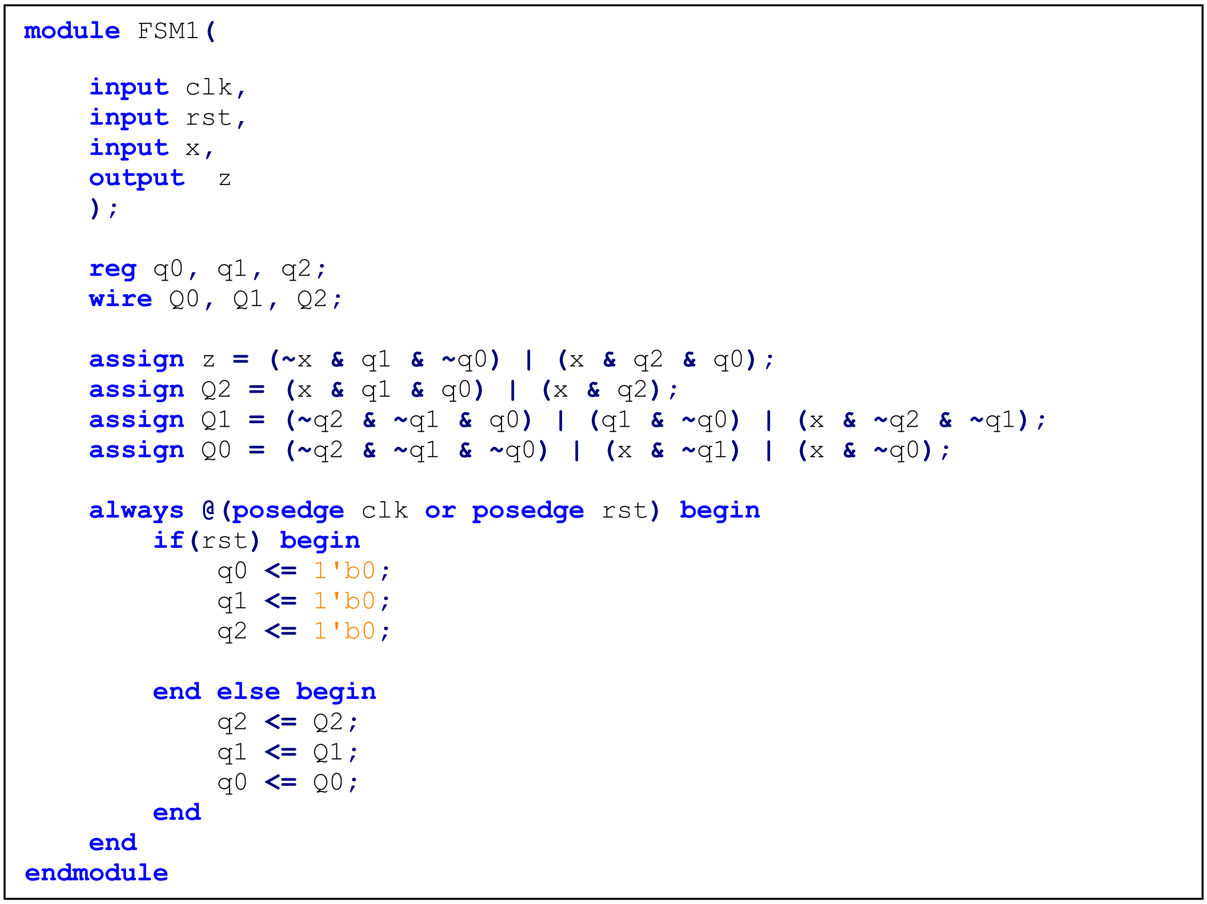
```
`timescale 1ns / 1ps

module FSM1(
    input clk,
    input rst,
    input x,
    output z
    );

    reg q0, q1, q2;

    assign z = (~x & q1 & ~q0) | (x & q2 & q0);

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            q0 <= 1'b0;
            q1 <= 1'b0;
            q2 <= 1'b0;
        end else begin
            q2 <= (x & q1 & q0) | (x & q2);
            q1 <= (~q2 & ~q1 & q0) | (q1 & ~q0) | (x & ~q2 & ~q1);
            q0 <= (~q2 & ~q1 & ~q0) | (x & ~q1) | (x & ~q0);
        end
    end
endmodule
```

FSM Verilog Code

Yusuf Tekin
040200043

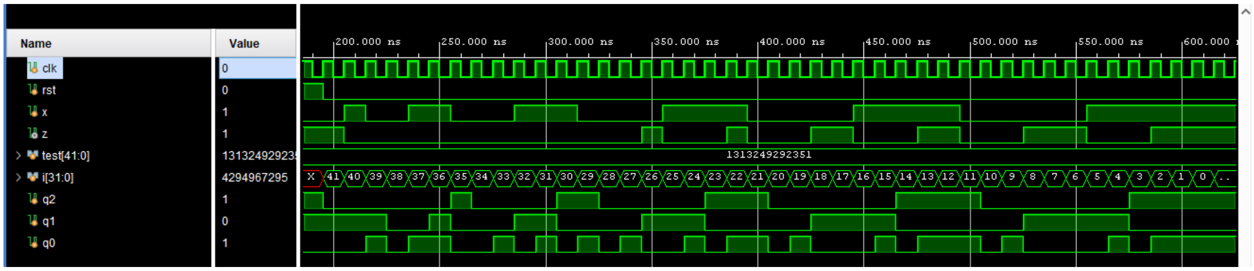
!!! This is an addition page to report!!!

After making FSM2, I fixed my code for FSM1 but after correcting my code there are not any faulty one or zero occurred in the simulation. Because of that I don't know what to do so, I put the fixed code with simulation in this addition page. (I use the same testbench)



```
module FSM1(

    input clk,
    input rst,
    input x,
    output  z
    );

    reg q0, q1, q2;
    wire Q0, Q1, Q2;

    assign z = (~x & q1 & ~q0) | (x & q2 & q0);
    assign Q2 = (x & q1 & q0) | (x & q2);
    assign Q1 = (~q2 & ~q1 & q0) | (q1 & ~q0) | (x & ~q2 & ~q1);
    assign Q0 = (~q2 & ~q1 & ~q0) | (x & ~q1) | (x & ~q0);

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            q0 <= 1'b0;
            q1 <= 1'b0;
            q2 <= 1'b0;

        end else begin
            q2 <= Q2;
            q1 <= Q1;
            q0 <= Q0;
        end
    end
endmodule
```

FSM1 Verilog Code (works correctly)

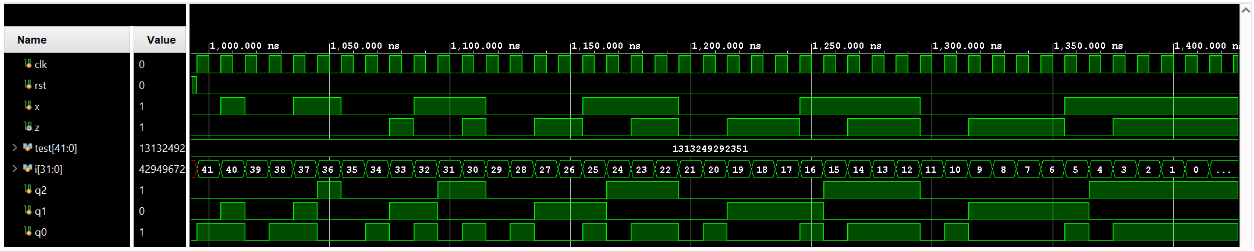


FSM1 Behavioral Simulation (no faulty)

From now on the report continues as wanted order.

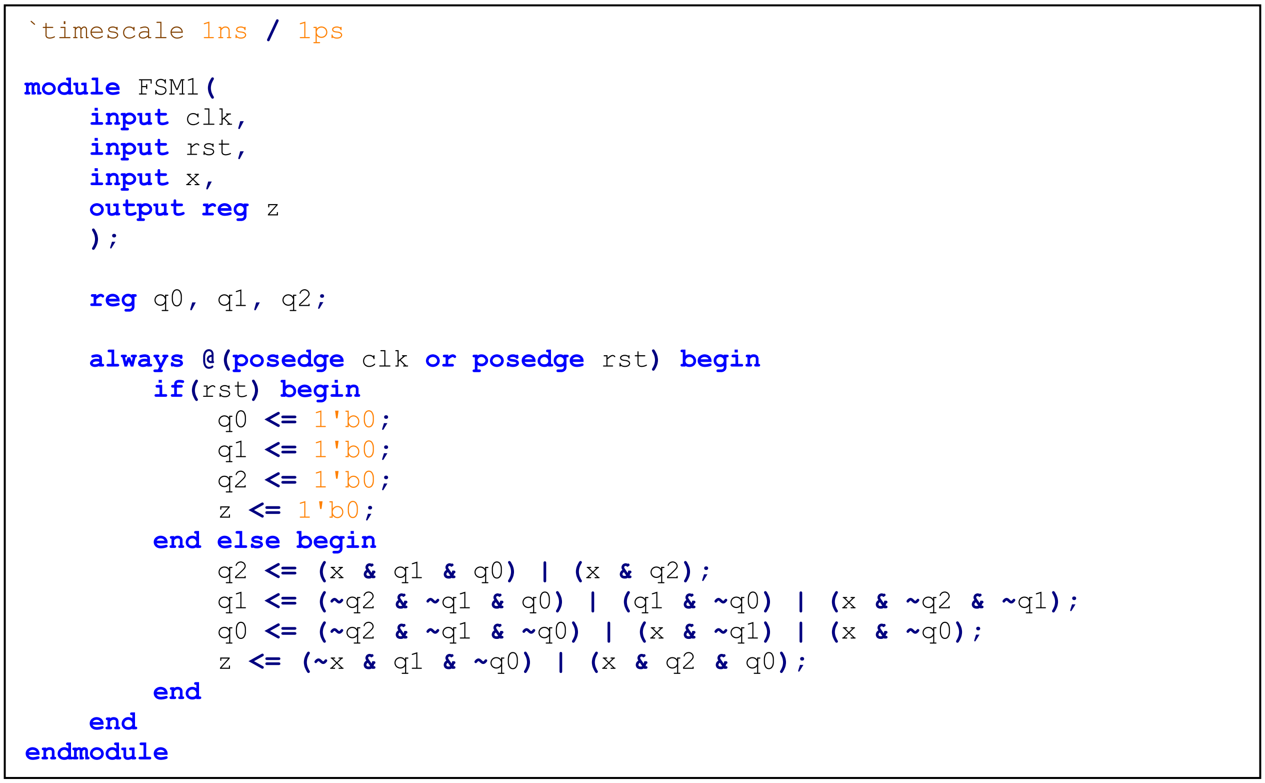
```
`timescale 1ns / 1ps

module FSM1_tb();

    reg   clk = 0;
    reg   rst = 0;
    reg   x   = 0;
    wire  z;

    FSM1 dut(
        .clk(clk),
        .rst(rst),
        .x(x),
        .z(z));

    always #5 clk = ~clk;

    reg [41:0] test =
42'b01_0011_0001_1100_0011_1100_0001_1111_0000_0011_1111;
    integer i;

    initial begin
        repeat(80) @(posedge clk);
        rst = 1;
        repeat(20) @(posedge clk);
        rst = 0;

        for(i=41; i>=0; i=i-1) begin
            x = test[i];
            @(posedge clk);
        end
    end
endmodule
```

FSM Testbench Code



FSM Behavioral Simulation

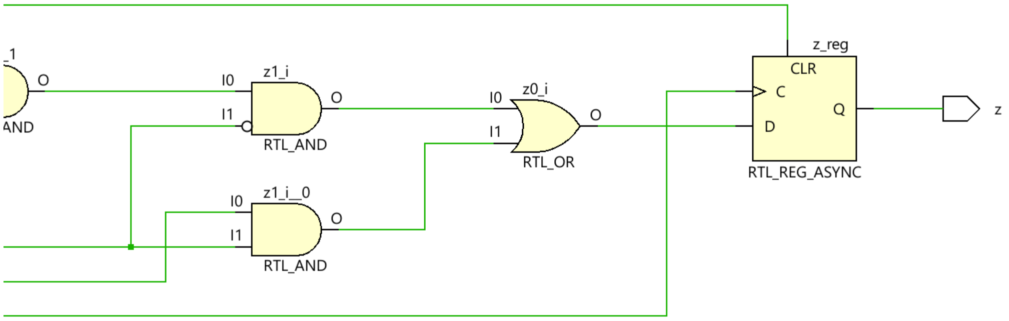
This design works as a Mealy Machine which resulted in the output to go high within three clock cycles rather than four. This is because of the way Mealy Machine gives output. Whenever the state registered the output changes without waiting the clock which means the device works as a “Three consecutive 1s or 0s”. This is rather a design choice.



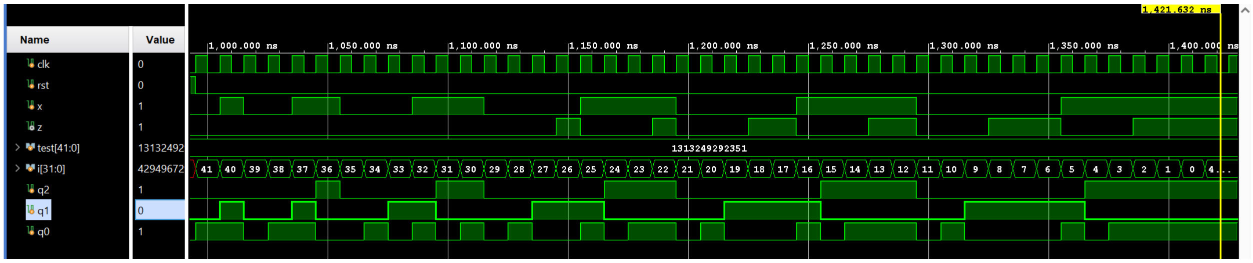
```
`timescale 1ns / 1ps

module FSM1(
    input clk,
    input rst,
    input x,
    output reg z
    );

    reg q0, q1, q2;

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            q0 <= 1'b0;
            q1 <= 1'b0;
            q2 <= 1'b0;
            z <= 1'b0;
        end else begin
            q2 <= (x & q1 & q0) | (x & q2);
            q1 <= (~q2 & ~q1 & q0) | (q1 & ~q0) | (x & ~q2 & ~q1);
            q0 <= (~q2 & ~q1 & ~q0) | (x & ~q1) | (x & ~q0);
            z <= (~x & q1 & ~q0) | (x & q2 & q0);
        end
    end
endmodule
```

Moore FSM Verilog Code

To convert a Mealy Machine into Moore Machine, the output must be synchronized with the clock. With the code above, the output z can be synchronized with the clock using “always” block. In the RTL Schematic, the DFF connected to the output z can be seen.

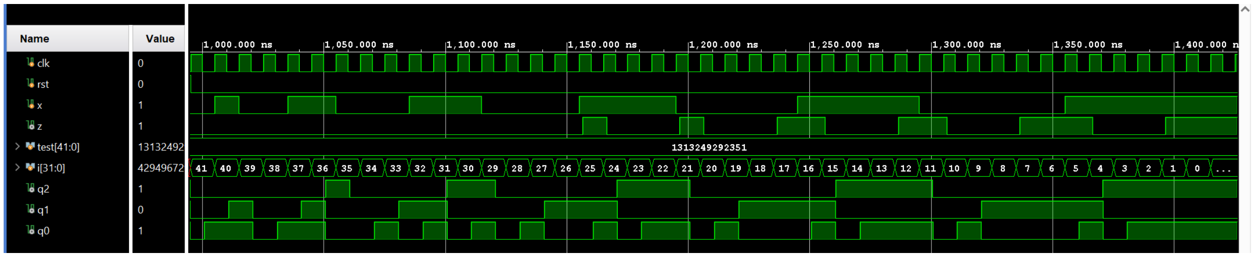


RTL Schematic of The Output Z (Moore)

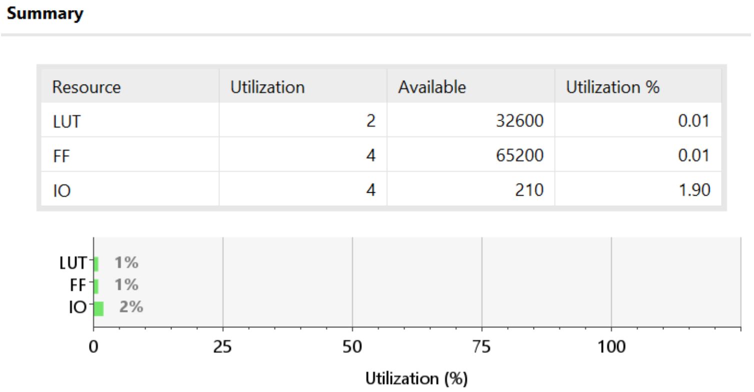


FSM Moore Behavioral Simulation

By making the design Moore, the output value synchronized with the clock which resulted an increasement in clock cycle of detecting the 1s and 0s. This is caused by the one clock cycle time requirement on the output due to the DFF. Instead of giving the output immediately after the 3rd clock cycle, while waiting for the clock for the output it receives the 4th state value correctly. As it can be seen on the simulation figure above, z only gives “1” when the input is either zero or one for four clock cycles instead of the three in the Mealy Machine. This time the design works as intended.



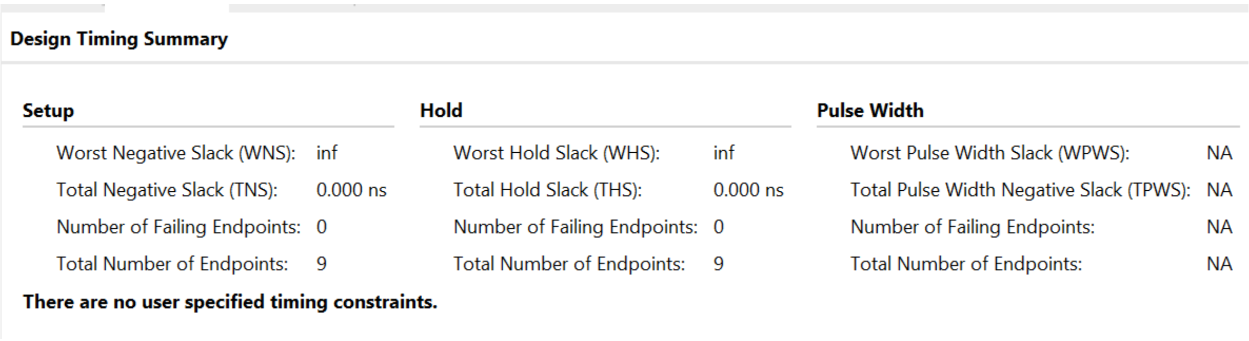
FSM Moore Post-Implementation Timing Simulation



Summary

Resource	Utilization	Available	Utilization %
LUT	2	32600	0.01
FF	4	65200	0.01
IO	4	210	1.90

FSM Moore Utilization Summary

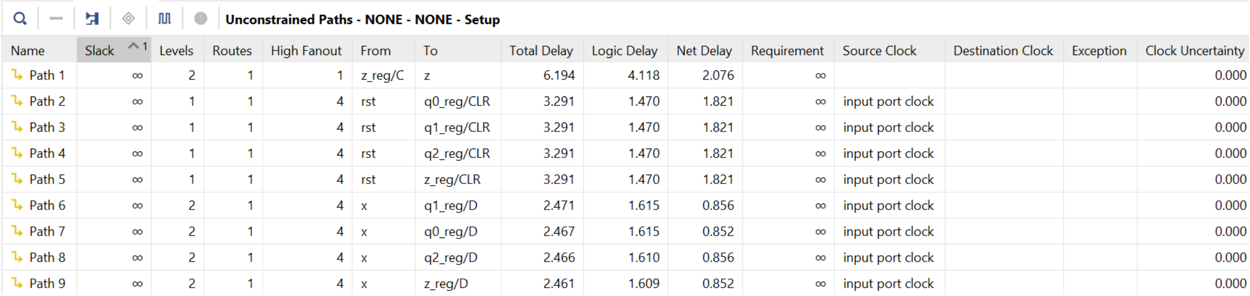


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	9	Total Number of Endpoints:	9	Total Number of Endpoints:	NA

There are no user specified timing constraints.

FSM Moore Design Timing Summary

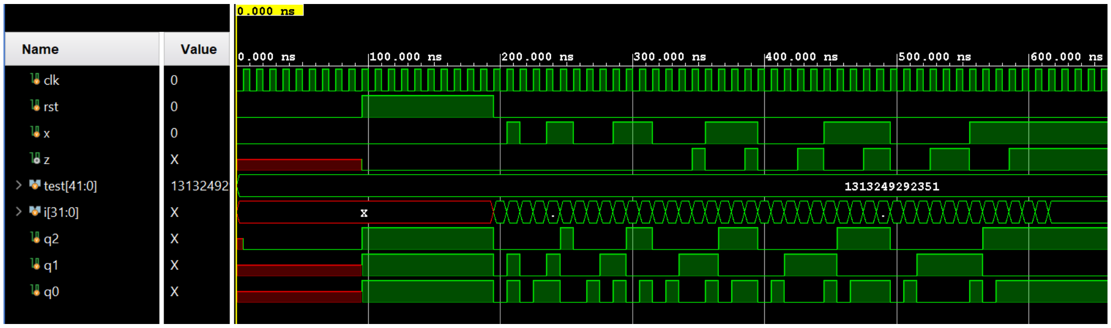


Unconstrained Paths - NONE - NONE - Setup

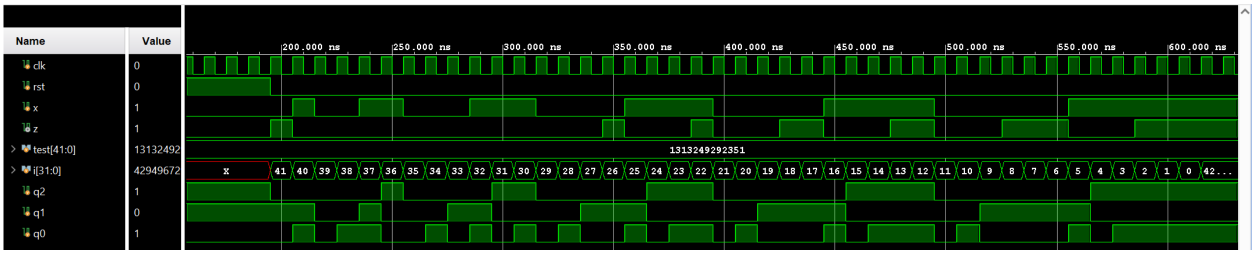
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	2	1	1	z_reg/C	z	6.194	4.118	2.076	∞				0.000
Path 2	∞	1	1	4	rst	q0_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 3	∞	1	1	4	rst	q1_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 4	∞	1	1	4	rst	q2_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 5	∞	1	1	4	rst	z_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 6	∞	2	1	4	x	q1_reg/D	2.471	1.615	0.856	∞	input port clock			0.000
Path 7	∞	2	1	4	x	q0_reg/D	2.467	1.615	0.852	∞	input port clock			0.000
Path 8	∞	2	1	4	x	q2_reg/D	2.466	1.610	0.856	∞	input port clock			0.000
Path 9	∞	2	1	4	x	z_reg/D	2.461	1.609	0.852	∞	input port clock			0.000

FSM Moore Path Delays - Setup

STUCKING IN UNDESIREBLE STATES:



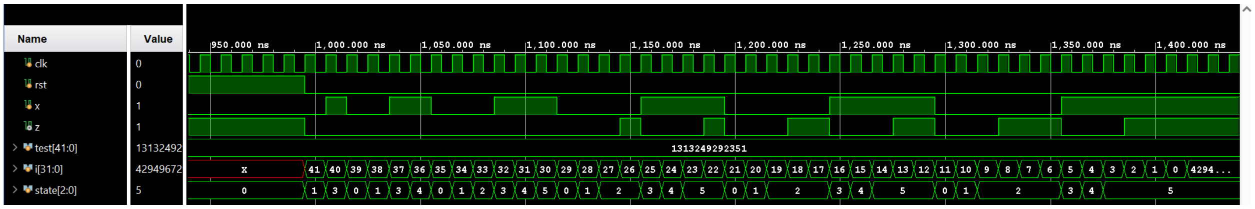
Behavioral Simulation for "111" state



Behavioral Simulation for "110" state

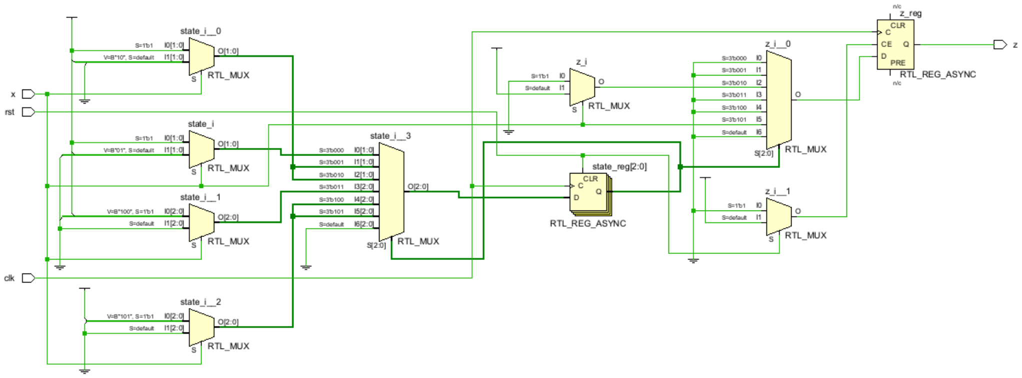
As it can be seen in the simulations initiating from the states “111” and “110” do not force device to be stuck in arbitrary states. However, the first z output value of the “110” state initiation is wrong.

BEHAVIORAL DESIGN:



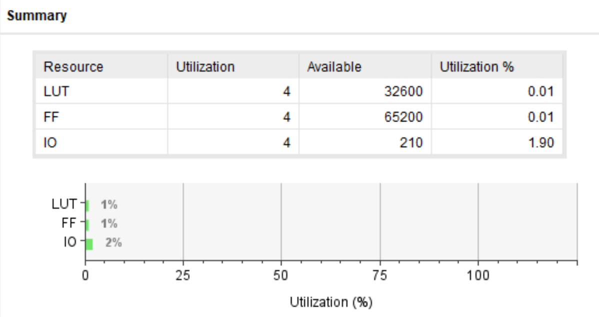
Behavioral FSM Behavioral Simulation

The simulations are the same for behavioral and Moore design. However, Moore design requires significantly more pre-computations. Behavioral design is better in terms of coding simplicity.



Behavioral FSM RTL Schematic

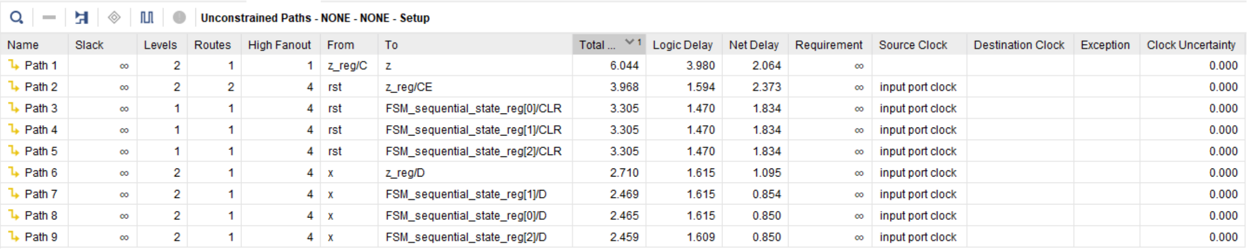
In terms of RTL placement, in behavioral design, instead of using numerous gates and paths, all the cases are represented with MUXs and registers. However, this does not mean that the technology schematic would decrease similarly.



Summary

Resource	Utilization	Available	Utilization %
LUT	4	32600	0.01
FF	4	65200	0.01
IO	4	210	1.90

Behavioral FSM Utilization Summary

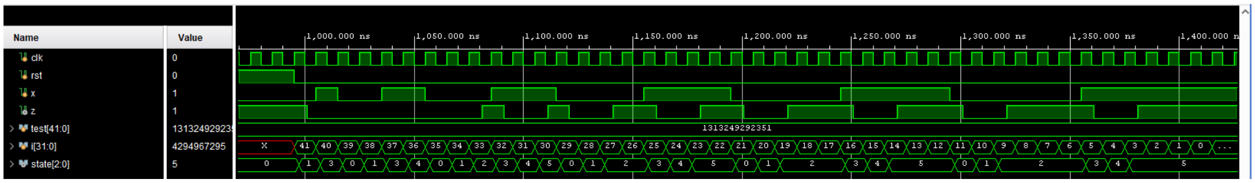


Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total ...	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	2	1	1	z_reg/C	z	6.044	3.980	2.064	∞				0.000
Path 2	∞	2	2	4	rst	z_reg/CE	3.968	1.594	2.373	∞	input port clock			0.000
Path 3	∞	1	1	4	rst	FSM_sequential_state_reg[0]/CLR	3.305	1.470	1.834	∞	input port clock			0.000
Path 4	∞	1	1	4	rst	FSM_sequential_state_reg[1]/CLR	3.305	1.470	1.834	∞	input port clock			0.000
Path 5	∞	1	1	4	rst	FSM_sequential_state_reg[2]/CLR	3.305	1.470	1.834	∞	input port clock			0.000
Path 6	∞	2	1	4	x	z_reg/D	2.710	1.615	1.095	∞	input port clock			0.000
Path 7	∞	2	1	4	x	FSM_sequential_state_reg[1]/D	2.469	1.615	0.854	∞	input port clock			0.000
Path 8	∞	2	1	4	x	FSM_sequential_state_reg[0]/D	2.465	1.615	0.850	∞	input port clock			0.000
Path 9	∞	2	1	4	x	FSM_sequential_state_reg[2]/D	2.459	1.609	0.850	∞	input port clock			0.000

Behavioral FSM Path Delays - Setup

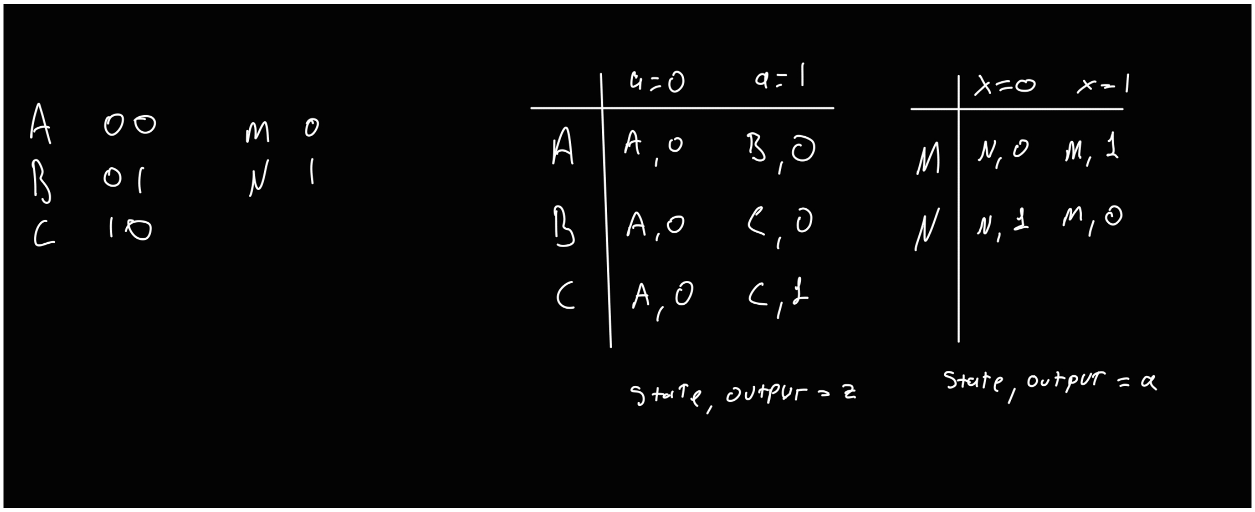
As it can be seen in the Utilization Summaries, the LUT usage is increase to 4 from 2 in the behavioral design. However, the longest path delay is decreased considerably. Beside the increase in “rst to z_reg” path, all the other paths decreased a little.



Name	Value
clk	0
rst	0
x	1
z	1
test[41:0]	13132492923
i[31:0]	4294967295
state[2:0]	5

Behavioral FSM Post-Implementation Timing Simulation

2. Design with Divided State Diagrams



A 00
B 01
C 10

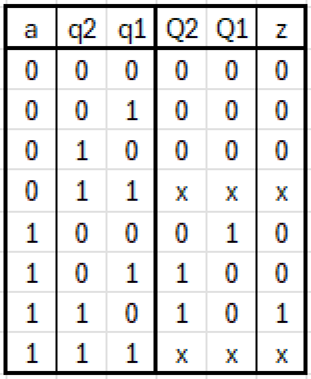
M 0
N 1

	a=0	a=1
A	A,0	B,0
B	A,0	C,0
C	A,0	C,1

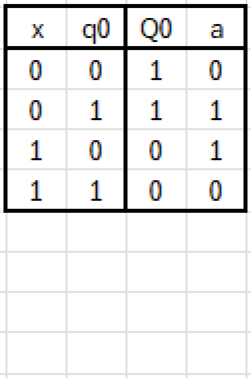
StaTe, output = z

	x=0	x=1
M	N,0	M,1
N	N,1	M,0

State, output = a



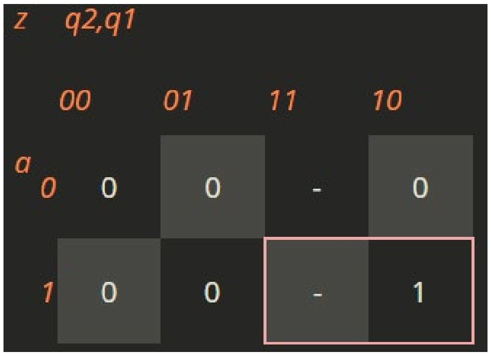
a	q2	q1	Q2	Q1	z
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	x	x	x
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	x	x	x



x	q0	Q0	a
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	0

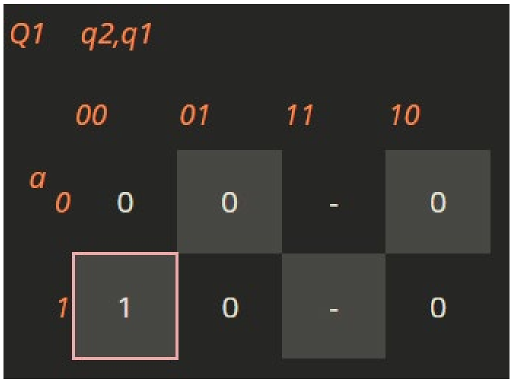
$$a = x \text{ ^ } q0$$

$$Q0 = !x$$



z	q2,q1			
a	00	01	11	10
0	0	0	-	0
1	0	0	-	1

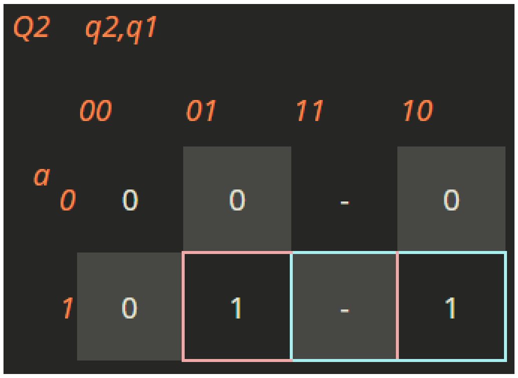
$$z(a, q2, q1) \ = \ aq2$$



Q1 q2,q1

a	00	01	11	10
0	0	0	-	0
1	1	0	-	0

$$Q1(a, q2, q1) = aq2'q1'$$



Q2 q2,q1

a	00	01	11	10
0	0	0	-	0
1	0	1	-	1

$$Q2(a, q2, q1) = aq1 + aq2$$

REDUCED EQUATIONS

$$a = x \,\hat{}\, q0$$

$$Q0 = !\,x$$

$$Q1 = a \,\&\, (!\,q1) \,\&\, (!\,q2)$$

$$Q2 = (q1 \,\&\, a) \mid (a \,\&\, q2)$$

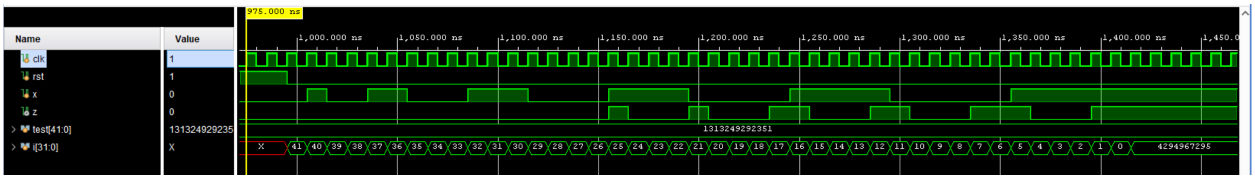
$$Z = a \,\&\, q2$$

```
`timescale 1ns / 1ps
module FSM2(
    input clk,
    input rst,
    input x,
    output reg z
    );

    reg q0, q1, q2;
    wire a, Z;
    wire Q0, Q1, Q2;

    assign a  = x ^ q0;
    assign Q0 = !x;
    assign Q1 = a & (!q1) & (!q2);
    assign Q2 = (q1 & a) | (a & q2);
    assign Z  = a & q2;

    always @(posedge clk or posedge rst)
begin
        if (rst) begin
            q2 <= 1'b0;
            q1 <= 1'b0;
            q0 <= 1'b0;
            z  <= 1'b0;
        end else begin
            q2 <= Q2;
            q1 <= Q1;
            q0 <= Q0;
            z  <= Z;
        end
    end
endmodule
```

FSM2 Verilog Code

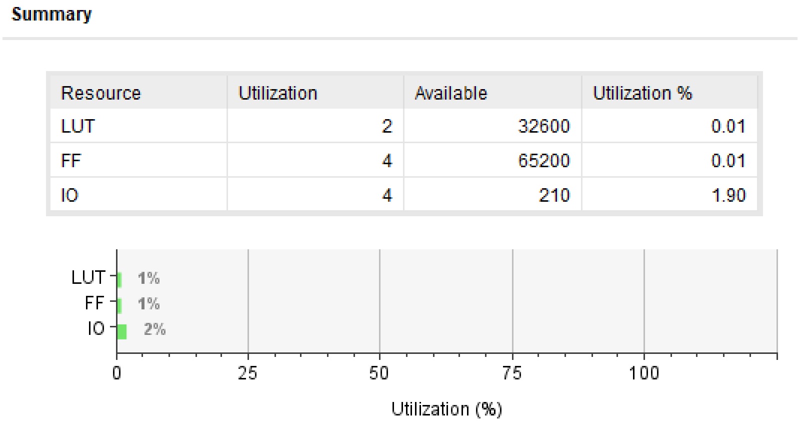
```
`timescale 1ns / 1ps

module FSM2_tb();

    reg   clk = 0;
    reg   rst = 0;
    reg   x   = 0;
    wire  z;

    FSM2 dUT(
        .clk(clk),
        .rst(rst),
        .x(x),
        .z(z)
    );

    always #5 clk = ~clk;

    reg [41:0] test =
42'b0100_1100_0111_0000_1111_0000_0111_1100_0000_1111_11;
    integer i;

    initial
    begin
        repeat(80) @(posedge clk);
        rst = 1;
        repeat(20) @(posedge clk);
        rst = 0;

        for(i=41; i>=0; i=i-1)
        begin
            x = test[i];
            @(posedge clk);
        end
    end
endmodule
```

FSM2 Testbench Code



Name	Value
clk	1
rst	1
x	0
z	0
test[41:0]	131324929235
i[31:0]	X

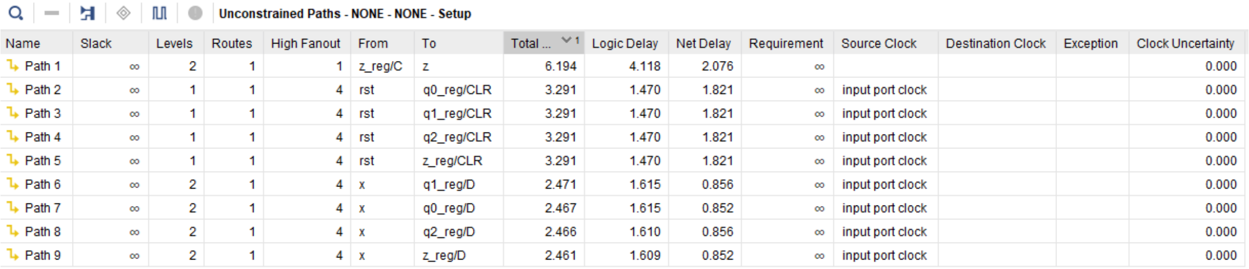
FSM2 Behavioral Simulation



Summary

Resource	Utilization	Available	Utilization %
LUT	2	32600	0.01
FF	4	65200	0.01
IO	4	210	1.90

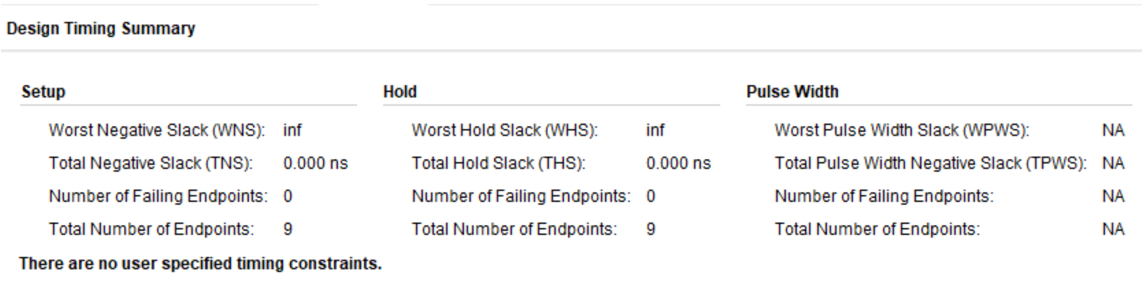
Utilization Summary



Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total ... ˅ 1	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	2	1	1	z_reg/C	z	6.194	4.118	2.076	∞				0.000
Path 2	∞	1	1	4	rst	q0_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 3	∞	1	1	4	rst	q1_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 4	∞	1	1	4	rst	q2_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 5	∞	1	1	4	rst	z_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 6	∞	2	1	4	x	q1_reg/D	2.471	1.615	0.856	∞	input port clock			0.000
Path 7	∞	2	1	4	x	q0_reg/D	2.467	1.615	0.852	∞	input port clock			0.000
Path 8	∞	2	1	4	x	q2_reg/D	2.466	1.610	0.856	∞	input port clock			0.000
Path 9	∞	2	1	4	x	z_reg/D	2.461	1.609	0.852	∞	input port clock			0.000

FSM2 Path Delays

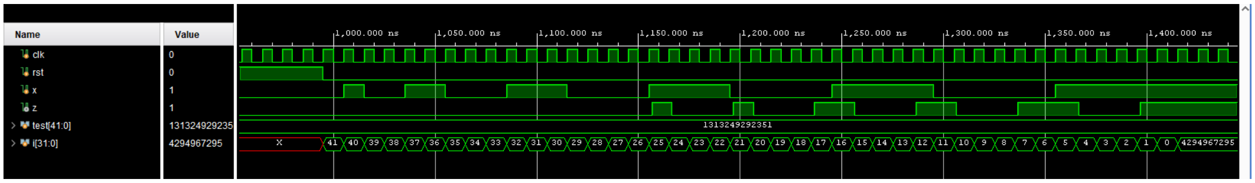


Design Timing Summary

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	9	Total Number of Endpoints:	9	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Timing Summary

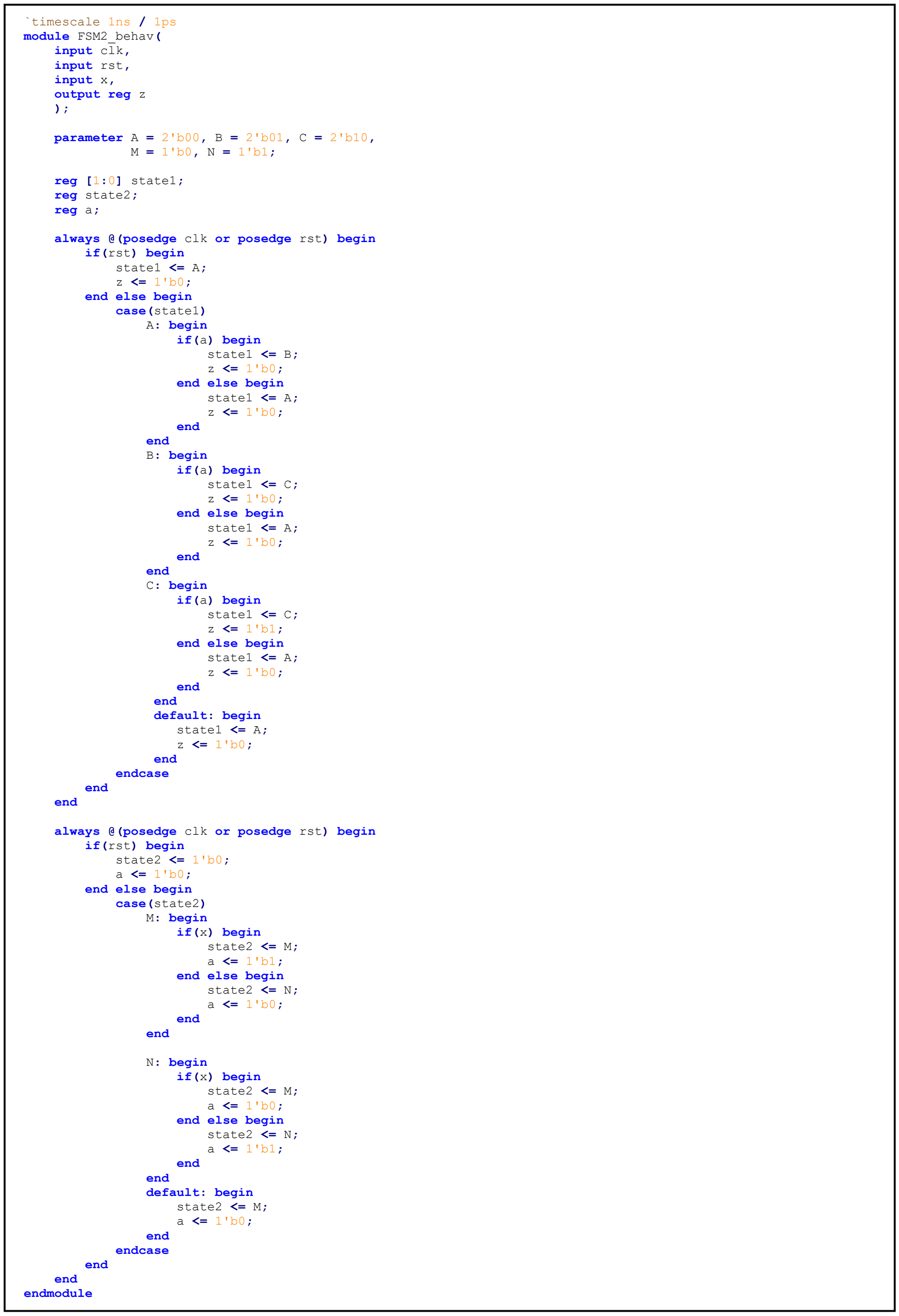


Name	Value
clk	0
rst	0
x	1
z	1
test[41:0]	131324929235
i[31:0]	4294967295

FSM2 Post-Implementatiın Timing Simulation

Yusuf Tekin
040200043

FSM2 BEHAVIORAL:



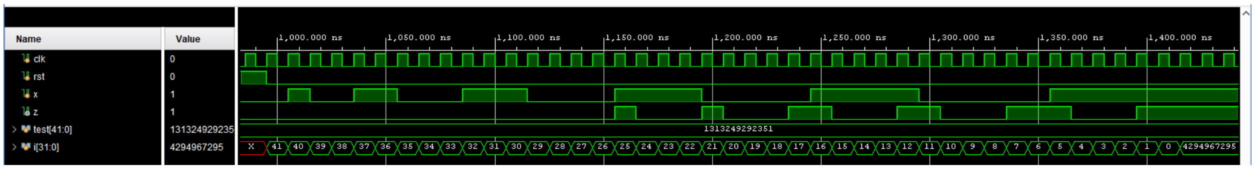
```
`timescale 1ns / 1ps
module FSM2_behav(
    input clk,
    input rst,
    input x,
    output reg z
    );

    parameter A = 2'b00, B = 2'b01, C = 2'b10,
              M = 1'b0, N = 1'b1;

    reg [1:0] state1;
    reg state2;
    reg a;

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            state1 <= A;
            z <= 1'b0;
        end else begin
            case(state1)
                A: begin
                    if(a) begin
                        state1 <= B;
                        z <= 1'b0;
                    end else begin
                        state1 <= A;
                        z <= 1'b0;
                    end
                end
                B: begin
                    if(a) begin
                        state1 <= C;
                        z <= 1'b0;
                    end else begin
                        state1 <= A;
                        z <= 1'b0;
                    end
                end
                C: begin
                    if(a) begin
                        state1 <= C;
                        z <= 1'b1;
                    end else begin
                        state1 <= A;
                        z <= 1'b0;
                    end
                 end
                 default: begin
                    state1 <= A;
                    z <= 1'b0;
                 end
            endcase
        end
    end

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            state2 <= 1'b0;
            a <= 1'b0;
        end else begin
            case(state2)
                M: begin
                    if(x) begin
                        state2 <= M;
                        a <= 1'b1;
                    end else begin
                        state2 <= N;
                        a <= 1'b0;
                    end
                end

                N: begin
                    if(x) begin
                        state2 <= M;
                        a <= 1'b0;
                    end else begin
                        state2 <= N;
                        a <= 1'b1;
                    end
                end
                default: begin
                    state2 <= M;
                    a <= 1'b0;
                end
            endcase
        end
    end
endmodule
```

FSM2 Behavioral Verilog Code

Yusuf Tekin
040200043

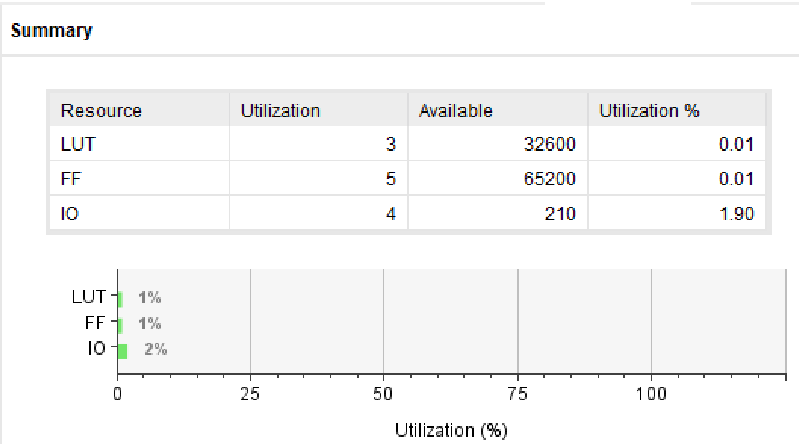
```
`timescale 1ns / 1ps

module FSM2_behav_tb();

    reg   clk = 0;
    reg   rst = 0;
    reg   x   = 0;
    wire  z;

    FSM2_behav DUT(
        .clk(clk),
        .rst(rst),
        .x(x),
        .z(z)
    );

    always #5 clk = ~clk;

    reg [41:0] test =
42'b0100_1100_0111_0000_1111_0000_0111_1100_0000_1111_11;
    integer i;

    initial
    begin
        repeat(80) @(posedge clk);
        rst = 1;
        repeat(20) @(posedge clk);
        rst = 0;

        for(i=41; i>=0; i=i-1)
        begin
            x = test[i];
            @(posedge clk);
        end
    end
endmodule
```

FSM2 Behavioral Testbench Code



Name	Value
clk	0
rst	0
x	1
z	1
test[41:0]	131324929235
i[31:0]	4294967295

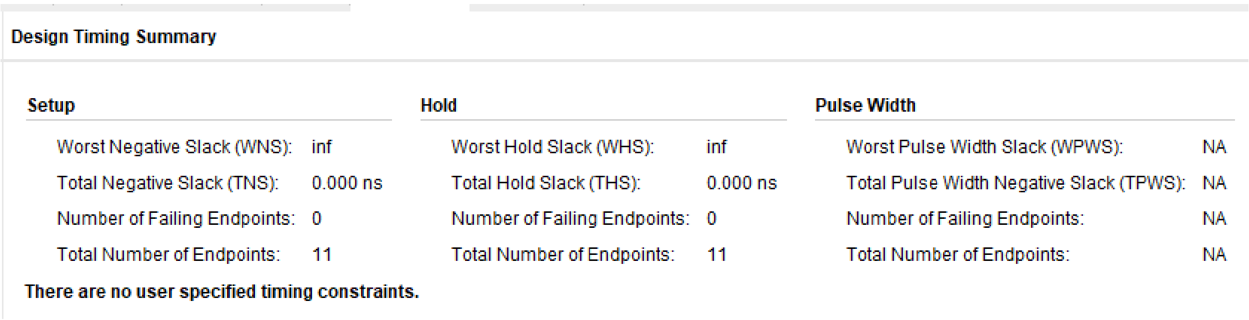
FSM2 Behavioral - Behavioral Simulation



Summary

Resource	Utilization	Available	Utilization %
LUT	3	32600	0.01
FF	5	65200	0.01
IO	4	210	1.90

FSM2 Behavioral Utilization Summary



Design Timing Summary

Setup

Worst Negative Slack (WNS): inf
Total Negative Slack (TNS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 11

Hold

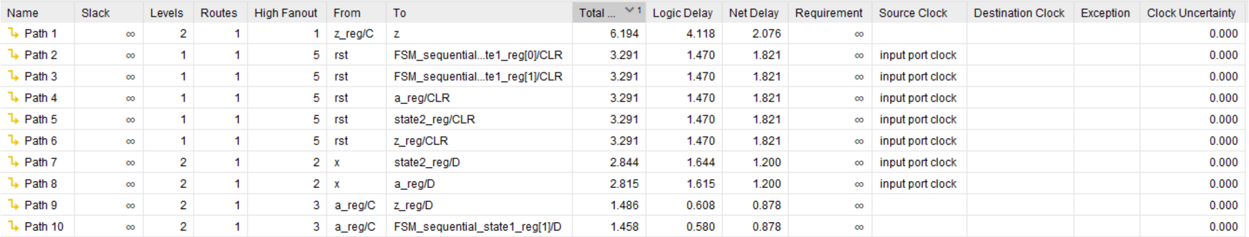
Worst Hold Slack (WHS): inf
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 11

Pulse Width

Worst Pulse Width Slack (WPWS): NA
Total Pulse Width Negative Slack (TPWS): NA
Number of Failing Endpoints: NA
Total Number of Endpoints: NA

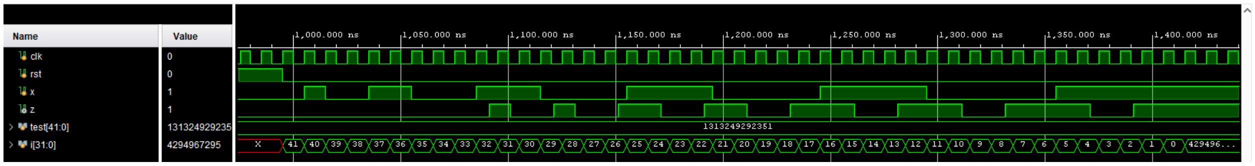
There are no user specified timing constraints.

FSM2 Behavioral Timing Summary



Name	Slack	Levels	Routes	High Fanout	From	To	Total ...	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	2	1	1	z_reg/C	z	6.194	4.118	2.076	∞				0.000
Path 2	∞	1	1	5	rst	FSM_sequential...te1_reg[0]/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 3	∞	1	1	5	rst	FSM_sequential...te1_reg[1]/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 4	∞	1	1	5	rst	a_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 5	∞	1	1	5	rst	state2_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 6	∞	1	1	5	rst	z_reg/CLR	3.291	1.470	1.821	∞	input port clock			0.000
Path 7	∞	2	1	2	x	state2_reg/D	2.844	1.644	1.200	∞	input port clock			0.000
Path 8	∞	2	1	2	x	a_reg/D	2.815	1.615	1.200	∞	input port clock			0.000
Path 9	∞	2	1	3	a_reg/C	z_reg/D	1.486	0.608	0.878	∞				0.000
Path 10	∞	2	1	3	a_reg/C	FSM_sequential_state1_reg[1]/D	1.458	0.580	0.878	∞				0.000

FSM2 behavioral Path Delays - Setup



Name	Value
clk	0
rst	0
x	1
z	1
test[41:0]	131324929235
i[31:0]	4294967295

FSM2 Behavioral Post-Implementation Timing Simulation

To compare two different models of FSM2, it can be seen on the utilization summaries that both the LUT and FF usage is increased by 1 for behavioral model. These additions are caused by the registers that contain states. However, besides the addition of the two new paths, the existing path delays have not changed at all.

To conclude, in FSM2 the behavioral model's coding and design is simpler but it takes more space than the first design. However, unlike FSM1, making the design behavioral does not affect the path delays of FSM2.

References

1- https://www.allaboutcircuits.com/technical-articles/encoding-the-states-of-a-finite-state-machine-vhdl/