DIGITAL SYSTEM DESIGN APPLICATION

EHB436E CRN: 11280

Salih Ömer Ongün
040220780

Experiment 7

Salih Ömer Ongün
040220780

Encoding Methods

In Binary encoding, numbers are coded in ascending binary order. Therefore, it is simply for understanding. It uses more than one flip flop simultaneously so that it use less flip-flop for all coding. It is durable against to noise and it have advantages in terms of memory usage.

State	Binary Encoding
State A	000
State B	001
State C	010

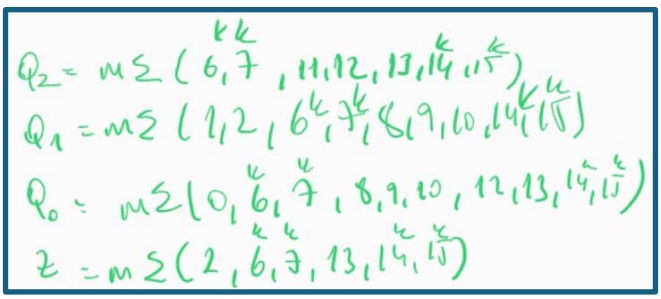
In Gray coding, bit changes occur during state transitions. It is used in Karnaugh maps. It has 00,01,11,10 encoding in Karnaugh maps. It has different advantages. It changes one bit. Therefore, it has less power consumption. Gary encoding uses protection for asynchronous outputs from glitches. It is another advantage of Gray encoding.

State	Gray Encoding
State A	000
State B	001
State C	011

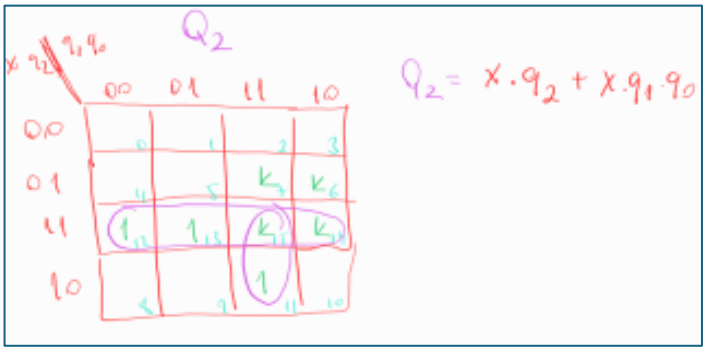
In One-Hot encoding, states have one logic 1 bit. Other bits are logic 0. Logic 1 bit is shifted left during state transition. One-hot encoding makes these combinational circuits simpler, which reduces propagation delay, which in turn makes the FSM compatible with higher clock frequencies. One-hot encoding increases the number of FFs and it increase memory usage.***[1]***

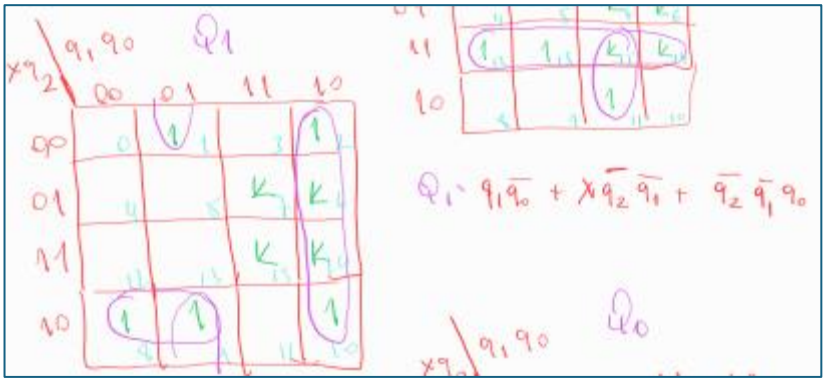
State	One- Hot Encoding
State A	001
State B	010
State C	100

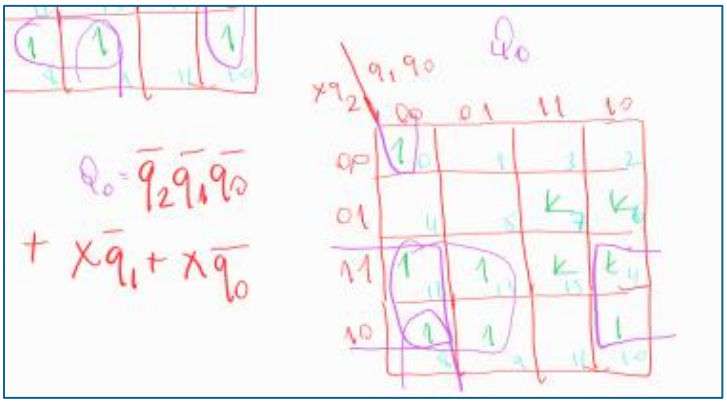
CONSECUTIVE 1 OR 0 MODEL

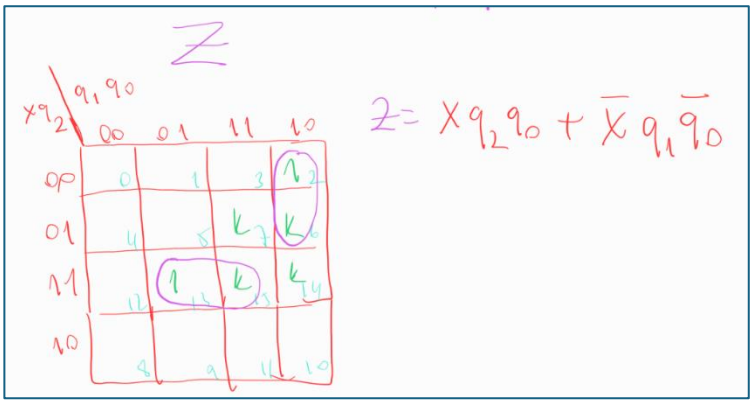


Salih Ömer Ongün
040220780









Design Source

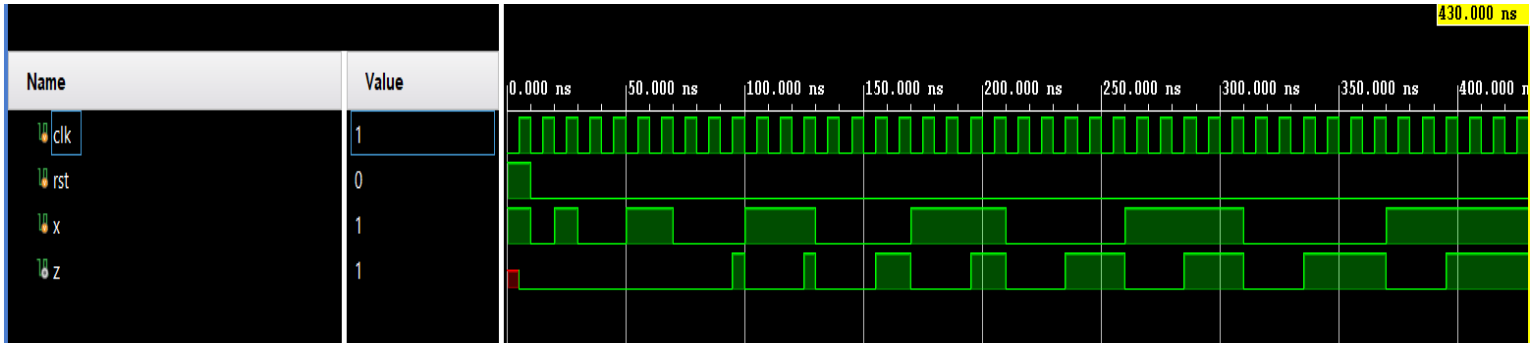
```
module consec
(
    input clk,
    input rst,
    input x,
    output z
);
    reg q0,q1,q2;
    wire Q0,Q1,Q2;
    reg z_reg;
    assign Q2 = (x & q2) | (x & q1 & q0);
    assign Q1 = (q1 & ~q0) | (x & ~q2 & ~q1) | (~q2 & ~q1 & q0);
    assign Q0 = (x & ~q1) | (~q0 & ~q1 & ~q2) | (x & ~q0);
    assign z = (x & q2 & q0) | (~x & q1 & ~q0);
    always @(posedge clk) begin
        if(rst == 1'b1) begin
            q2 <= 1'b0;
            q1 <= 1'b0;
            q0 <= 1'b0;
        end
        else begin
            q2<= Q2;
            q1<= Q1;
            q0<= Q0;
        end
    end
endmodule
```

Mealy machine type code

Simulation Source

```
module consec_tb();

    reg clk = 1'b0;
    reg rst = 1'b1;
    reg x = 1'b1;
    wire z;
    reg [41:0] test = 42'b010011000111000011110000011111000000111111;
    integer i= 41;
    consec uut
    (
        .clk(clk),
        .rst(rst),
        .x(x),
        .z(z)
    );

    always begin
        #5 clk = ~clk;
    end

    initial begin
        #10;
        rst = 1'b0;
        for(i = 41; i>=0; i = i-1) begin
            x = test[i];
            #10;
        end
        $finish;
    end
endmodule
```

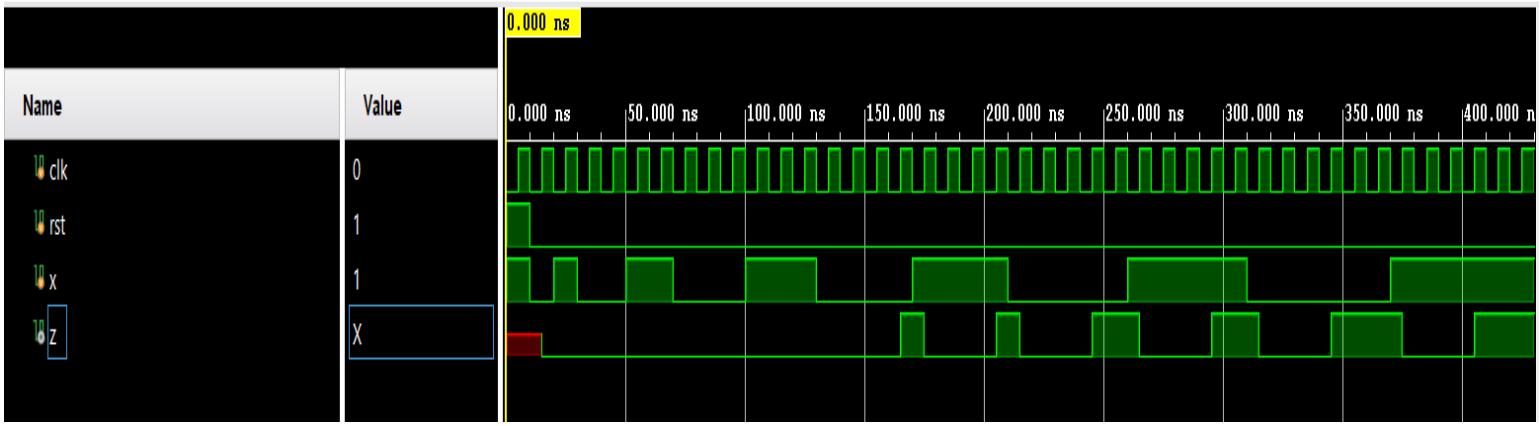
Simulation Waveform



z obtain logic 1 value after three or more consecutive 1s or 0s. It should obtain logic 1 after four consecutive 1s or 0s. It has faulty 0 and faulty 1 values in three consecutive 1s and 0s.

```
module consec
(
    input clk,
    input rst,
    input x,
    output z
);
    reg q0,q1,q2;
    wire Q0,Q1,Q2,z_moore;
    reg z_reg;
    assign Q2 = (x & q2) | (x & q1 & q0);
    assign Q1 = (q1 & ~q0) | (x & ~q2 & ~q1) | (~q2 & ~q1 & q0);
    assign Q0 = (x & ~q1) | (~q0 & ~q1 & ~q2) | (x & ~q0);
    assign z_moore = (x & q2 & q0) | (~x & q1 & ~q0);
    always @(posedge clk) begin
        if(rst == 1'b1) begin
            q2 <= 1'b0;
            q1 <= 1'b0;
            q0 <= 1'b0;
        end
        else begin
            q2<= Q2;
            q1<= Q1;
            q0<= Q0;
            z_reg<= z_moore;
        end
    end
    assign z = z_reg;
endmodule
```

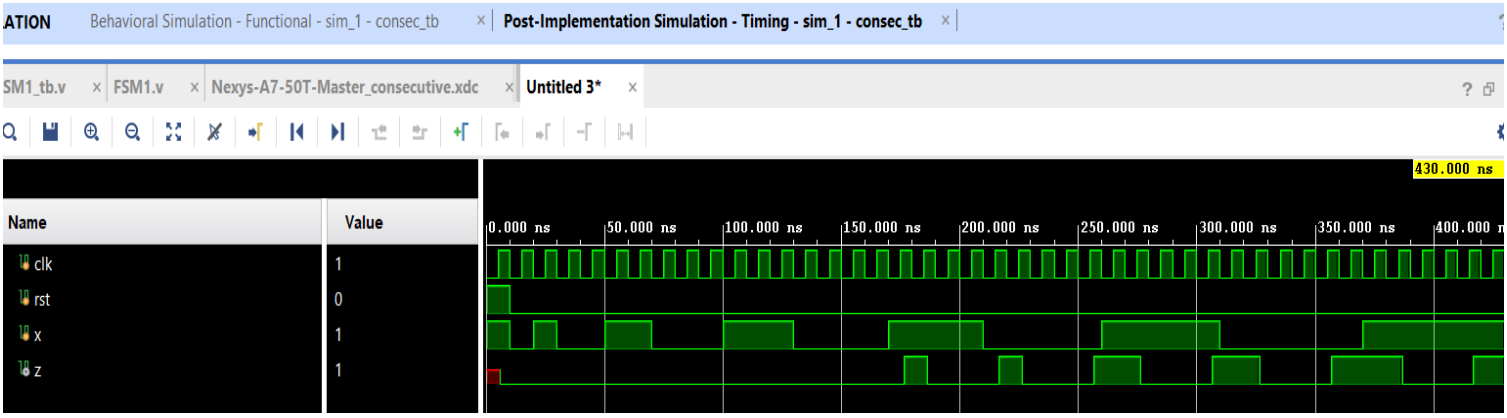
Moore machine type code



Moore type machine gives correct results. z output obtains logic 1 value after four consecutive 1s or 0s.

Salih Ömer Ongün
040220780

Post -Implementation Timing Simulation



Post -Implementation Timing Simulation have same results with Moore type simulation.

Utilization Report

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	4	32600	0.01
FF	4	65200	0.01
IO	4	210	1.90
BUFG	1	32	3.13

Timing Report

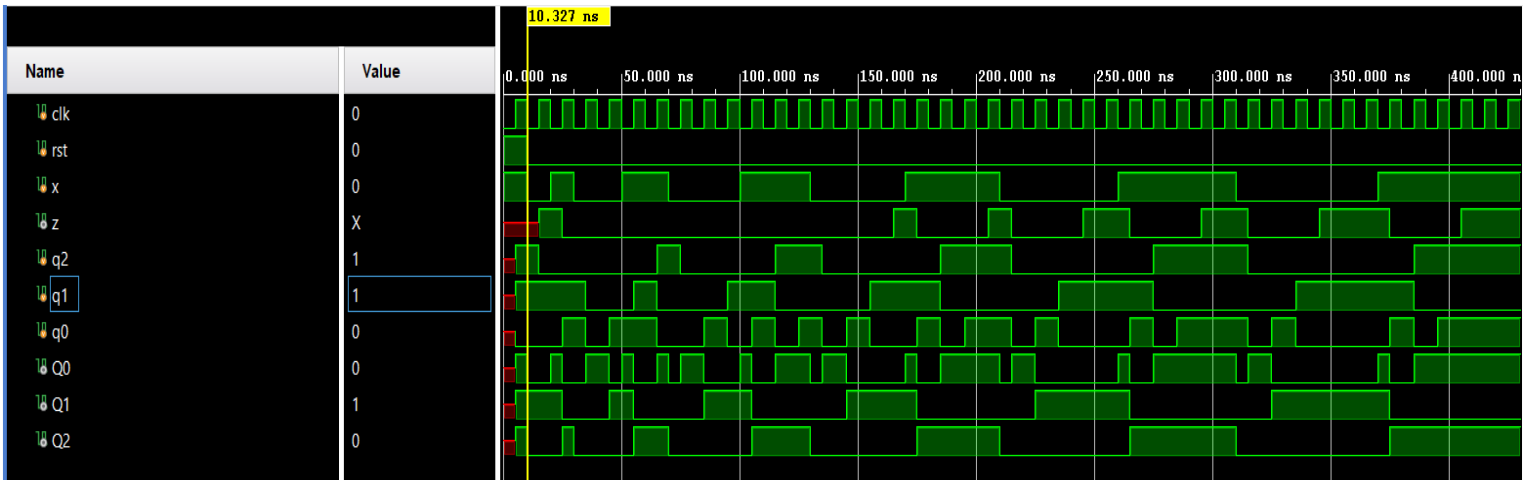
Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	2	1	1	z_reg_reg/C	z	6.044	3.980	2.064	∞	
Path 2	∞	2	2	4	rst	z_reg_reg/CE	4.022	1.603	2.419	∞	input port clock
Path 3	∞	2	1	4	rst	q2_reg/D	3.263	1.631	1.632	∞	input port clock
Path 4	∞	2	1	4	rst	q1_reg/D	3.237	1.603	1.634	∞	input port clock
Path 5	∞	2	1	4	rst	q0_reg/D	3.235	1.603	1.632	∞	input port clock
Path 6	∞	2	1	4	x	z_reg_reg/D	2.710	1.615	1.095	∞	input port clock

Unconstrained Paths - NONE - NONE - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 7	∞	2	1	4	q2_reg/C	q1_reg/D	0.296	0.227	0.069	-∞	
Path 8	∞	2	1	4	q0_reg/C	z_reg_reg/D	0.305	0.186	0.119	-∞	
Path 9	∞	2	1	4	q0_reg/C	q2_reg/D	0.361	0.183	0.178	-∞	
Path 10	∞	2	1	4	q0_reg/C	q0_reg/D	0.364	0.186	0.178	-∞	
Path 11	∞	2	2	4	rst	z_reg_reg/CE	1.186	0.292	0.894	-∞	input port clock
Path 12	∞	2	1	1	z_reg_reg/C	z	1.855	1.366	0.489	-∞	

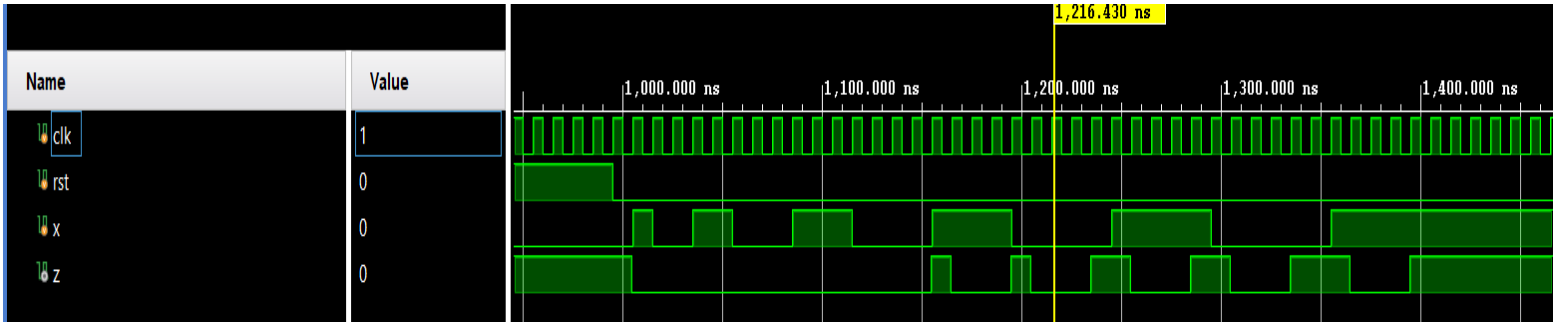
Undesirable States



I give 110 state for initialization. Circuit does not stuck any state. It just changes the order of events.

Behavioral Model

Post -Implementation Timing Simulation



It gives correct results like previous designs.

Utilization Report

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	4	32600	0.01
FF	4	65200	0.01
IO	4	210	1.90
BUFG	1	32	3.13

Salih Ömer Ongün
040220780

Timing Report

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	2	1	1	z_reg/C	z	4.763	3.095	1.668	∞	
Path 2	∞	2	2	4	rst	z_reg/CE	2.855	1.090	1.765	∞	input port clock
Path 3	∞	1	1	4	rst	FSM_sequential_state_reg[0]/CLR	2.213	0.966	1.248	∞	input port clock
Path 4	∞	1	1	4	rst	FSM_sequential_state_reg[1]/CLR	2.213	0.966	1.248	∞	input port clock
Path 5	∞	1	1	4	rst	FSM_sequential_state_reg[2]/CLR	2.213	0.966	1.248	∞	input port clock
Path 6	∞	2	1	4	x	z_reg/D	2.176	1.094	1.082	∞	input port clock
Path 7	∞	2	1	4	x	FSM_sequential_state_reg[1]/D	1.963	1.094	0.869	∞	input port clock
Path 8	∞	2	1	4	x	FSM_sequential_state_reg[0]/D	1.959	1.094	0.865	∞	input port clock
Path 9	∞	2	1	4	x	FSM_sequential_state_reg[2]/D	1.953	1.088	0.865	∞	input port clock

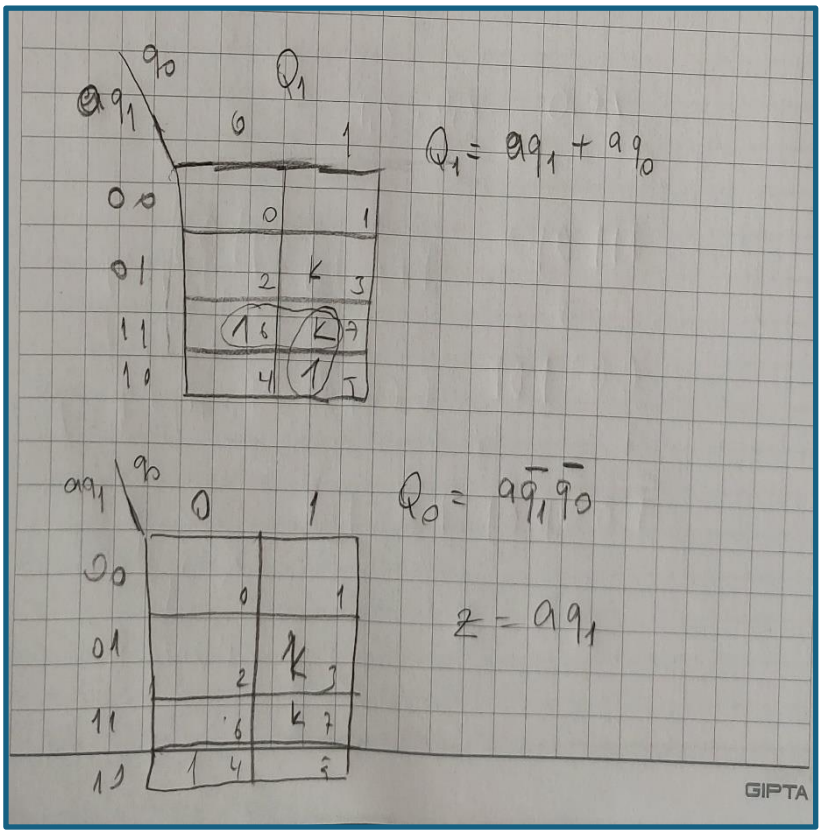
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 10	∞	2	1	4	FSM_sequential_state_reg[0]/C	z_reg/D	0.294	0.186	0.108	-∞
Path 11	∞	2	1	4	FSM_sequential_state_reg[2]/C	FSM_sequential_state_reg[1]/D	0.302	0.227	0.075	-∞
Path 12	∞	2	1	4	FSM_sequential_state_reg[0]/C	FSM_sequential_state_reg[2]/D	0.378	0.183	0.195	-∞
Path 13	∞	2	1	4	FSM_sequential_state_reg[0]/C	FSM_sequential_state_reg[0]/D	0.381	0.186	0.195	-∞
Path 14	∞	1	1	4	rst	FSM_sequential_state_reg[0]/CLR	0.675	0.195	0.481	-∞
Path 15	∞	1	1	4	rst	FSM_sequential_state_reg[1]/CLR	0.675	0.195	0.481	-∞
Path 16	∞	1	1	4	rst	FSM_sequential_state_reg[2]/CLR	0.675	0.195	0.481	-∞
Path 17	∞	2	2	4	rst	z_reg/CE	0.886	0.240	0.646	-∞
Path 18	∞	2	1	1	z_reg/C	z	1.616	1.297	0.319	-∞

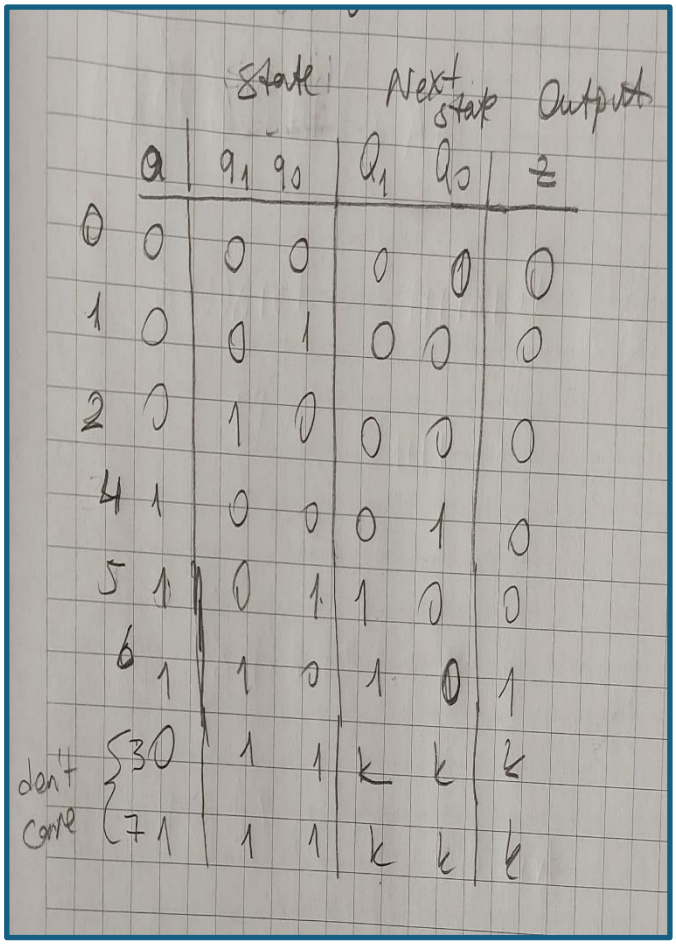
Behavioral model and my previous model have the same resource usage. However, behavioral model has less delay results than previous design. Max delay of previous design is 6.044 ns but max delay of behavioral design is 4.763ns. There is a small difference in terms of delays.

Salih Ömer Ongün
040220780

DIVIDED MODEL

State	Binary Encoding
A	0 0
B	0 1
C	1 0





Design Source

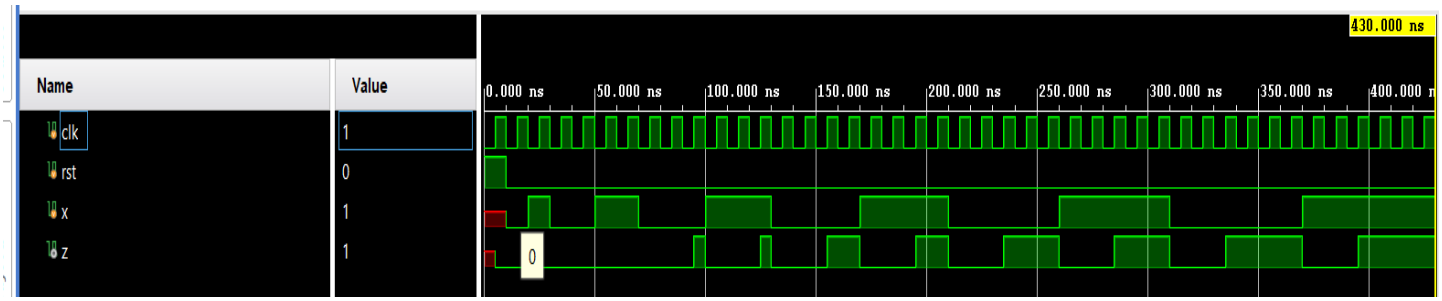
```
module divide
(
    input clk,
    input rst,
    input x,
    output z
);
    reg q0,q1,x_cs,z_reg;
    wire Q0,Q1,a;
    assign Q1 = (a & q1) | (a & q0) ;
    assign Q0 = (a & ~q1 & ~q0);
    assign a = ~(x_cs ^ x);
    assign z = (a & q1);
    always @(posedge clk) begin
        if(rst == 1'b1) begin
            q1 <= 1'b0;
            q0 <= 1'b0;
        end
        else begin
            q1 <= Q1;
            q0 <= Q0;
            x_cs<= x;
        end
    end
endmodule
```

Mealy type code

Simulation Source

```
module divide_tb();

    reg clk = 1'b0;
    reg rst = 1'b1;
    reg x;
    wire z;
    reg [41:0] test = 42'b010011000111000011110000011111000000111111;
    integer i= 41;
    divide uut
    (
        .clk(clk),
        .rst(rst),
        .x(x),
        .z(z)
    );

    always begin
        #5 clk = ~clk;
    end

    initial begin
        #10;
        rst = 1'b0;
        for(i = 41; i>=0; i = i-1) begin
            x = test[i];
            #10;
        end
        $finish;
    end
endmodule
```

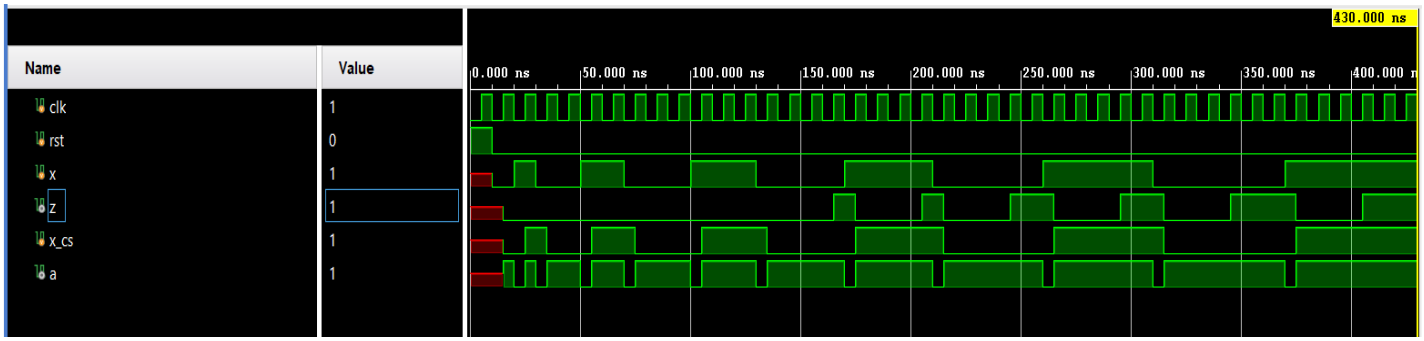
Simulation Waveform



z obtain logic 1 value after three or more consecutive 1s or 0s. It should obtain logic 1 after four consecutive 1s or 0s. It has faulty 0 and faulty 1 values in three consecutive 1s and 0s.

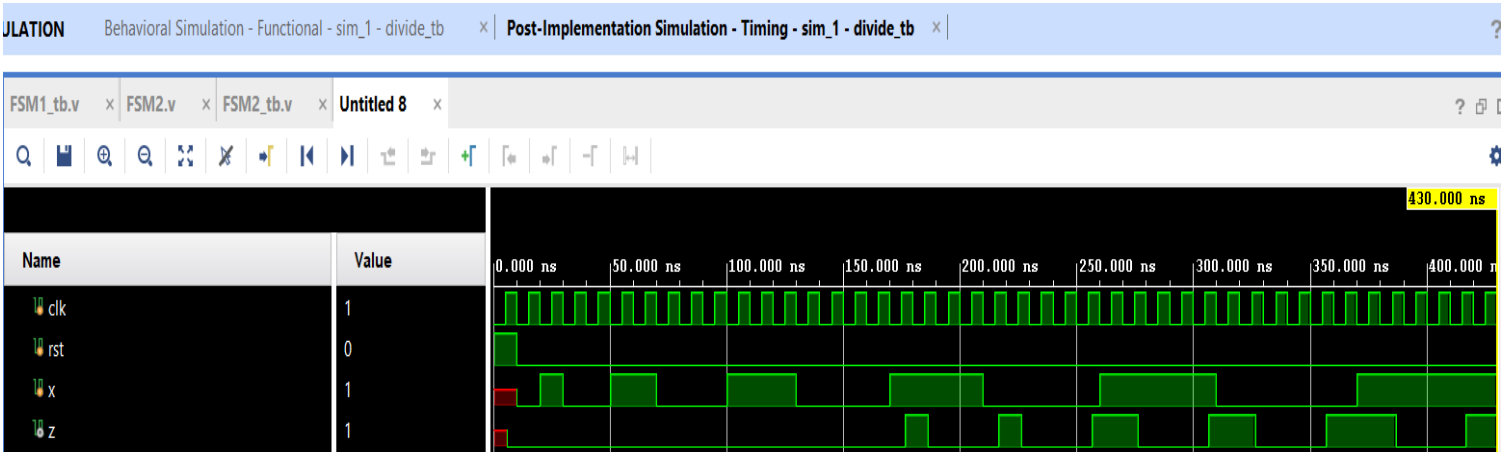
```
module divide
(
    input clk,
    input rst,
    input x,
    output z
);
    reg q0,q1,x_cs,z_reg;
    wire Q0,Q1,a,z_moore;
    assign Q1 = (a & q1) | (a & q0) ;
    assign Q0 = (a & ~q1 & ~q0);
    assign a = ~(x_cs ^ x);
    assign z_moore = (a & q1);
    always @(posedge clk) begin
        if(rst == 1'b1) begin
            q1 <= 1'b0;
            q0 <= 1'b0;

        end
        else begin
            q1 <= Q1;
            q0 <= Q0;
            x_cs<= x;
            z_reg<= z_moore;
        end
    end
    assign z = z_reg;
endmodule
```

Moore type code



Moore type machine gives correct results. z output obtains logic 1 value after four consecutive 1s or 0s.

Post -Implementation Timing Simulation



Post -Implementation Timing Simulation have same results with Moore type simulation.

Utilization Report

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	3	32600	0.01
FF	4	65200	0.01
IO	4	210	1.90
BUFG	1	32	3.13

Salih Ömer Ongün
040220780

Timing Report

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	2	1	1	z_reg_reg/C	z	6.040	3.980	2.060	∞	
Path 2	∞	2	2	3	rst	x_cs_reg/CE	3.569	1.603	1.966	∞	input port clock
Path 3	∞	2	2	3	rst	z_reg_reg/CE	3.569	1.603	1.966	∞	input port clock
Path 4	∞	2	1	3	rst	q1_reg/D	3.414	1.631	1.783	∞	input port clock
Path 5	∞	2	1	3	rst	q0_reg/D	3.386	1.603	1.783	∞	input port clock
Path 6	∞	2	1	4	x	z_reg_reg/D	2.697	1.615	1.082	∞	input port clock
Path 7	∞	1	1	4	x	x_cs_reg/D	2.270	1.491	0.779	∞	input port clock

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 8	∞	2	1	3	x_cs_reg/C	z_reg_reg/D	0.296	0.227	0.069	-∞	
Path 9	∞	2	1	2	q0_reg/C	q1_reg/D	0.356	0.183	0.173	-∞	
Path 10	∞	2	1	2	q0_reg/C	q0_reg/D	0.359	0.186	0.173	-∞	
Path 11	∞	1	1	4	x	x_cs_reg/D	0.578	0.259	0.319	-∞	input port clock
Path 12	∞	2	2	3	rst	x_cs_reg/CE	1.041	0.292	0.749	-∞	input port clock
Path 13	∞	2	2	3	rst	z_reg_reg/CE	1.041	0.292	0.749	-∞	input port clock
Path 14	∞	2	1	1	z_reg_reg/C	z	1.851	1.366	0.485	-∞	

Behavioral Model

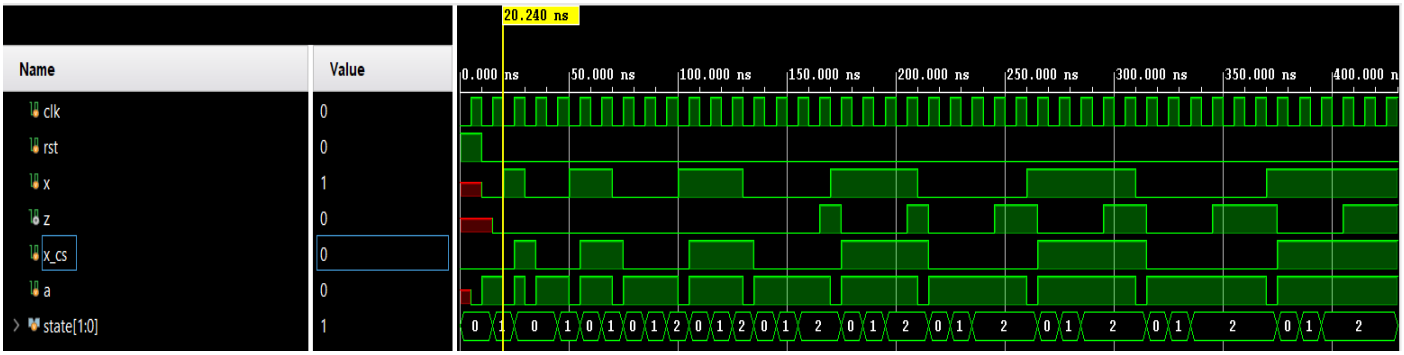
Design Source

```
module div_behav
(
    input clk,
    input rst,
    input x,
    output z
);
    reg z_reg,a,x_cs;
    localparam A = 2'b00;
    localparam B = 2'b01;
    localparam C = 2'b10;
    reg [1:0] state;

    always@(*) begin
        a<= ~(x_cs ^ x);
    end

    always @(posedge clk, posedge rst) begin
        if(rst == 1) begin
            state <= A;
            a<=0;
            x_cs<=0;
        end
        else begin
            x_cs<=x;
            case(state)
                A : begin
                    if(a==1) begin
                        state <= B;
                    end
                    else begin
                        state <= A;
                    end
                    z_reg<= 0;
                end
                B: begin
                    if(a==1) begin
                        state<= C;
                    end
                    else begin
                        state<= A;
                    end
                    z_reg<=0;
                end
                C: begin
                    if(a==1) begin
                        state<= C;
                        z_reg<= 1;
                    end
                    else begin
                        state<= A;
                        z_reg<=0;
                    end
                end
                default: begin
                    state <= A;
                    z_reg<= 0;
                end
            endcase
        end
    end
    assign z = z_reg;
endmodule
```

Simulation Source

```
module div_behav_tb();

    reg clk = 1'b0;
    reg rst = 1'b1;
    reg x;
    wire z;
    reg [41:0] test = 42'b010011000111000011110000011111000000111111;
    integer i= 41;
    div_behav uut
    (
        .clk(clk),
        .rst(rst),
        .x(x),
        .z(z)
    );

    always begin
        #5 clk = ~clk;
    end

    initial begin
        #10
        rst = 1'b0;
        for(i = 41; i>=0; i = i-1) begin
            x = test[i];
            #10;
        end
        $finish;
    end
endmodule
```

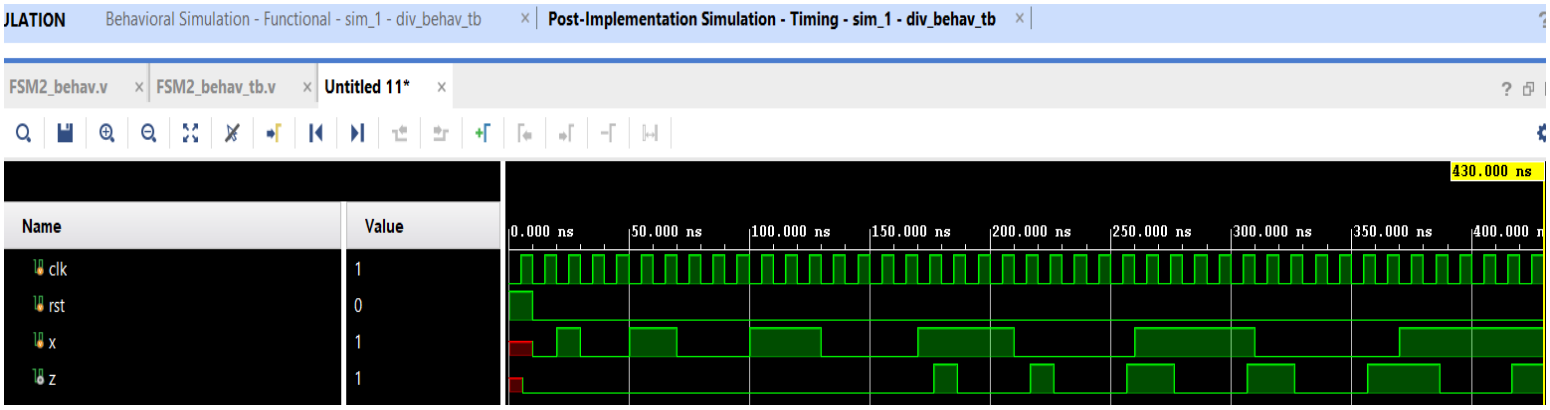
Simulation Waveform



Behavioral model gives correct results. z output obtains logic 1 value after four consecutive 1s or 0s.

Salih Ömer Ongün
040220780

Post -Implementation Timing Simulation



Utilization Report

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	3	32600	0.01
FF	4	65200	0.01
IO	4	210	1.90
BUFG	1	32	3.13

Timing Report

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	2	1	1	z_reg_reg/C	z	6.040	3.980	2.060	∞	
Path 2	∞	2	2	3	rst	x_cs_reg/CE	3.569	1.603	1.966	∞	input port clock
Path 3	∞	2	2	3	rst	z_reg_reg/CE	3.569	1.603	1.966	∞	input port clock
Path 4	∞	1	1	3	rst	FSM_sequential_state_reg[0]/CLR	2.980	1.479	1.501	∞	input port clock
Path 5	∞	1	1	3	rst	FSM_sequential_state_reg[1]/CLR	2.980	1.479	1.501	∞	input port clock
Path 6	∞	2	1	4	x	z_reg_reg/D	2.697	1.615	1.082	∞	input port clock
Path 7	∞	2	1	4	x	FSM_sequential_state_reg[0]/D	2.480	1.615	0.865	∞	input port clock
Path 8	∞	2	1	4	x	FSM_sequential_state_reg[1]/D	2.474	1.609	0.865	∞	input port clock
Path 9	∞	1	1	4	x	x_cs_reg/D	2.270	1.491	0.779	∞	input port clock

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 10	∞	2	1	3	FSM_sequential_state_reg[0]/C	z_reg_reg/D	0.283	0.186	0.097	-∞
Path 11	∞	2	1	3	FSM_sequential_state_reg[0]/C	FSM_sequential_state_reg[1]/D	0.367	0.183	0.184	-∞
Path 12	∞	2	1	3	FSM_sequential_state_reg[0]/C	FSM_sequential_state_reg[0]/D	0.370	0.186	0.184	-∞
Path 13	∞	1	1	4	x	x_cs_reg/D	0.578	0.259	0.319	-∞
Path 14	∞	1	1	3	rst	FSM_sequential_state_reg[0]/CLR	0.844	0.247	0.597	-∞
Path 15	∞	1	1	3	rst	FSM_sequential_state_reg[1]/CLR	0.844	0.247	0.597	-∞
Path 16	∞	2	2	3	rst	x_cs_reg/CE	1.041	0.292	0.749	-∞
Path 17	∞	2	2	3	rst	z_reg_reg/CE	1.041	0.292	0.749	-∞
Path 18	∞	2	1	1	z_reg_reg/C	z	1.851	1.366	0.485	-∞

Behavioral model and my previous model have the same resource usage. Behavioral model and my previous model have same delays.

REFERENCES

[1] Arar, S. (2018, March 5). *Encoding the states of a finite state machine in VHDL*. Technical Articles. https://www.allaboutcircuits.com/technical-articles/encoding-the-states-of-a-finite-state-machine-vhdl/