DIGITAL SYSTEM DESIGN APPLICATION PROJECT 6

STATE REDUCTION

a) We prepare table including state,next state input and output. İf there are states that the same as next state and output, we can reduction these.and we can delete one of the similar and write not deleted state name to deleted states.

b)

State	next state, output	
	x = 0	x = 1
A	B,0	D,0
B	C,0	D,0
C	C,1	D,0
D	A,0	E,0
E	A,0	F,0
F	A,0	F,1

İn this table we can not do reduction. There are not state that have same output and same next state.

STATE ENCODING

There are several methods of encoding state. We decide to which one. gains are effective like Speed, area power consumption. For example type of encoding binary encoding, one – hot encoding, gray encoding . A wrong decision may result in a state machine that uses too much logic, too slow, or both.

State Variables

State	One-Hot Code	Binary Code	Gray Code	
S0	00001	000	000	
S1	00010	001	001	
S2	00100	010	011	
S3	01000	011	010	
S4	10000	100	110	

Table 1:An example of state Encoding for a 4 state Machine

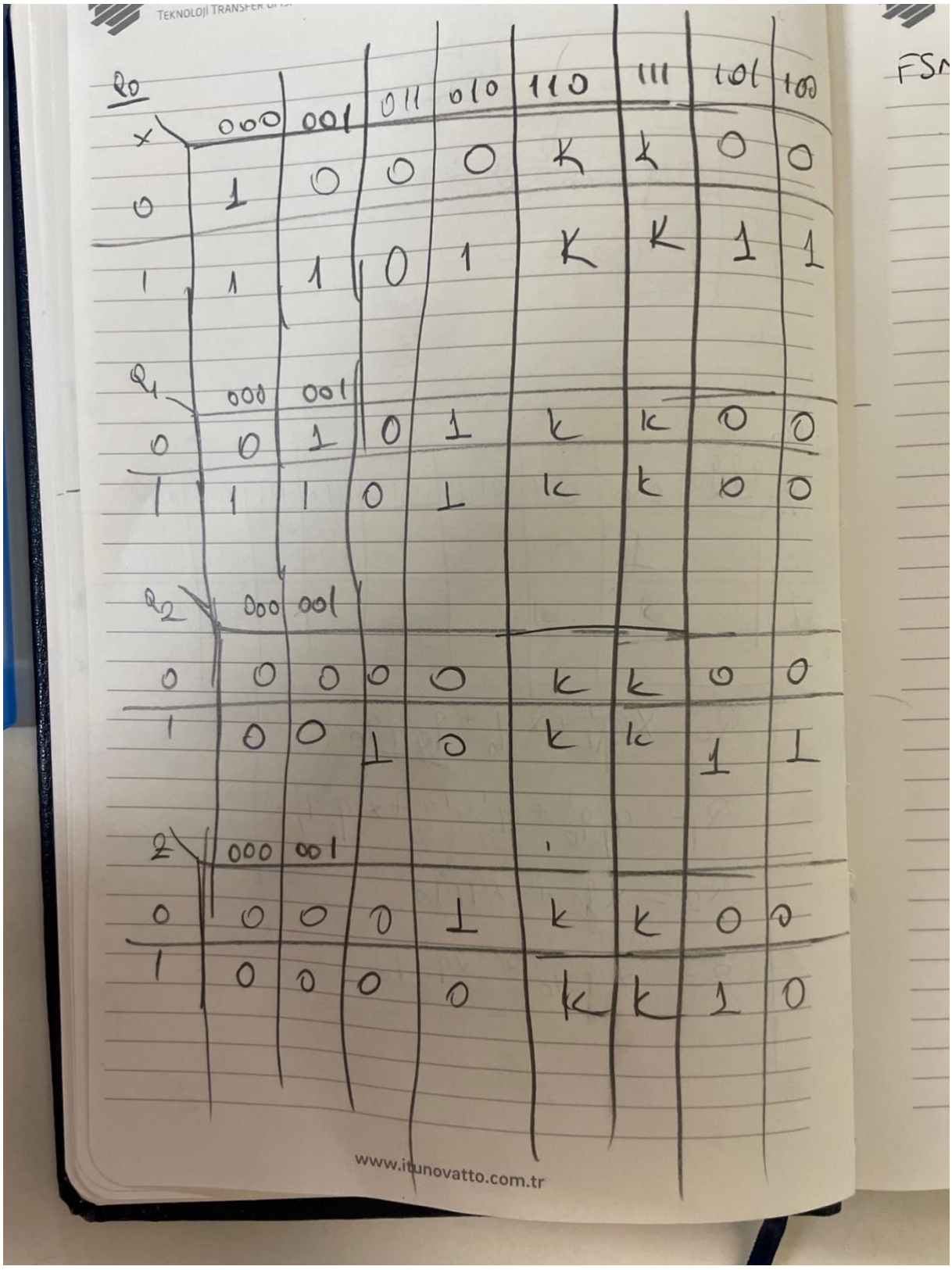
b) binary encoding

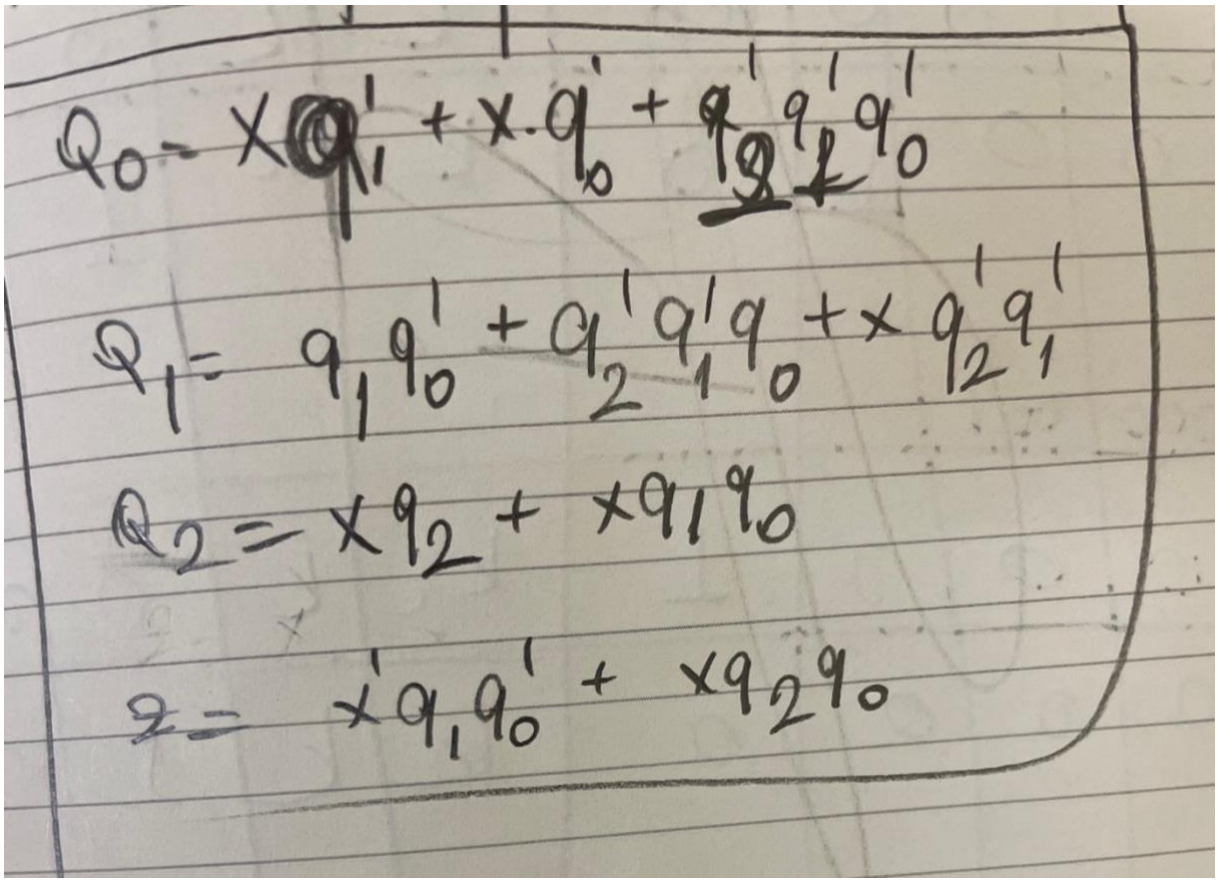
STATE	NS,X=0	NS, X=1	OUT , X=0	OUT, X=1	BİNARY
A	B	D	0	0	000
B	C	D	0	0	001
C	C	D	1	0	010
D	A	E	0	0	011
E	A	F	0	0	100
F	A	F	0	1	101

Binary encoding. Less flip flop than one-hot. This cause using more power.
We give sequential binary code to squential state name. And changing to table like bottom. Q=log2(n) ,n=6 cosest number q=3
Gates: gray > binary> one-hot
Flipflop: one -hot>binary=gray
Power consumption gray<binary?one-hot

x	q2	q1	q0	Q2	Q1	Q0	z
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	k	k	k	k
0	1	1	1	k	k	k	k
1	0	0	0	0	1	1	0
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	k	k	k	k
1	1	1	1	k	k	k	k

REDUCTION OF COMBINATORIAL PART





B) lut 4)

For Less lut use One – hot encoding .

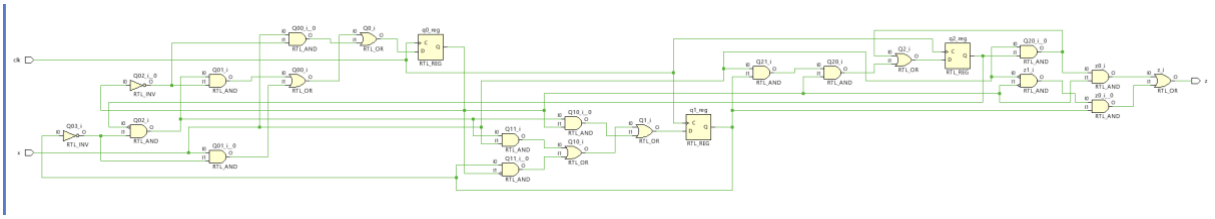
VERILOG ENCODING

```
`timescale 1ns / 1ps
module FSM1(
    input x,clk,
    output z
);
reg q0=1'b0,q1=1'b0,q2=1'b0;
wire Q0,Q1,Q2;
always@(posedge clk)
begin
    q0 <= Q0;
    q1 <= Q1;
    q2 <= Q2;
end

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

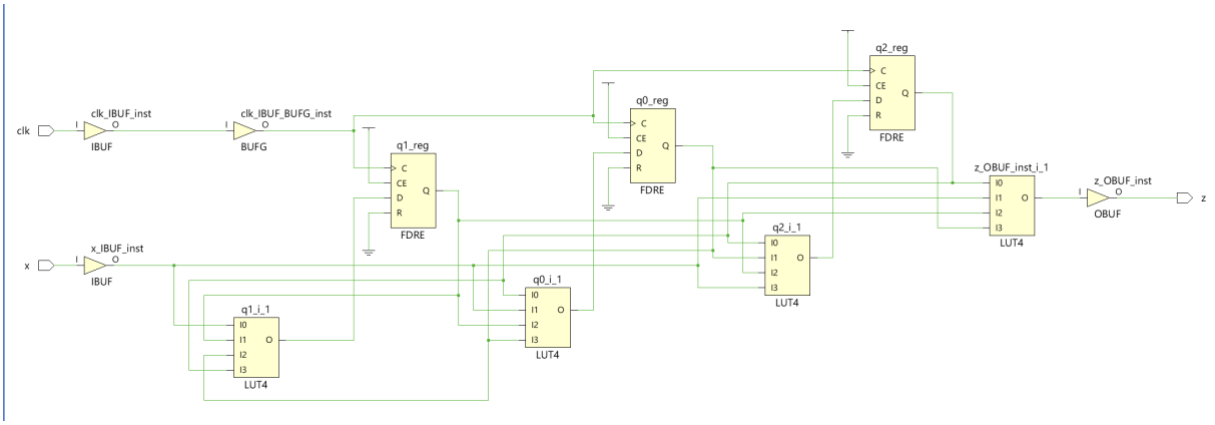
testbench

```
`timescale 1ns / 1ps
module experiment6tb;

  reg clk;
  wire z;
  reg x;
  reg [31:0]values;
  reg [9:0]values2;

  FSM1 uut(.x(x),.clk(clk),.z(z));
  integer i;
  initial
  begin
     clk=1'b0;
     values = 32'b01001100011100001111000001111100;
     values2 = 10'b0000111111;
     i=31;
     while(i>=0)
     begin
        clk = ~clk;
        x=values[i];
        #5
        clk <=~clk;
        #5
        i = i-1;
     end
        clk = ~clk;
        x=values2[9];   #5
        clk <=~clk;  #5
        clk = ~clk;
        x=values2[8];
        #5 clk <=~clk;
        #5     clk = ~clk;  x=values2[7];
        #5clk <=~clk;
        #5     clk = ~clk;x=values2[6];
        #5clk <=~clk;
        #5     clk = ~clk;
        x=values2[5];#5clk <=~clk;
        #5     clk = ~clk;x=values2[4];
        #5clk <=~clk;#5
        clk = ~clk;x=values2[3];
        #5clk <=~clk;
```

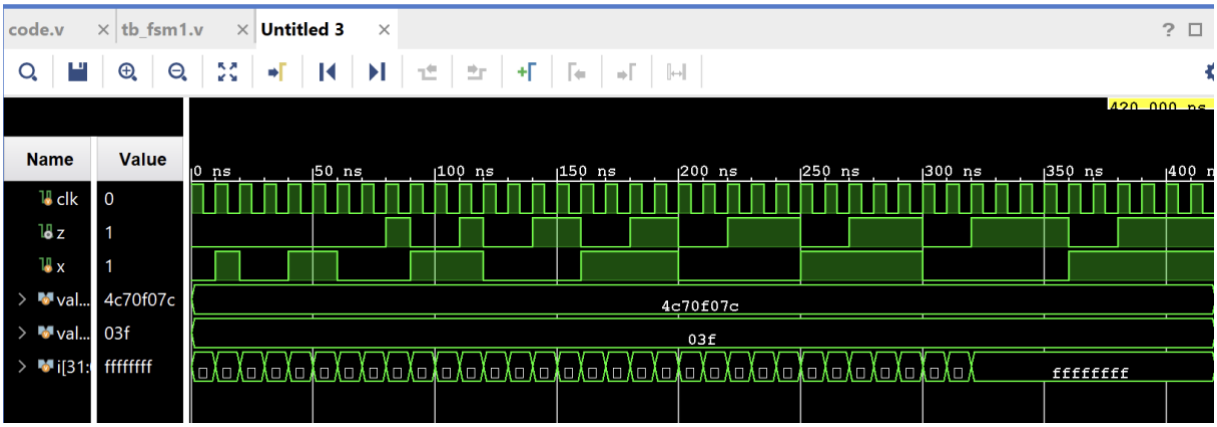
rtl



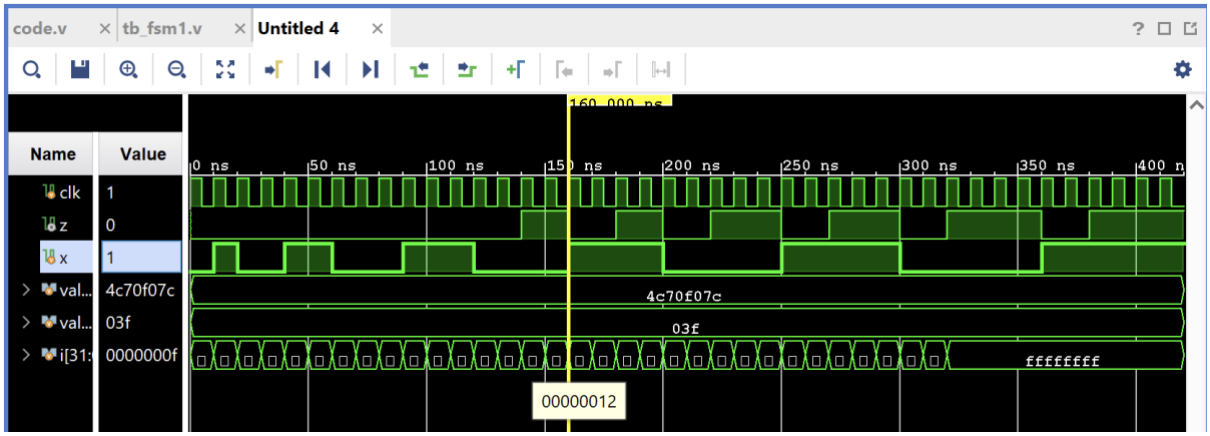
Technology schematic



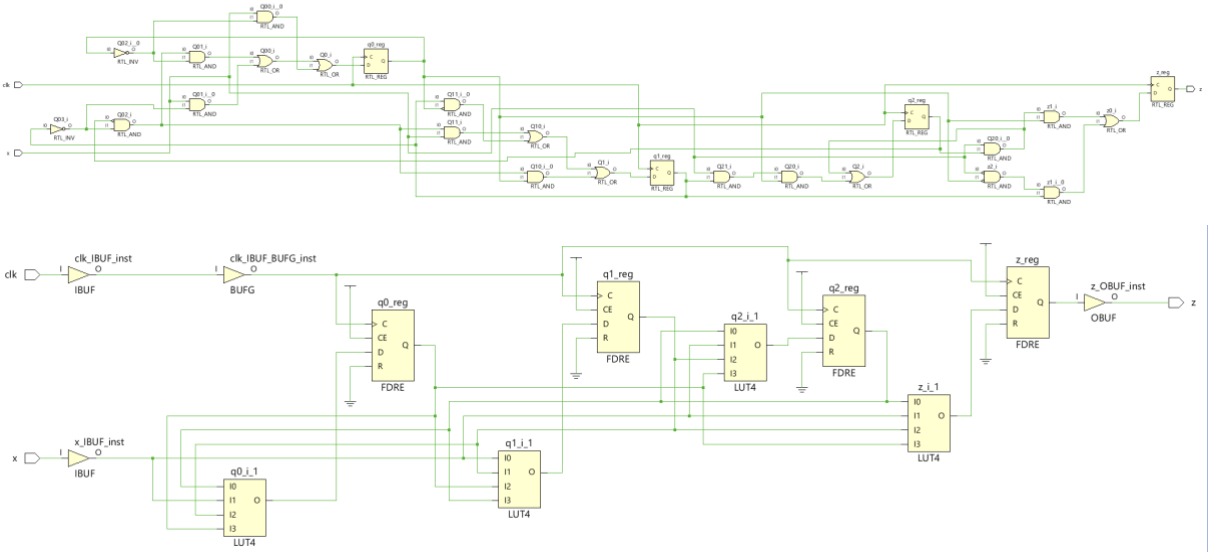
SIMULATION OF THE CIRCUIT

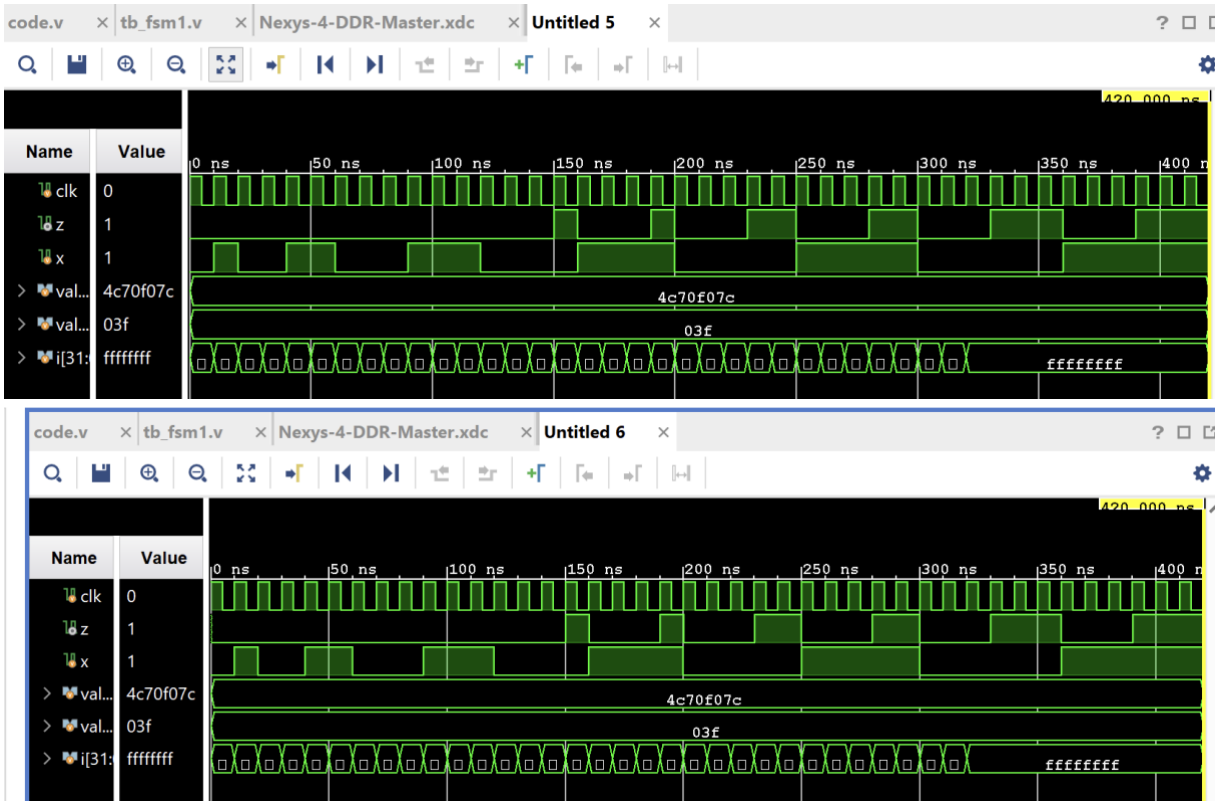


ANALYSIS OF FAULITY OUTPUTS

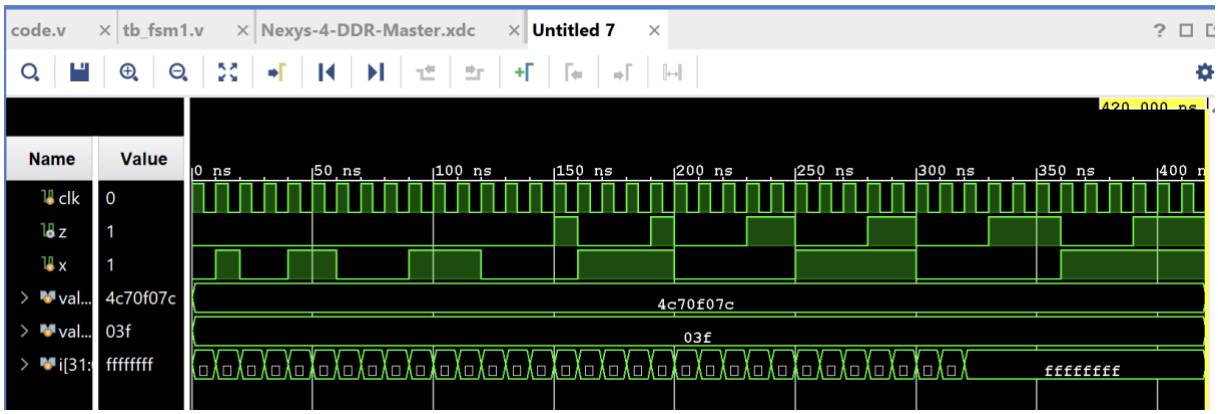


MACHINE TYPE CHANGING





STUCKING IN UNDESIRABLE STATES



Yes, it is working after undifined state

DESIGN WITH DIVIDED STATE DIAGRAMS