ELEC 204 DIGITAL SYSTEM AND DESIGN FINAL PROJECT REPORT

Student name: **Osman Raşit KÜLTÜR**

ID Number: **37250**

Instructor name: **Alper Tunga Erdoğan**

Project name: **Asyncronous Combination Lock**

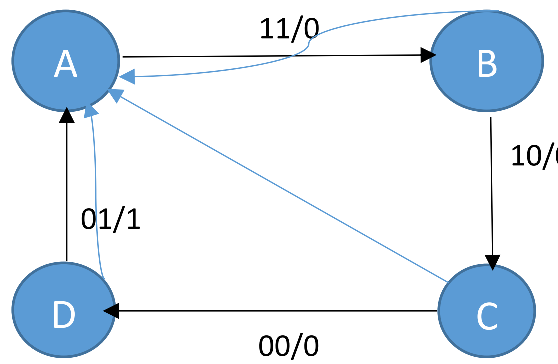
Date: **12/05/2016**

INTRODUCTION

This combinational lock has a minimum sequence of four two-bit input symbols as the combination and appears to the user as if it is an asynchronous circuit. Actually, it is a synchronous circuit with a fast clock and synchronization of the user inputs. For a given input combination, the circuit goes to a state and cycles there until the input changes to a new symbol; thus, the combination cannot contain consecutive appearances of the same symbol. The lock is locked using an asynchronous RESET.

The project includes a D Flip Flop and Mealy State Diagram. The lock unlocks itself after entering four two-bit number in correct sequence. In this project we have assumed that it is 11100001. When each two-bit number specified correctly, corresponding LEDs will be turned on and if the user enter the 8-bit sequence correctly, all 8 LEDs will also be turned on. If the user makes a mistake in any state, the password has to be entered from the beginning. To unlock the circuit, the transitions between the states (given in the below) must be correct (from A to D means first to last). The D-Type Flip Flops save the inputs simultaneously with the manual clock to remember the old inputs. In this project, a manual clock is preferred instead of FPGA's clock.

STATE DIAGRAM



TRUTH TABLE

PRESENT STATE	INPUT	NEXT STATE
00	00	00
	01	10
	10	00
	11	00
01	00	00
	01	00
	10	11
	11	00
10	00	01
	01	00
	10	11
	11	00
11	00	01
	01	00
	10	00
	11	00

VHDL CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- My password is 11100001  in this case.
entity deneme is
   Port ( Clock : in  STD_LOGIC;
                     Reset : in  STD_LOGIC;
                     x1 : in  STD_LOGIC; -- MSB
                     x0 : in  STD_LOGIC; -- LSB
                     led1 : out  STD_LOGIC;
                     led2 : out  STD_LOGIC;
                     led3 : out  STD_LOGIC;
                     led4 : out  STD_LOGIC;
                     led5 : out  STD_LOGIC;
                     led6 : out  STD_LOGIC;
                     led7 : out  STD_LOGIC;
                     led8 : out  STD_LOGIC);
end deneme;
architecture Behavioral of deneme is
-- Intermediate Signals
type state_type is (A,B,C,D);
signal state : state_type;
```

```
Begin

-- If user enters wrong digits, the machine goes back to initial state. (goes state A)

process(Clock,Reset)

begin

if Reset='1' then state <=A;

elsif Clock'event and Clock='1' then

case state is


when A => if x1='1' and x0='1' then

state <= B;

else

state <= A;

end if;


when B => if x1='1' and x0='0' then

state <= C;

else

state <= A;

end if;


when C => if x1='0' and x0='0' then

state <= D;

else

state <= A;

end if;
```

```
when D => if x1='0' and x0='1' then
state <= A;
end if;
end case;
end if;
end process;

process(state,x1,x0)
begin
case state is
-- If first two digits are entered correctly, then first led will be on. Otherwise all leds are off.
when A => if x1='1' and x0='1' then
led1 <= '1';
led2 <= '0';
led3 <= '0';
led4 <= '0';
led5 <= '0';
led6 <= '0';
led7 <= '0';
led8 <= '0';
else
led1 <= '0';
led2 <= '0';
led3 <= '0';
led4 <= '0';
led5 <= '0';
led6 <= '0';
```

led7 <= '0';

led8 <= '0';

end if;

-- If second two digits are entered correctly, then first and second led will be on. Otherwise all leds are off.

when B => if x1='1' and x0='0' then

led1 <= '1';

led2 <= '1';

led3 <= '0';

led4 <= '0';

led5 <= '0';

led6 <= '0';

led7 <= '0';

led8 <= '0';

else

led1 <= '0';

led2 <= '0';

led3 <= '0';

led4 <= '0';

led5 <= '0';

led6 <= '0';

led7 <= '0';

led8 <= '0';

end if;

-- If third two digits are entered correctly, then first, second and third leds will be on. Otherwise all leds are off.

```
when C => if x1='0' and x0='0' then

led1 <= '1';
led2 <= '1';
led3 <= '1';
led4 <= '0';
led5 <= '0';
led6 <= '0';
led7 <= '0';
led8 <= '0';
else
led1 <= '0';
led2 <= '0';
led3 <= '0';
led4 <= '0';
led5 <= '0';
led6 <= '0';
led7 <= '0';
led8 <= '0';
end if;
-- If last two digits are entered correctly, then all eight leds will be on. Otherwise all leds are off.
when D => if x1='0' and x0='1' then

led1 <= '1';
led2 <= '1';
led3 <= '1';
led4 <= '1';
```

led5 <= '1';

led6 <= '1';

led7 <= '1';

led8 <= '1';

else

led1 <= '0';

led2 <= '0';

led3 <= '0';

led4 <= '0';

led5 <= '0';

led6 <= '0';

led7 <= '0';

led8 <= '0';

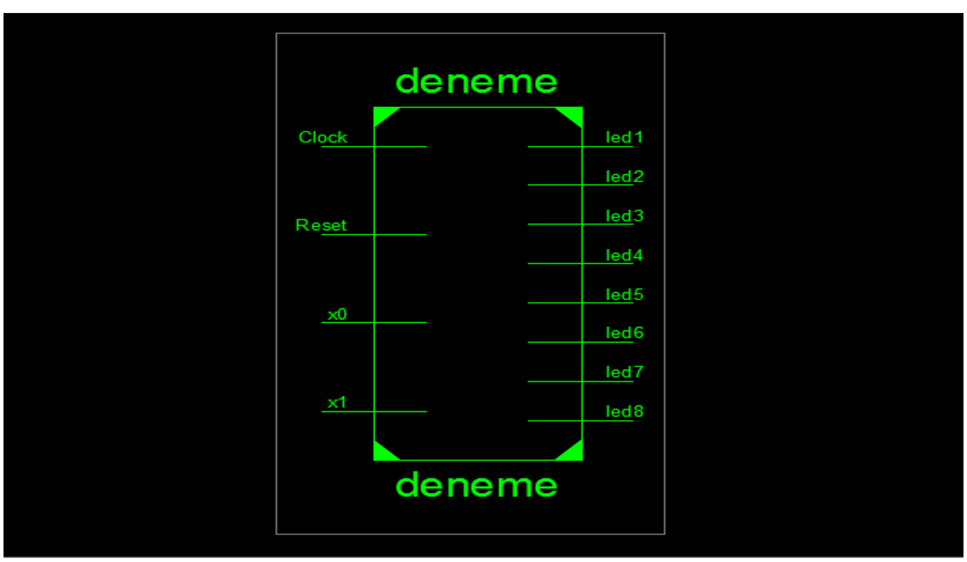
end if;

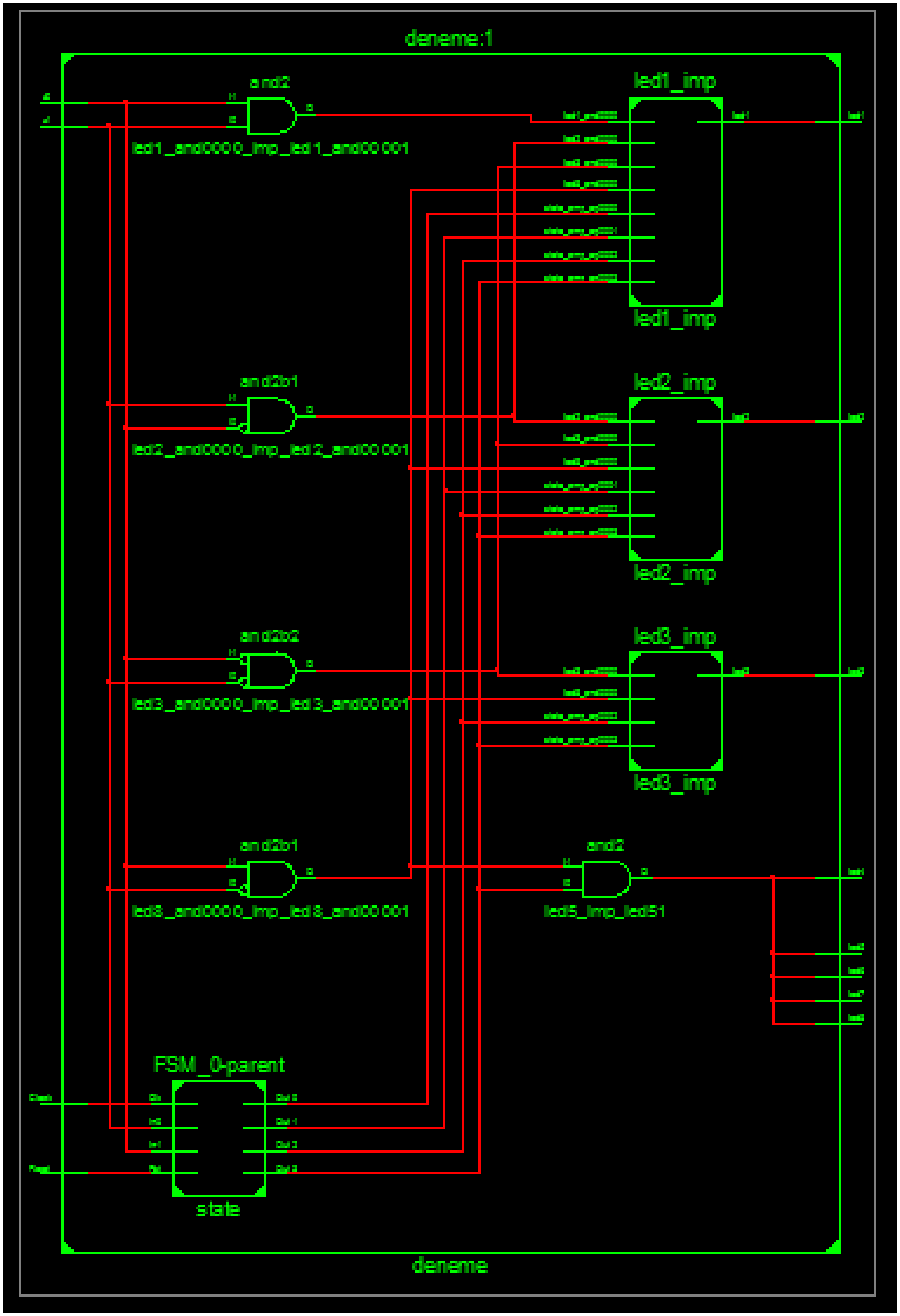
end case;

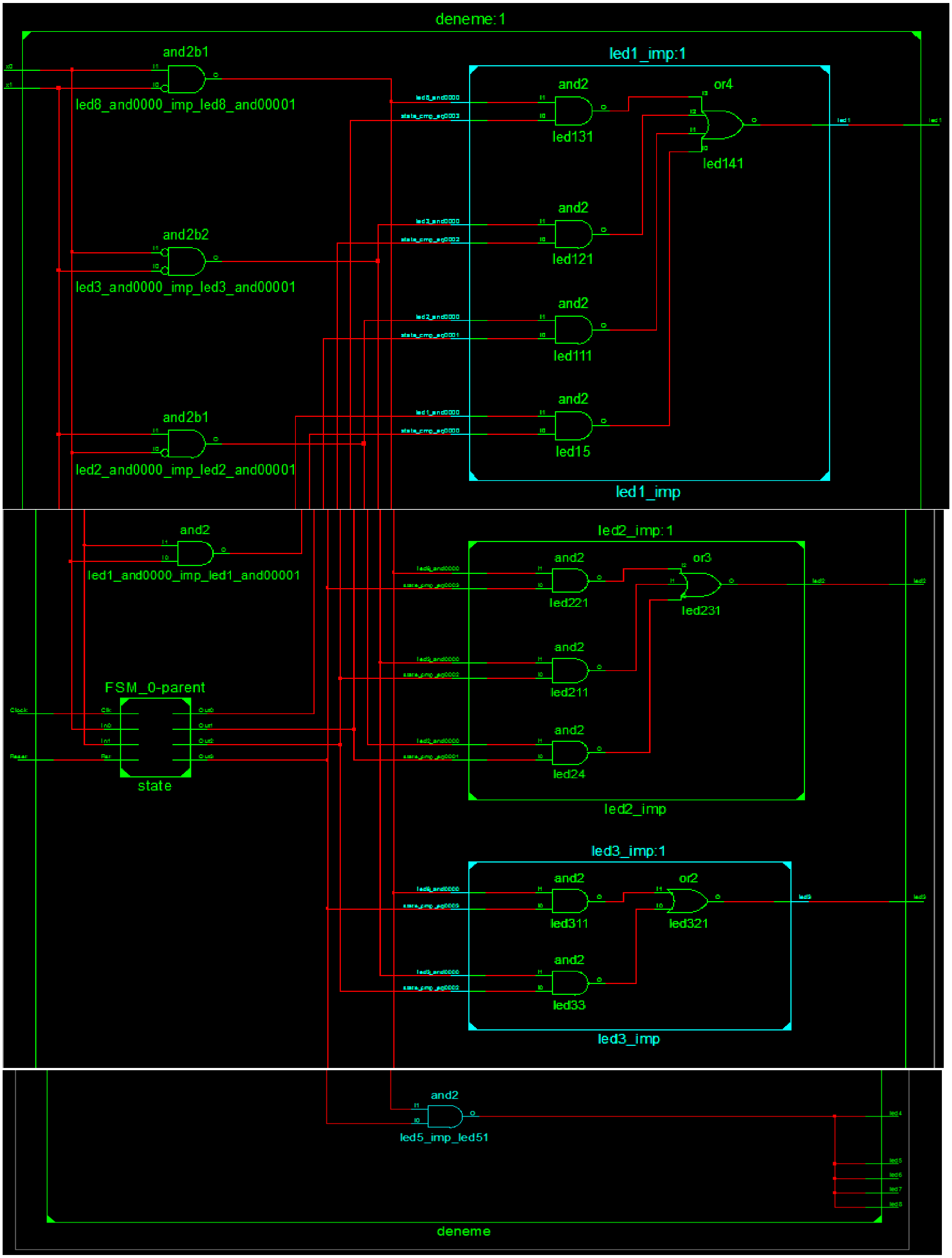
end process;

end Behavioral;

RTL SCHEMATICS




deneme:1
and2
led1_and0000_imp_led1_and00001
led1_imp
and2b1
led2_and0000_imp_led2_and00001
led2_imp
and2b2
led3_and0000_imp_led3_and00001
led3_imp
and2b1
led8_and0000_imp_led8_and00001
and2
led5_imp_led51
FSM_0-parent
state
deneme



deneme:1
and2b1
led8_and0000_imp_led8_and00001
led1_imp:1
and2
or4
led131
led141
and2
led121
and2b2
led3_and0000_imp_led3_and00001
and2
led111
and2b1
and2
led15
led2_and0000_imp_led2_and00001
led1_imp
and2
led1_and0000_imp_led1_and00001
led2_imp:1
and2
or3
led221
led231
and2
led211
FSM_0-parent
and2
led24
state
led2_imp
led3_imp:1
and2
or2
led311
led321
and2
led33
led3_imp
and2
led5_imp_led51
deneme

USER CONSTRAINTS

I/O Ports

Name	Direction	Neg Diff Pair	Site
Clock	Input		G12
led1	Output		M5
led2	Output		M11
led3	Output		P7
led4	Output		P6
led5	Output		N5
led6	Output		N4
led7	Output		P4
led8	Output		G1
Reset	Input		K3
x0	Input		P11
x1	Input		L3

Tcl Console | Package Pins | I/O Ports

CONCLUSION

In this project, by using D-flip flops and state machines, I designed asynchronous combination lock. I used the development environment ISE 14.7 in my PC and ISE 10.1 in the lab. Rather than FPGA's itself clock, I have preferred to use manual clock to avoid any kind of frequency and time conflict. Syntax errors were pointed out by the ISE and I fixed them before the project presentation. As a result, we successfully unlocked the lock by entering 11100001 in a sequence. It is easy to change the password by changing a bit of code. By making this project and copleting lab experiments, I'm convinced that I gained enough VHDL experience to do better jobs in the industry.