Digital System Design Applications

PROJECT - 2

SEQUENTIAL CIRCUITS

EHB436E
CRN: 10345

Group:1

Nurgül Gürler
040190251

Hatice Nur Andı
040200203

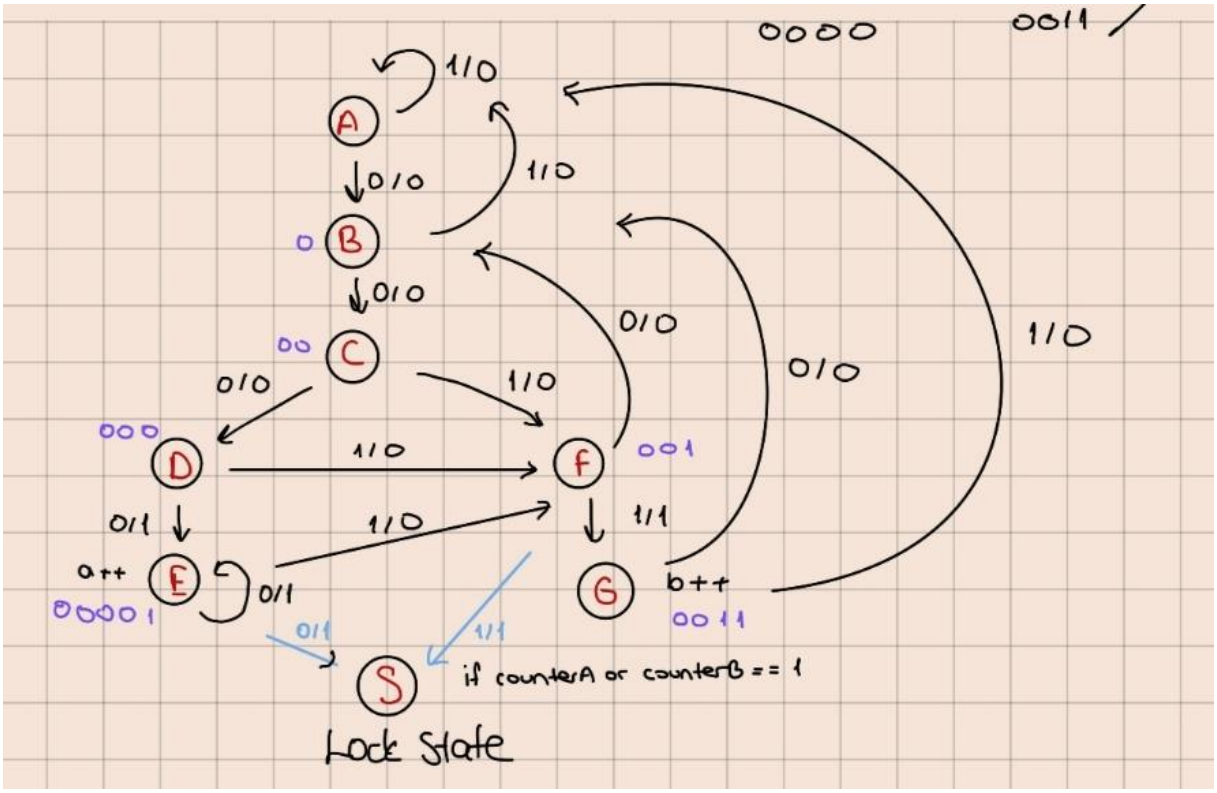
Nurgül Gürler 040190251
Hatice Nur Andı 040200203

Design Steps

1. State Diagram

The circuit purpose is detection of the two 4 bit numbers seperately. In our case, the chosen school number is 040200203. So, A=0000 and B=0011.

Here is the state diagram of our design;



In our design, the circuit detects both A and B and when one of these occurs, two counter is counting it. So, when second one occurs for A and B seperately, the circuits goes to another state that indicated as S which is denotes the lock state.

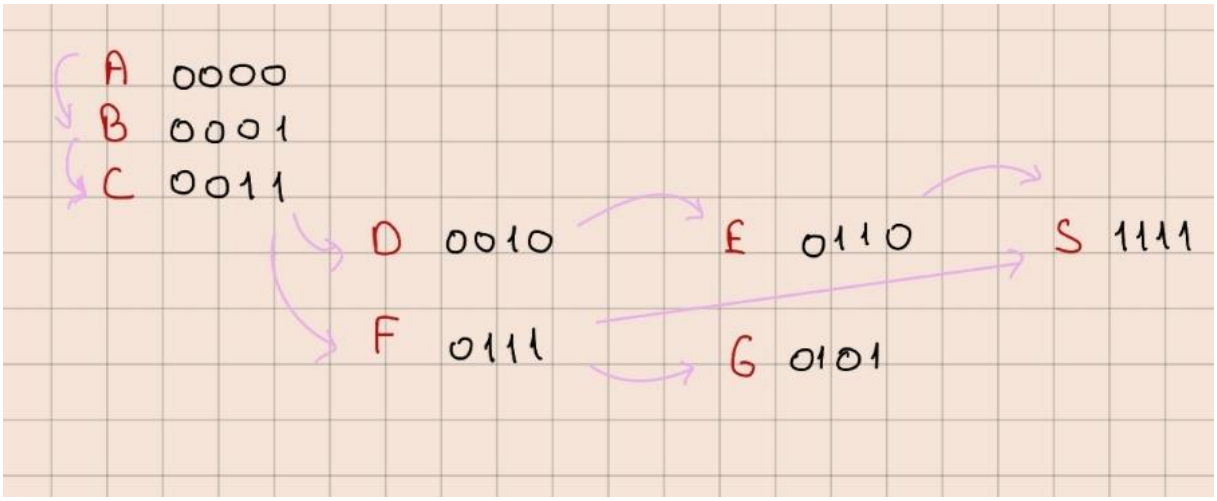
Here is the code for state machine;

```
`timescale 1ns / 1ps

module state_machine(
input X,
input CLK,
input RST,
output Y
);
```

```
reg [3:0] state;
reg [1:0] counterA=2'b00, counterB=2'b00;
reg flaga=0, flagb=0;
parameter

A= 4'b0000,
B= 4'b0001,
C= 4'b0011,
D= 4'b0010,
E= 4'b0110,
F= 4'b0111,
G= 4'b0101,
S= 4'b1111;

always @(posedge RST or posedge CLK)

begin

if  (RST) begin
    state<=A;
    counterA<=0;
    counterB<=0;
    end

else begin

case (state)

    A: if (X) state <=A;
    else state <=B;


    B: if (X) state <=A;
    else state <=C;


    C: if (X) state <=F;
    else state <=D;

    D: if (X) state <=F;
    else begin
    state <=E;
    end

     E: begin
       if (counterA < 2'b10) counterA <= counterA+1;
        if (counterA==2'b01) begin
            if (X==0)begin
```

```
        state <=S;
        end
       end
        else begin
         state <=F;
        end

     end

  F: begin
        if (counterB==2'b01) begin
            if (X==1) state<=S;
            else state<=B;
        end

        else if(X) state <=G;
        else begin state <=B;
       end
        end

     G: begin
     if (counterB < 2'b10) counterB<= counterB+1;
     if (X) state <=A;
    else begin
    state <=B;
    end
    end

      S: state <=S;

     default state <= A ;

 endcase

    if (counterB==2'b10 || counterA==2'b10) begin
        state<=S;
    end
 end
 end

 assign Y = (state==S) | (state==E) | (state==G);

endmodule
```

2. Encoding the states



To implement the circuit with less power consumption, we coded the states systematically as changing only one bit when state changes according to the relation of the states.

3. FSM Reduction Techniques

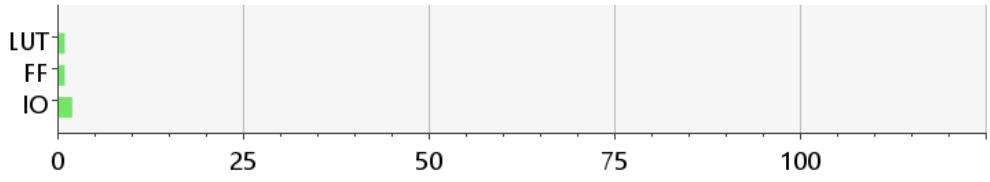
State Elimination, Equivalent State Identification, State Encoding, Dead State Elimination, State Assignment, Partitioning, Sequential Equivalence Checking, Optimizing Transitions, and Technology Mapping are reduction techniques used in finite state machines.

One of the FSM reduction technique is the equivalent state reduction. For this technique, the equivalent states in terms of state transition, output and input are elimineted to one state. In our state machine, we consider this method from start so, our machine has not got any states that needs to be reducted.

4. Post Implementation Reports

From the post-implementation reports, here is the some report results;

Resource	Utilization	Available	Utilization %
LUT	10	63400	0.02
FF	7	126800	0.01
IO	4	210	1.90



The total LUT count is 10, FF count is 7 and lastly, total IO port number is 4 according to the utilization report.

From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
FSM_sequential_state_reg[0]/C	Y	6.902	4.100	2.802	∞	
X	FSM_sequential_state_reg[1]/D	4.101	1.733	2.368	∞	input port clock
X	FSM_sequential_state_reg[2]/D	3.961	1.733	2.228	∞	input port clock
X	FSM_sequential_state_reg[0]/D	3.861	1.733	2.127	∞	input port clock
RST	FSM_sequential_state_reg[2]/CLR	2.538	1.478	1.060	∞	input port clock
RST	counterA_reg[0]/CLR	2.538	1.478	1.060	∞	input port clock
RST	counterA_reg[1]/CLR	2.538	1.478	1.060	∞	input port clock
RST	FSM_sequential_state_reg[0]/CLR	2.533	1.478	1.056	∞	input port clock
RST	counterB_reg[0]/CLR	2.533	1.478	1.056	∞	input port clock
RST	counterB_reg[1]/CLR	2.533	1.478	1.056	∞	input port clock

On the view of timing, the maximum path delay is 6.902 ns. So, the maximum frequency of the circuit is 144.8 MHz.

5. Constraints

“set_max_delay -from [get_cells -hierarchical *] -through [get_nets -hierarchical *] -to [get_cells -hierarchical *] 1.000”

The design could not achieve the goal.

From	To	Total Delay	Logic Delay	Net Delay	Requirement
FSM_sequential_state_reg[1]/C	FSM_sequential_state_reg[0]/D	2.488	0.704	1.784	3.0
FSM_sequential_state_reg[1]/C	FSM_sequential_state_reg[1]/D	2.404	0.704	1.700	3.0
counterA_reg[1]/C	FSM_sequential_state_reg[2]/D	2.282	0.839	1.443	3.0
FSM_sequential_state_reg[0]/C	counterA_reg[0]/D	1.482	0.580	0.902	3.0
FSM_sequential_state_reg[0]/C	counterA_reg[1]/D	1.508	0.606	0.902	3.0
counterB_reg[1]/C	counterB_reg[0]/D	1.415	0.715	0.700	3.0
counterB_reg[1]/C	counterB_reg[1]/D	1.443	0.743	0.700	3.0

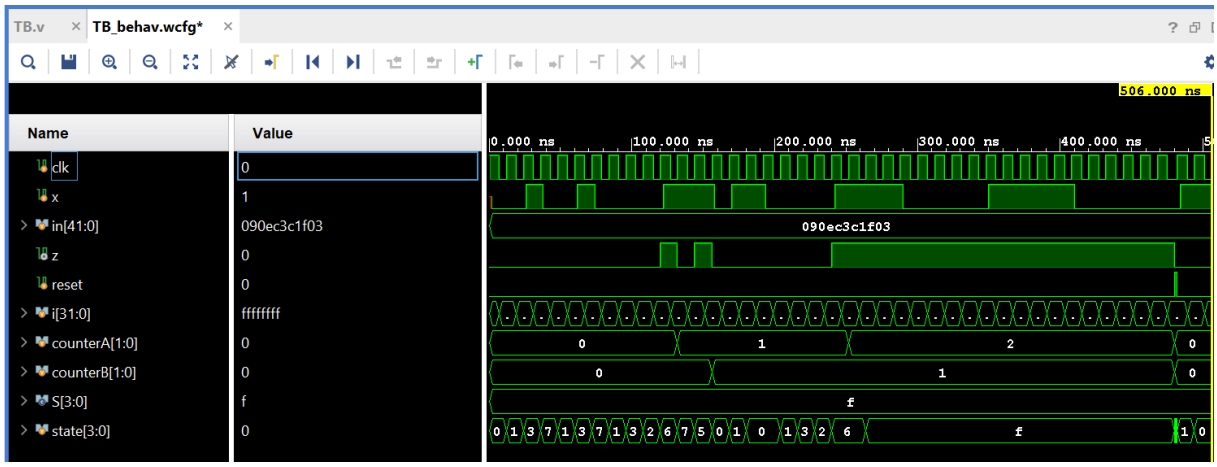
6. Waveforms

Here is the testbench code;

```
`timescale 1ns / 1ps

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

```
  reg reset;

  state_machine uut(.X(x), .CLK(clk), .Y(z), .RST(reset));

  integer i;
  initial begin
  reset=1'b0;
  clk=1'b0;
  in=42'b001001000011011000011111000001111100000011;
  i=41;
  while(i>=0)
  begin

  if (i==1) begin
    reset<=1;
    #2
    reset<=0;
    end


  clk<=~clk;
  #2
  x=in[i];
  #5
  clk<=~clk;
  i=i-1;
  #5;

  end
  $finish;

  end
endmodule
```

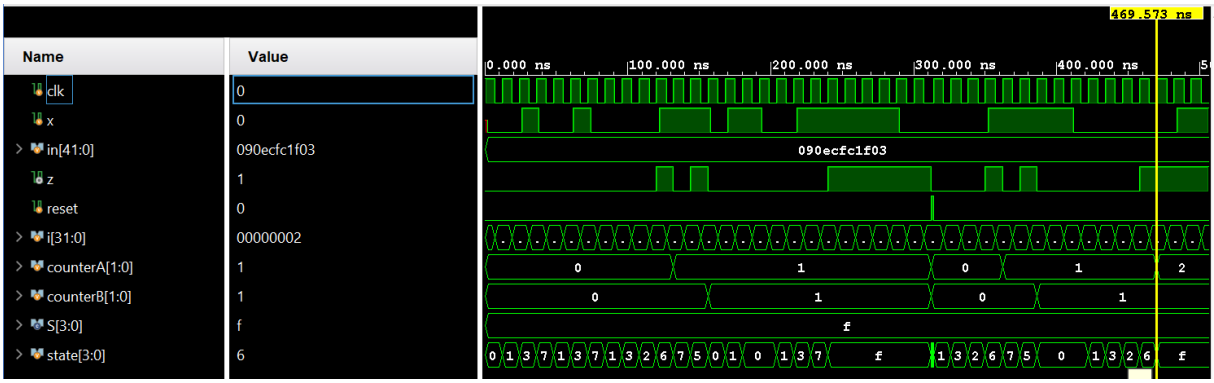
For this code, the related waveform is attached below. We can see the detection of first A and B and the second A with the lock state. At the end, we can see the RST signal and escaping from the lock state.



Here is another example;

```
integer i;
initial begin
reset=1'b0;
clk=1'b0;
in=42'b001001000011101100111111100000111111100000011;
i=41;
while(i>=0)
begin
```

In this input X, we are testing the circuit abilities that are detecting A and B when overlap situation, Lock state when the second B comes. Also we added a reset signal after lock state to see escaping from lock state.



In this waveform, we can see that after A and B detected once, when the second B came, the circuit can go to the lock state until RST signal come. After reset occur, the circuit can escape the lock state.

7. Circuit Type

Our circuit is a Moore Type circuit, the output is only depends on the state changes, not also inputs. There are no fauty outputs also.