University of Birmingham

Advanced Digital Design

Simple Combinational Lock

Folorunsho Solomon Opeyemi 1148183

12

Contents

INTRODUCTION 3
DESIGN ORGANISATION 4
MainModule: 4
InternalClock: 5
Display: 5
DisplayMultiplexer: 5
MessageMultiplexer: 6
RandomNumber: 6
SIMULATION RESULTS 7
SYNTHESIS RESULTS 12
CONCLUSIONS 13
What I have learnt 13
Possible Improvements 13
APPENDIX 14
InternalClock.vhd 14
RandomNumber.vhd 15
Display.vhd 16
DisplayMultiplexer.vhd 17
MessageMultiplexer.vhd 19
MainModule.vhd 20
Mainodule_tb.vhd 26

INTRODUCTION

This document describes the design of a simple combinational lock using VHDL, by implementing a synchronous finite state machine.

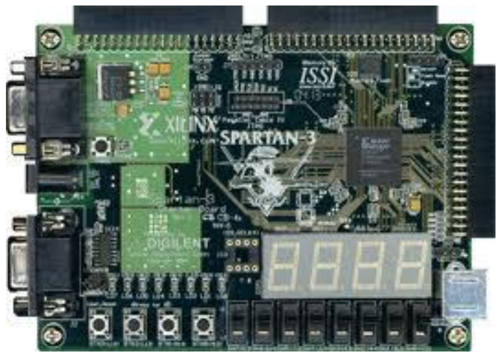

Figure 1

This was prototyped using a nexus 3 Spartan1000 prototyping ,it consists of four seven segment display ,eight slider switches ,four push buttons ,a JTAG programming connector and many more features.

The lock has two operating modes, Normal Mode, Random Number Mode, When in normal mode the lock expects the user to enter all four keys that represents the key combination, When in Random Number mode the user is expect to enter only two of the four keys and these two are randomly generated.

To Enter a key the slider switches are set then BUTTON(1) is pressed and the key is read the process is the same in both modes.

To reset BUTTON(0) is pressed.

To change mode to Random Number Mode BUTTON(2) is pressed.

DESIGN ORGANISATION

The system is broken into 5 modules, The Top-level module for the state machine, a display module, a clock generator module, a display multiplexer module ,a message multiplexer module, random number generator module and a display module, each module has only one responsibility.

MainModule:

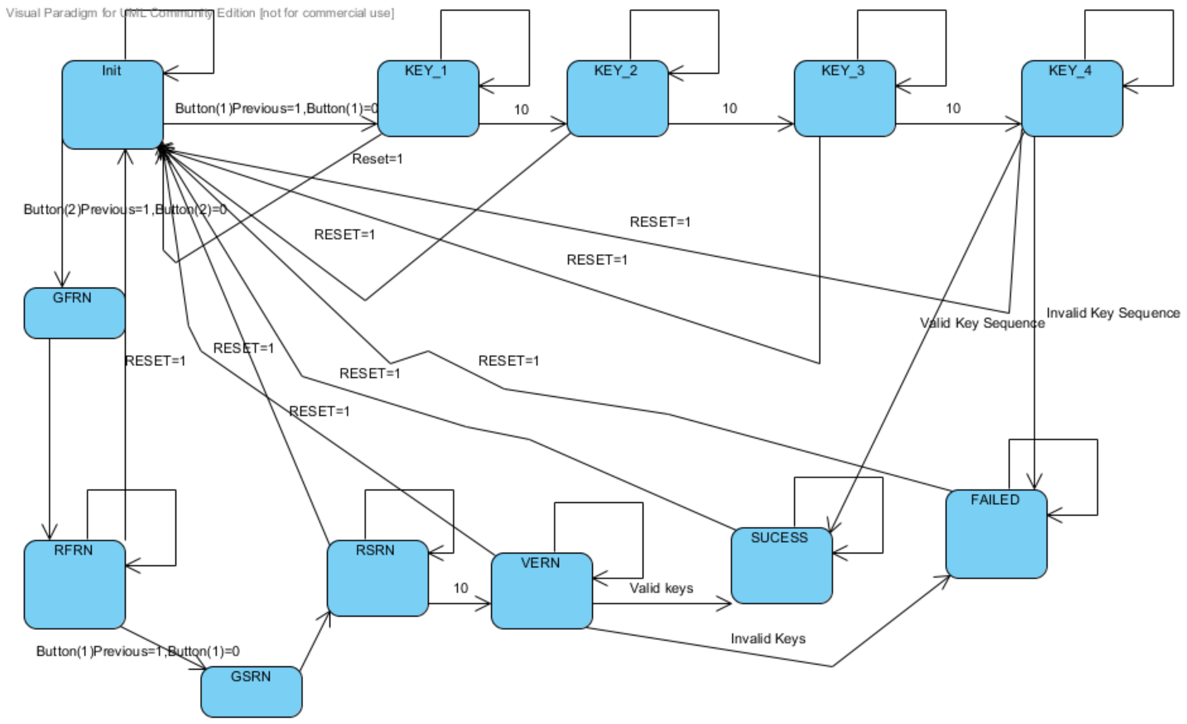

Figure 2

When powered the device is in state init, this is the state where all the device is in a known state and all initial values are zero.

The user adjusts the slider switches to set the first key   and then presses button(1),because the state machine is running at a much slower clock rate as generated by the internal clock the values of button(1) moves from 1 to 0 within a clock cycle when pressed, this reads the value in the slider switches and the state machine moves to the next state.

This process is repeated until the machine state of Key_4   at which all four keys have been read and at state key_4 all four keys are validated.

As the keys are been read   the entered keys are displayed on the Four seven segment Display ,this process is managed by two other modules namely the Display module and DisplayMultiplexer module.

If the four keys are valid the state machine moves to state Success else it moves to state Failed at state success and Eight Led's light up while at state failed all Led's Light up except the two most significant bits.

Also when in initial state and button(2) is pressed making it also go from 1 to 0 within a clock cycle the mode changes to random number mode and the state machine transit's to state GFRN(generate first random number), The required random number position is displayed on the seven segment it uses a zero based index and it immediately transits to RFRN(read first random number)

Once the slider switches are set button(1) is pressed to read the number and the state machine then transits to the GSRN(generate second random number), The required random number position is displayed on the seven segment it uses a zero based index and it immediately transits to RSRN (read second random number)

Once the slider switches are set button(1) is pressed to read the number and the state machine then transits to VERN(validate entered random number),at this state the entered numbers are validated

If validation is successful the state machine transits to state SUCCESS else it transits to state FAIL .

Also at state success the four seven segments displays an OK message and also flash's between the four keys entered and the message ,else at state failed the four seven segments displays Error message and also flash's the four keys entered and the message.

The operation of displaying messages and controlling the flashing is delegated to three modules the

Display module , MessageMultiplexer Module and DisplayMultiplexer module.

InternalClock:

This module is responsible for generating the clock signal required by the state machine and all other modules within the combinational lock system.

It is basically a 32bit counter with a bit input and a bit output ,the input is the external clock which drives the 32bit counter and the output is the rate of change of the 15th bit of the 32bit counter.

Display:

This is the module that actually drive's the seven segment display's this module has a four bit input and one 8 bit output.

The four bit input represents the number meant to be displayed and the 8 bit represents the driving patterns for the seven segments.

DisplayMultiplexer:

This module is responsible for letting the Display module create the impression it is driving more than one seven segment at a time.

It has one sixteen bit inputs which are all the four numbers the system wants' displayed at the same time, two four bit output and one clock input.

Of the two four bit output one (DigitM) represents the presently driven seven segments and the other (DV) represents the integer to be displayed.

The multiplexing operation is controlled by an internal 32 bit counter and the bits 10 and 9 of this counter are used as control signal to determine which seven segments to be driven and which digit to be driven by.

Because of the rate which this occurs and the resolution of the human eyes it creates the effect that all Seven segments a driven together at once.

MessageMultiplexer:

This module is responsible for creating the toggling effect between the entered code and the displayed message OK or Error.

This Module has two 16 bit input ,two 1 bit input and one 16 bit output ,the two 16 bit input represents the Key code sequence and the response message which are to be toggled ,the two 1 bit inputs represent the clock and the start bit(this indicates when the multiplexer should start).

The 16bit output represents that data that would be passed to the displaymultiplexer so that it could be displayed across the four seven segment display.

The operation is controlled by a 32 bit counter and the 26th bit of the counter controls which message to be displayed.

RandomNumber:

This module is responsible for generating two 2 bit random numbers.

It has two 1bit inputs and two 2bit outputs, the 1 bit inputs are for clock and control.

The 2 bit output represents the two random numbers.

The modules uses a 32 bit counter, the values in bit position XX and XX represents the first random number while an invert of this random number represents the second random number this is to ensure the two random numbers are never the same, the control bit was added for future modifications.

SIMULATION RESULTS

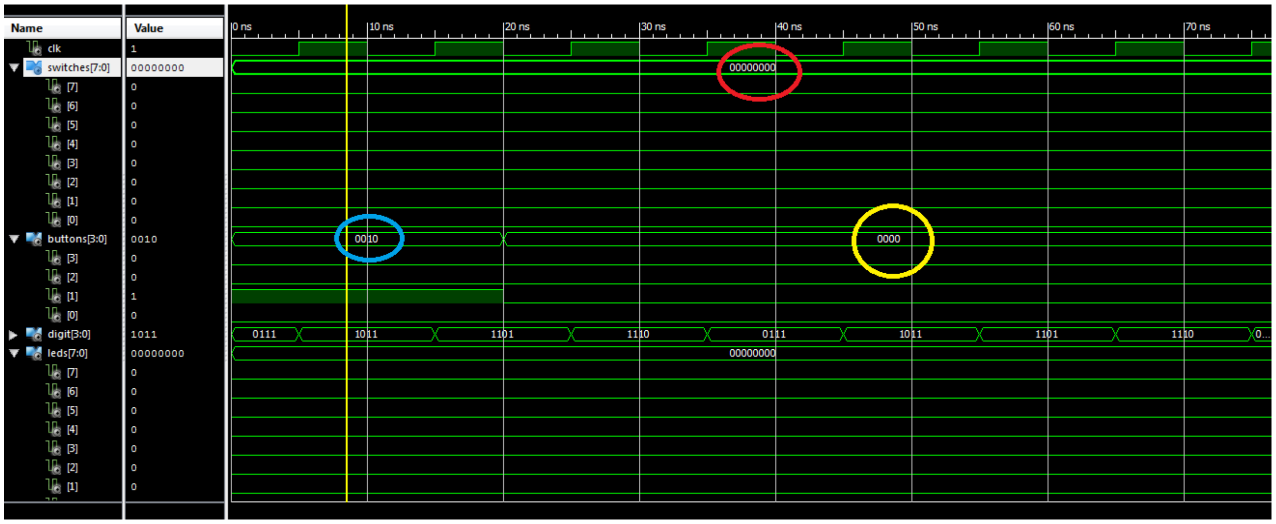

Figure 3,Key_1 State transition

This shows the press and release effect of button(1),at about 10ns button(1) circled in sky blue and at 20ns it is back to zero, while the switches are all at zero this is the first key in my sequence.

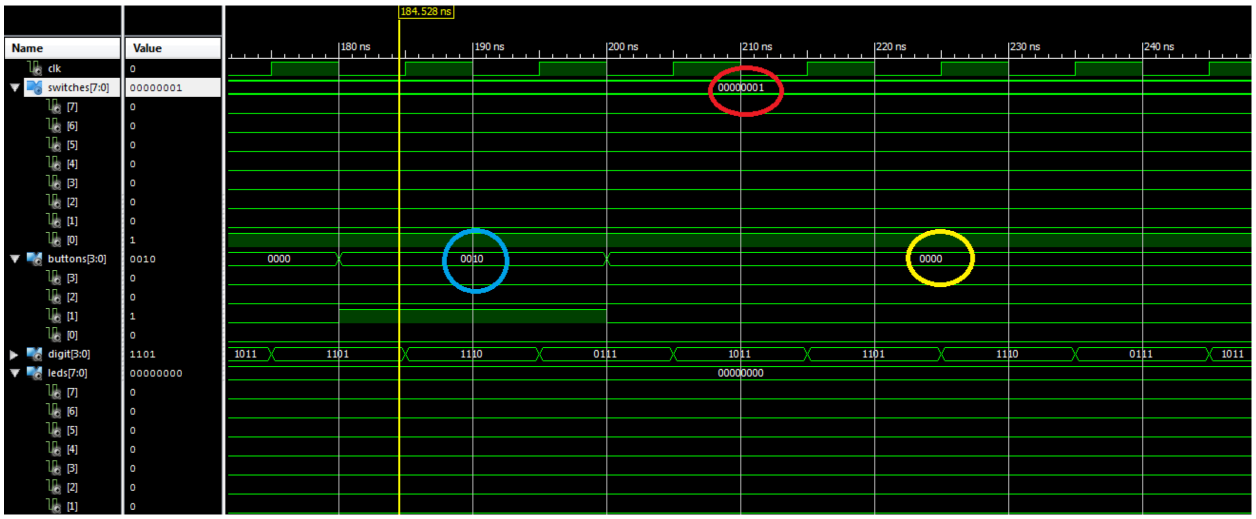

Figure 4,Key_2 state transition

This shows the press and release effect of button(1),at about 190ns button(1) circled in sky blue and at 200ns it is back to zero ,while the switches are all at number one (00000001)this is the second key in my sequence.

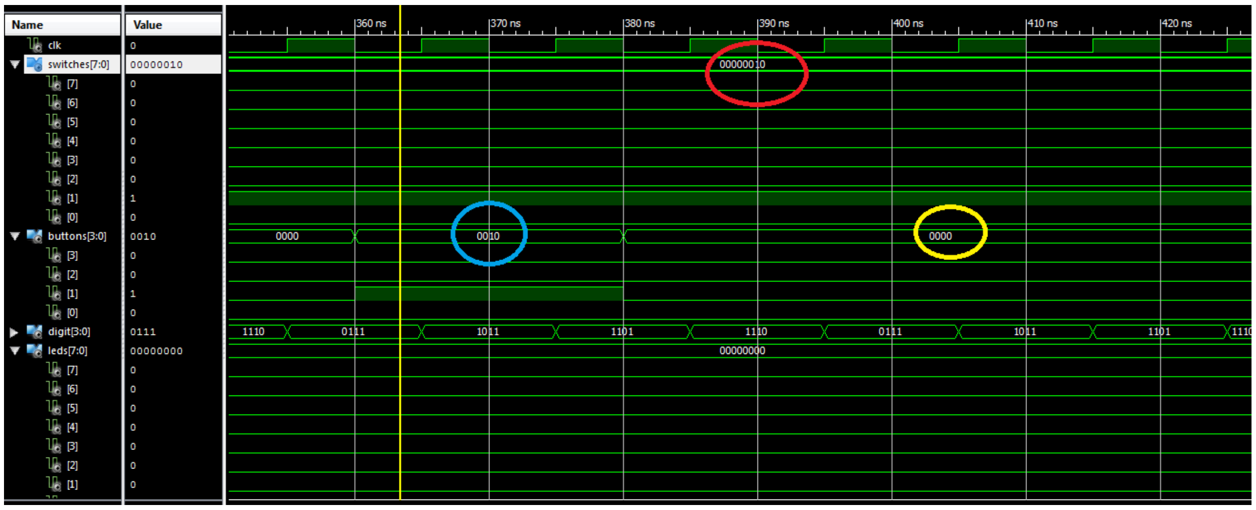

Figure 5,Key_3 State Transition

This shows the press and release effect of button(1),at about 370ns button(1) circled in sky blue and at 380ns it is back to zero ,while the switches are all at number two (00000010)this is the third key in my sequence.

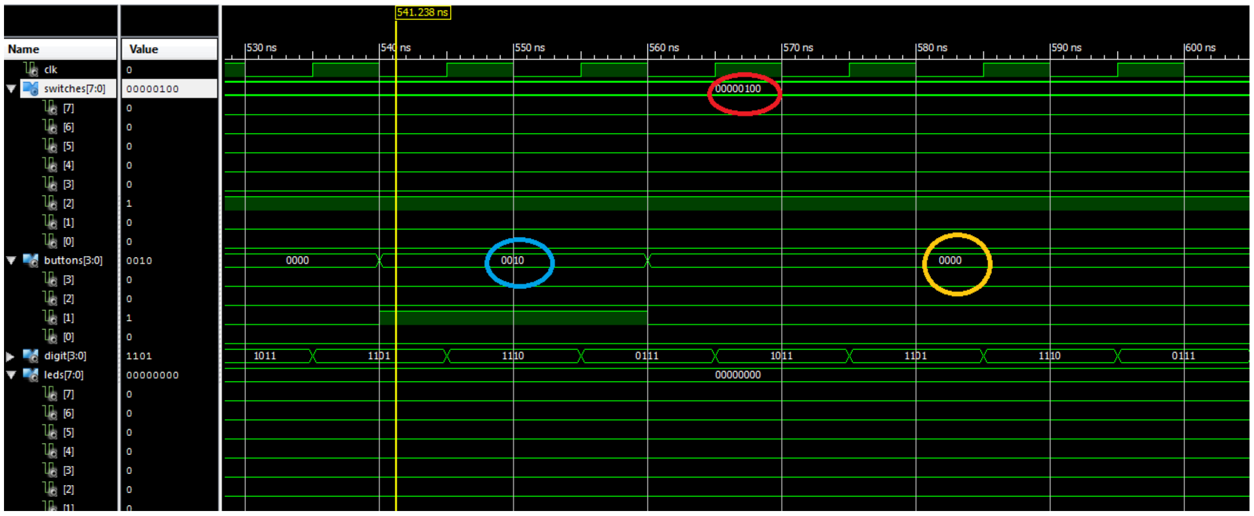

Figure 6,Key_4 state transition

This shows the press and release effect of button(1),at about 550ns button(1) circled in sky blue and at 560ns it is back to zero ,while the switches are all at number four (00000100)this is the fourth key in my sequence.

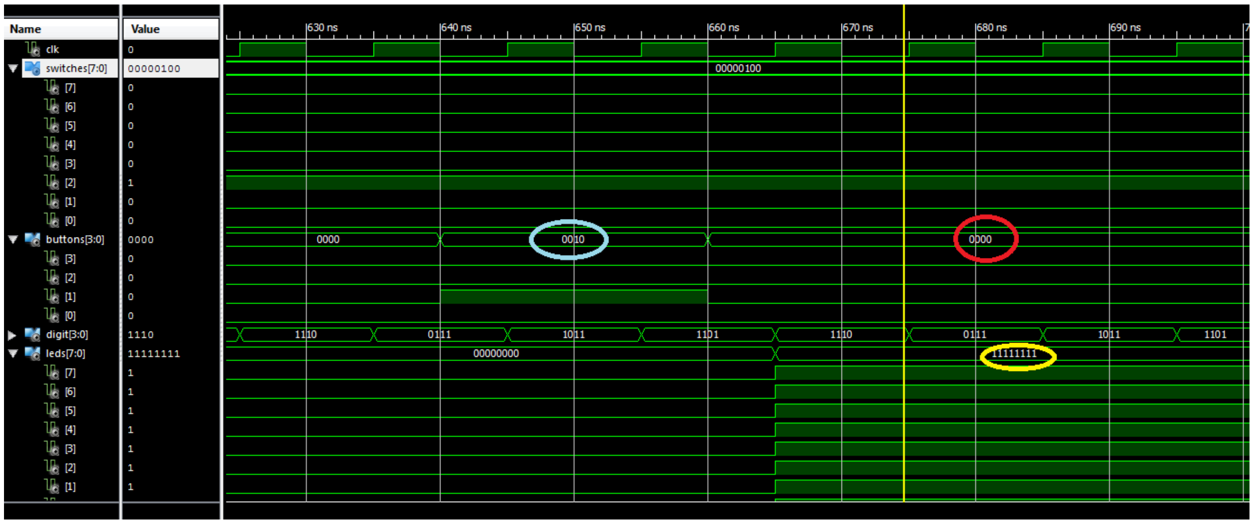

Figure 7,Validation Success

This shows the press and release effect of button(1),at about 650ns button(1) circled in sky blue and at 660ns it is back to zero ,this transit the state machine to success if the key sequence entered are correct, in the diagram above the key sequence was correct hence validation success as indicated by the LEDS(circled in yellow) all glowing .

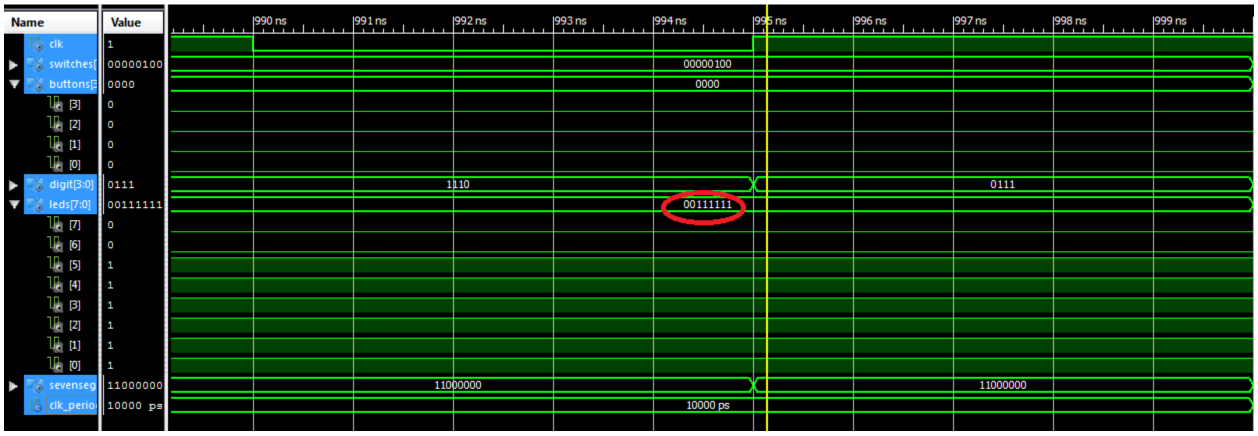

Figure 8,Validation failed

This shows the state of the LEDS(circled in red) if the validation fails.

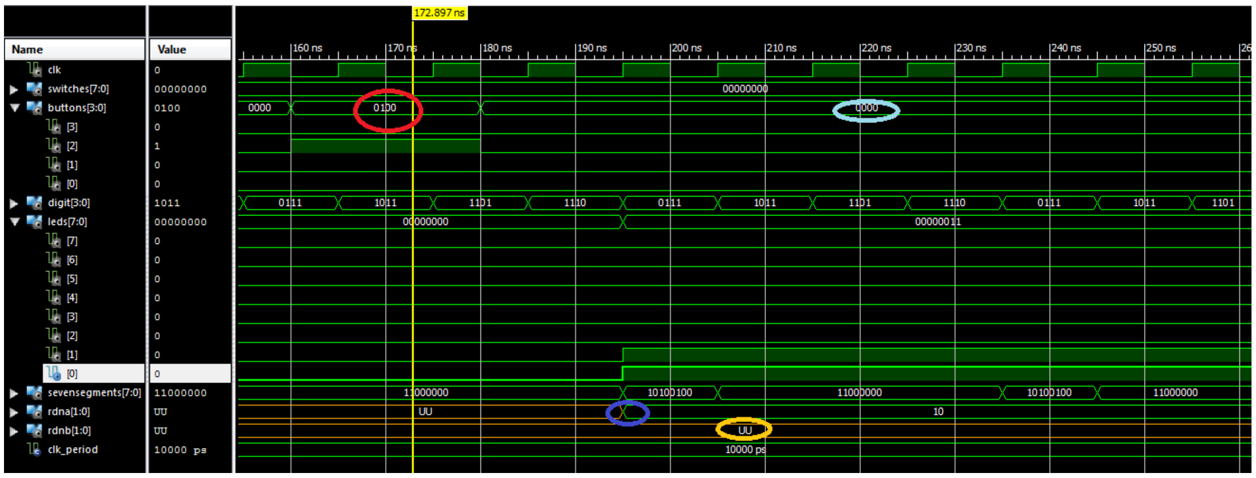

Figure 9,Generate first random number transition

This shows Button(2) been pressed at 170ns and released at 180ns and the first random no remain undefined until 195ns when the present release effect transits to generate first random number (10)and the second random number still remain undefined(circled in yellow).

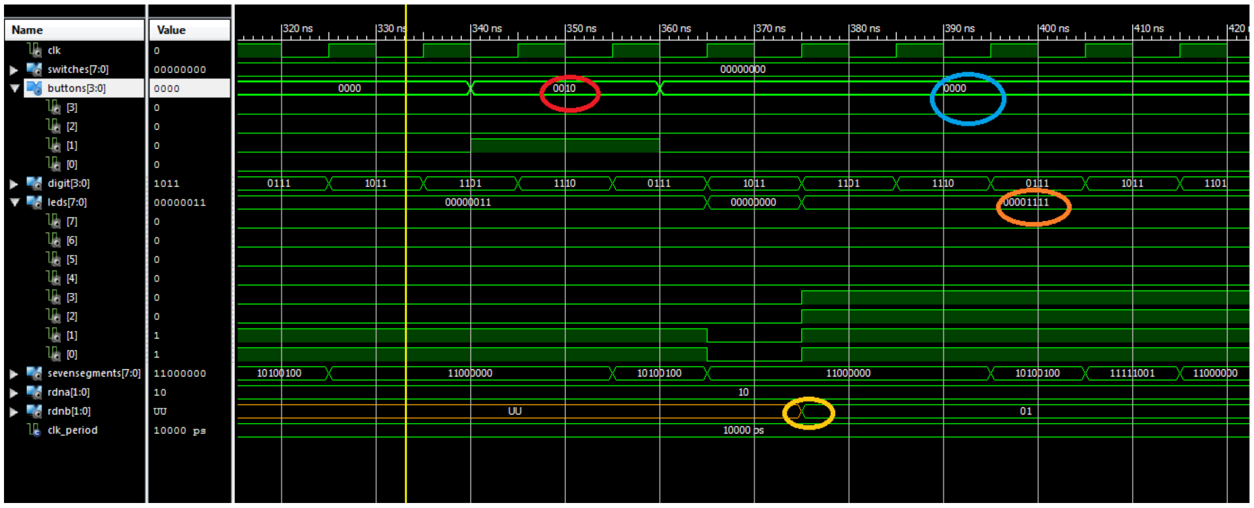

Figure 10,Read First Random Number and transit to generate Second Random Number

This shows Button(1) been pressed at 350ns (circled in red) and released at 360ns ,the value in the switches (00000000) are read in and the state machines transits to the generate second random number the values of the second random number (01)circled in yellow.

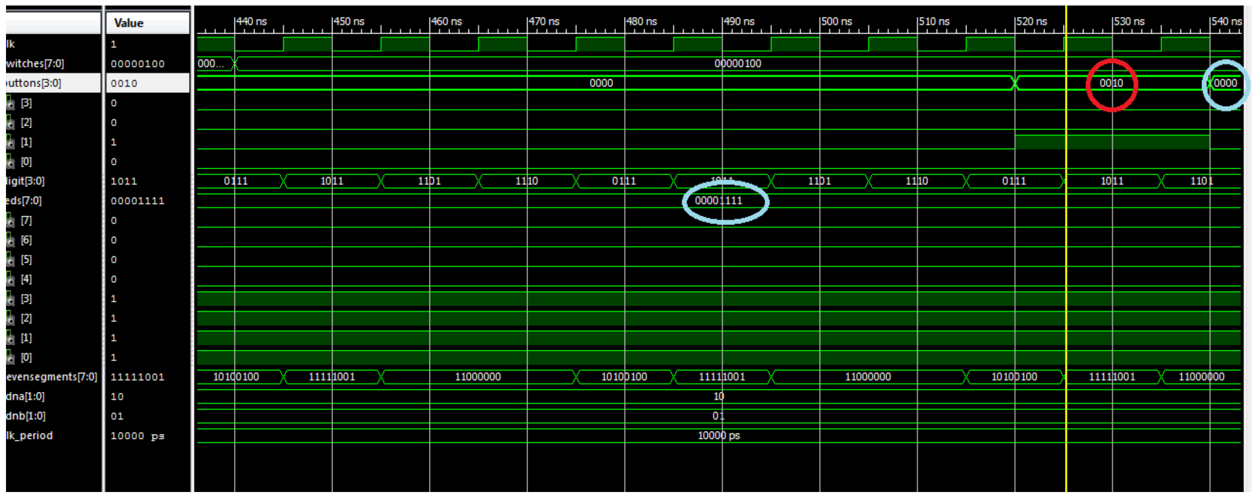
Figure 11,Read Second Random Number

This shows Button(1) been pressed at 530ns (circled in red) and released at 540ns ,the value in the switches (00000100) are read in and the state machines transits to the validate entered random number and in this case transits to validation failed state.

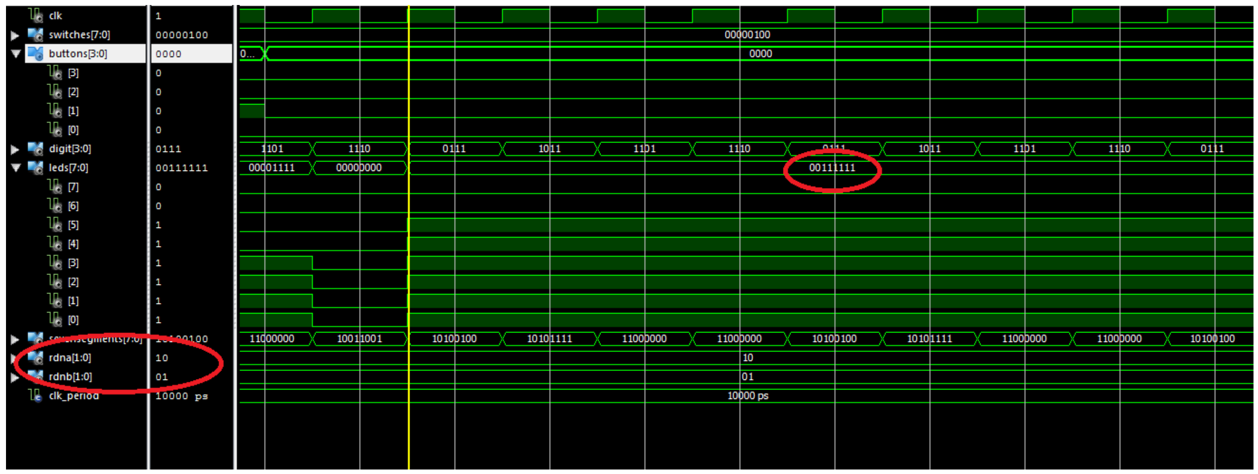
Figure 12,Random Number Mode Validation Failed

This shows the state of the LEDS(circled in red) if the validation fails, also in the random number positions

SYNTHESIS RESULTS

Device Utilization Summary (xc3s1000-5ft256)			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	80	15,360	1%
Number of 4 input LUTs	115	15,360	1%
Number of occupied Slices	85	7,680	1%
Number of Slices containing only related logic	85	85	100%
Number of Slices containing unrelated logic	0	85	0%
Total Number of 4 input LUTs	156	15,360	1%
Number used as logic	115		
Number used as a route-thru	41		
Number of bonded IOBs	33	173	19%
Number of BUFGMUXs	2	8	25%
Average Fan-out of Non-Clock Nets	2.85		

From this summary it can be deduced that if prototype was to go into production I would need the cheapest FPGA that has more than 115 4 input lookup tables, more 85 slice with more than 80 flip-flops per slice, more than 33 Input /output blocks.

Device Utilization Summary (estimated values)	(xc3s50-5pq208)		
Logic Utilization	Used	Available	Utilization
Number of Slices	82	768	10%
Number of Slice Flip Flops	80	1536	5%
Number of 4 input LUTs	159	1536	10%
Number of bonded IOBs	28	124	22%
Number of GCLKs	2	8	25%

From this summary ,compared with actual requirement I can conclude my design would fit into a Spartan Xc3s50,which is a lot cheaper than the 1000.

CONCLUSIONS

What I have learnt

- I   have learnt how organize a VHDL project in a modularized , reusable and easily testable way
- I have learnt how to simulate a VHDL project, figuring out that in a lot case you might need to change parts of your code in order to make it simulate able.
-

Possible Improvements

- The user interface could be improved, by adding some friendly text's to the seven segment display
- The system could be improved by adding a proper switch de-bouncing logic.

APPENDIX

InternalClock.vhd

```
---------------------------------------------------------------------------
--------
-- Company: The University of Birmingham
-- Engineer: Folorunsho Solomon Opeyemi
--
-- Create Date:    10:28:12 11/15/2012
-- Design Name:
-- Module Name:    InternalClock - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
---------------------------------------------------------------------------
--------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;



entity InternalClock is
    Port ( CLK : in  STD_LOGIC;
           OCLK : out  STD_LOGIC);
end InternalClock;

architecture Behavioral of InternalClock is
--interanl count variable
signal count  : std_logic_vector (31 downto 0):=x"00000000";
begin
process (CLK)
begin

if (rising_edge(CLK))
-- increase count
then count <= count + x"00000001";
end if;

end process;
```

```
--return new clock
OCLK<=count(15);
end Behavioral;
```

RandomNumber.vhd

```
--------------------------------------------------------------------------
--------
-- Company:
-- Engineer:
--
-- Create Date:    11:41:12 11/15/2012
-- Design Name:
-- Module Name:    RandomNumber - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
--------------------------------------------------------------------------
--------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity RandomNumber is
Port ( CLK : in  STD_LOGIC;
            STATUS: in STD_LOGIC;
            RNDNB : out  STD_LOGIC_VECTOR(1 downto 0);
       RNDN : out  STD_LOGIC_VECTOR(1 downto 0));
end RandomNumber;

architecture Behavioral of RandomNumber is
signal count  : std_logic_vector (31 downto 0):=x"00000000";
begin
process (CLK)

begin
```

```
if(rising_edge(clk)) then
--increase count
count <= count + x"00000001";
end if;
--Randomnumberone
RNDN <= count(10 downto 9);
--invert bit for randomnumber b
RNDNB(0) <=Not RANDONE(0);
RNDNB(1) <=Not RANDONE(1);

end process;
end Behavioral;
```

Display.vhd

```
---------------------------------------------------------------------------
--------
-- Company:
-- Engineer:
--
-- Create Date:    11:02:17 11/06/2012
-- Design Name:
-- Module Name:    Display - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
---------------------------------------------------------------------------
--------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;


-- 0    ---
--  5 |    | 1
--     ---   <- 6
--  4 |    | 2
--      ---
--       3

entity Display is
    Port ( number : in  STD_LOGIC_VECTOR (3 downto 0);
           segs : out  STD_LOGIC_VECTOR (7 downto 0));

end Display;

architecture Behavioral of Display is

begin
segs(7) <= '1';


with number SELect
   segs(6 downto 0 )<= "1111001" when "0001",   --1
         "0100100" when "0010",   --2
         "0110000" when "0011",   --3
         "0011001" when "0100",   --4
         "0010010" when "0101",   --5
         "0000010" when "0110",   --6
         "1111000" when "0111",   --7
         "0000000" when "1000",   --8
         "0010000" when "1001",   --9
         "0101111" when "1010",   --A
         "0001001" when "1011",   --b
         "1000110" when "1100",   --C
         "0100001" when "1101",   --d
         "0000110" when "1110",   --E
         "0001110" when "1111",   --F
         "1000000" when others;   --0

end Behavioral;
```

DisplayMultiplexer.vhd

```
---------------------------------------------------------------------------
--------
-- Company: Folorunsho Solomon Opeyemi
-- Engineer: The University of Birmingham
--
-- Create Date:    12:36:08 11/14/2012
-- Design Name:
-- Module Name:    DisplayMultiplexer - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
```

```
-- Revision 0.01 - File Created
-- Additional Comments:
--
---------------------------------------------------------------------------
--------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity DisplayMultiplexer is
port(
NA:in std_logic_vector(15 downto 0);
digitM : out  STD_LOGIC_VECTOR (3 downto 0);
DV: out std_logic_vector(3 downto 0);
clkM: in std_logic
);
attribute LOC:string;
end DisplayMultiplexer;

architecture Behavioral of DisplayMultiplexer is
signal count  : std_logic_vector (31 downto 0):=x"00000000";
signal control :std_logic_vector(1 downto 0) :="00";
begin
process (clkM)
begin

if (rising_edge(clkM))
then count <= count + x"00000001";
end if;

end process;

control<=count(10 downto 9);

DV <= NA(3 downto 0) when control="00"
           else NA(7 downto 4)  when control="01"
           else NA(11 downto 8) when control="10"
           else NA(15 downto 12) ;

digitM <= "0111" when control="00"
                 ELSE "1011" when control="01"
                 ELSE "1101" when control="10"
                 ELSE "1110";
end Behavioral;
```

MessageMultiplexer.vhd

```
----------------------------------------------------------------------------
--------
-- Company: The University of Birmingham
-- Engineer: Folorunsho Solomon Opeyemi
--
-- Create Date:    11:38:24 11/15/2012
-- Design Name:
-- Module Name:    MessageMultiplexer - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
--------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MessageMultiplexer is
    Port ( MA : in  STD_LOGIC_VECTOR (15 downto 0);
           MB : in  STD_LOGIC_VECTOR (15 downto 0);
           MC : out  STD_LOGIC_VECTOR (15 downto 0);
           CLK : in  STD_LOGIC;
                   START: in STD_LOGIC
                   );
end MessageMultiplexer;

architecture Behavioral of MessageMultiplexer is
signal count  : std_logic_vector (31 downto 0):=x"00000000";
signal control: std_logic;
begin
process (CLK)
begin

if (rising_edge(CLK))
then count <= count + x"00000001";
```

```
end if;

end process;
control<=count(26);
MC <= MA when control ='0' and START ='1' else
           MB  when control= '1' and START='1' else
           MA;
end Behavioral;
```

MainModule.vhd

```
--------------------------------------------------------------------------
--------
-- Company: University of Birmingham
-- Engineer:Folorunsho Solomon Opeyemi
--
-- Create Date:    11:38:26 11/08/2012
-- Design Name:  Combinational Lock
-- Module Name:    MainModule - Behavioral
-- Project Name: Advanced Digital design
-- Target Devices: Spartan
-- Tool versions:
-- Description:
--
-- Dependencies: Display
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
--------------------------------------------------------------------------
--------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MainModule is
port(
CLK                    : in  STD_LOGIC;
SWITCHES               : in  STD_LOGIC_VECTOR (7 downto 0);
BUTTONS                : in  STD_LOGIC_VECTOR (3 downto 0);
digit             : out  STD_LOGIC_VECTOR (3 downto 0);
LEDS              : out  STD_LOGIC_VECTOR (7 downto 0);
SEVENSEGMENTS     : out std_logic_vector(7 downto 0)
);
```

```
                 attribute LOC : string ;
         attribute LOC of CLK : signal is "T9";
         attribute LOC of SWITCHES : signal is
"K13,K14,J13,J14,H13,H14,G12,F12";
         attribute LOC of BUTTONS : signal is "L14,L13,M14,M13";
         attribute LOC of LEDS : signal is
"P11,P12,N12,P13,N14,L12,P14,K12";
         attribute LOC of digit : signal is "E13,F14,G14,D14";
         attribute LOC of SEVENSEGMENTS : signal is
"P16,N16,F13,R16,P15,N15,G13,E14";
end MainModule;

architecture Behavioral of MainModule is

type LockState is
(init,Key_0,Key_1,Key_2,Key_3,Key_4,GFRN,RFRN,GSRN,RSRN,VERN,Validate,Vali
dationSuccess,ValidationFailed);
signal presentstate :LockState:=init;
signal enterednumber : std_logic_vector(7 downto 0):="11111111";
signal fourbit :std_logic_vector(3 downto 0);
signal LargeNumber:std_logic_vector(15 downto 0):="0000000000000000";
signal BtnOneHistory:std_logic;
signal BtnTwoHistory:std_logic;
signal MyClk:std_logic;
signal Message: std_logic_vector(15 downto 0) :="0000000000000000";
signal MessageM: std_logic_vector(15 downto 0) :="0000000000000000";
signal START: std_logic:='0';
signal RNDN :std_logic_vector(1 downto 0);
signal RNDNB :std_logic_vector(1 downto 0);
signal FirstRandNo :std_logic_vector(1 downto 0);
signal SecondRandNo :std_logic_vector(1 downto 0);
COMPONENT Display
      PORT(
            number : IN std_logic_vector(3 downto 0);
            segs : OUT std_logic_vector(7 downto 0)
            );
      END COMPONENT;

      COMPONENT DisplayMultiplexer
      PORT(
            NA:in std_logic_vector(15 downto 0);
            digitM : out  STD_LOGIC_VECTOR (3 downto 0);
            DV: out std_logic_vector(3 downto 0);
            clkM: in std_logic
            );
      END COMPONENT;

      COMPONENT InternalClock

            Port
            (
                   CLK : in  STD_LOGIC;
           OCLK : out  STD_LOGIC
                    );
      END COMPONENT;

      COMPONENT MessageMultiplexer

            Port ( MA : in  STD_LOGIC_VECTOR (15 downto 0);
           MB : in  STD_LOGIC_VECTOR (15 downto 0);
           MC : out  STD_LOGIC_VECTOR (15 downto 0);
           CLK : in  STD_LOGIC;
                   START: in STD_LOGIC
                   );

      END COMPONENT;

      COMPONENT RandomNumber

            Port (
            CLK :      in  STD_LOGIC;
            RNDNB : out  STD_LOGIC_VECTOR(1 downto 0);
      RNDN :     out  STD_LOGIC_VECTOR(1 downto 0));

      END COMPONENT;
begin


process(CLK,BUTTONS(0))
variable IsFRNV : std_logic;
variable IsSRNV : std_logic;
begin
if(BUTTONS(0) = '1') then
--reset
presentstate <= init;
LargeNumber<="0000000000000000";
START <= '0';
elsif (rising_edge(MyClk)) then
--Save initial state
BtnOneHistory <=BUTTONS(1);
BtnTwoHistory <=BUTTONS(2);
case presentstate is

when init => if BUTTONS(1)= '0' and BtnOneHistory = '1' then

                       LargeNumber(3 downto 0) <= switches(3 downto 0);
                       presentstate <= Key_1;
                       elsif  BUTTONS(2)= '0' and BtnTwoHistory = '1' then
                       FirstRandNo <= RNDN;
                       SecondRandNo <= RNDNB;
                       presentstate <= GFRN;
                       else
                        presentstate <= init;
                        end if;
when Key_1 => if BUTTONS(1)= '0' and BtnOneHistory = '1' then

                        LargeNumber(7 downto 4) <= switches(3 downto 0);
                        presentstate <= Key_2;
                        else
                        presentstate <= Key_1;
                        end if;
when Key_2 => if BUTTONS(1)= '0' and BtnOneHistory = '1' then

                         LargeNumber(11 downto 8) <= switches(3 downto 0);
                          presentstate <= Key_3;
                        else
                        presentstate <= Key_2;
                        end if;
when Key_3 => if BUTTONS(1)= '0' and BtnOneHistory = '1' then

                         LargeNumber(15 downto 12) <= switches(3 downto
0);
                          presentstate <= Key_4;
                        else
                        presentstate <= Key_3;
                        end if;
when Key_4 =>
if BUTTONS(1)= '0' and BtnOneHistory = '1' then

 if LargeNumber(3 downto 0) ="00000000" and LargeNumber(7 downto
4)="00000001" and LargeNumber(11 downto 8)="00000010" and LargeNumber(15
downto 12)="00000100" then
  presentstate<=ValidationSuccess;
  Message <="0000000010110000";
  START <= '1';
 else
  presentstate<=ValidationFailed;
  Message<="0000101010101110";
  START <= '1';
  end if;
end if;

when GFRN=>

LargeNumber(3 downto 0) <= "00" & FirstRandNo;
presentstate <=RFRN;

when RFRN=>
                             if BUTTONS(1)= '0' and BtnOneHistory = '1'
then

                         LargeNumber(11 downto 8) <= switches(3 downto 0);
                          presentstate <= GSRN;

                        else
                        presentstate <= RFRN;
                        end if;
when GSRN=>

LargeNumber(7 downto 4) <= "00" & SecondRandNo;
presentstate <=RSRN;
```

```
when RSRN=>
                       if BUTTONS(1)= '0' and BtnOneHistory = '1' then

                         LargeNumber(15 downto 12) <= switches(3 downto
0);
                          presentstate <= VERN;

                        else
                        presentstate <= RSRN;
                        end if;
when VERN=>


CASE FirstRandNo IS
    WHEN  "00"  =>
      if LargeNumber(11 downto 8) = "0000" then
      IsFRNV := '1';
      else
      IsFRNV :=  '0';
      end if;
    WHEN  "01"  =>
            if LargeNumber(11 downto 8) = "0001" then
            IsFRNV := '1';
            else
            IsFRNV :=  '0';
      end if;
    WHEN  "10"  =>
            if LargeNumber(11 downto 8) = "0010" then
            IsFRNV := '1';
            else
            IsFRNV :=  '0';
            end if;
    WHEN  "11"  =>
       if LargeNumber(11 downto 8) = "0100" then
            IsFRNV := '1';
            else
            IsFRNV :=  '0';
            end if;
    WHEN OTHERS =>  IsFRNV := '0';
  END CASE;


  CASE SecondRandNo IS
    WHEN  "00"  =>
      if LargeNumber(15 downto 12) = "0000" then
      IsSRNV := '1';
      else
      IsSRNV :=  '0';
      end if;
    WHEN  "01"  =>
            if LargeNumber(15 downto 12) = "0001" then
            IsSRNV := '1';
            else
            IsSRNV :=  '0';
       end if;
     WHEN  "10"  =>
            if LargeNumber(15 downto 12) = "0010" then
            IsSRNV := '1';
            else
            IsSRNV :=  '0';
            end if;
     WHEN  "11"  =>
        if LargeNumber(15 downto 12) = "0100" then
            IsSRNV := '1';
            else
            IsSRNV :=  '0';
            end if;
     WHEN OTHERS =>  IsSRNV := '0';
   END CASE;

if IsSRNV ='1' and IsFRNV ='1' then
presentstate<=ValidationSuccess;
Message <="0000000010110000";
 START <= '1';
else
presentstate <= ValidationFailed;
Message<="0000101010101110";
  START <= '1';
end if;
when ValidationSuccess => presentstate <= ValidationSuccess;
when ValidationFailed => presentstate <= ValidationFailed;

when others =>presentstate<= init;
end case;
end if;
end process;


LEDS <=     "11111111"  when  presentstate = ValidationSuccess else
         "00111111" when presentstate = ValidationFailed else
                 "00000011" when presentstate = RFRN else
                 "00001111" when presentstate = RSRN
                  else "00000000";


dd: Display PORT MAP(number=>fourbit,segs=> SEVENSEGMENTS);
DM :DisplayMultiplexer PORT
MAP(clkM=>CLK,NA=>MessageM,digitM=>digit,DV=>fourbit);
MC:InternalClock PORT MAP(CLK=>CLK,OCLK=>MyClk);
MM :MessageMultiplexer PORT
MAP(CLK=>CLK,MA=>LargeNumber,MB=>Message,MC=>MessageM,START=>START);
RN :RandomNumber PORT MAP(CLK=>CLK,RNDN=>RNDN,RNDNB=>RNDNB);
end Behavioral;
```

Mainodule_tb.vhd

```
---------------------------------------------------------------------------
------
-- Company:
-- Engineer:
--
-- Create Date:   10:46:01 12/03/2012
-- Design Name:
-- Module Name:
Y:/Documents/DesignsHDL/CombinationalLock/Mainodule_tb.vhd
-- Project Name:  CombinationalLock
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: MainModule
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic
and
-- std_logic_vector for the ports of the unit under test.  Xilinx
recommends
-- that these types always be used for the top-level I/O of a design in
order
-- to guarantee that the testbench will bind correctly to the post-
implementation
-- simulation model.
---------------------------------------------------------------------------
------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY Mainodule_tb IS
END Mainodule_tb;

ARCHITECTURE behavior OF Mainodule_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT MainModule
    PORT(
         CLK : IN  std_logic;
         SWITCHES : IN  std_logic_vector(7 downto 0);
         BUTTONS : IN  std_logic_vector(3 downto 0);
         digit : OUT  std_logic_vector(3 downto 0);
         LEDS : OUT  std_logic_vector(7 downto 0);
                 RdnA:OUT std_logic_vector(1 downto 0);
                 RdnB:OUT std_logic_vector(1 downto 0);
         SEVENSEGMENTS : OUT  std_logic_vector(7 downto 0)

        );
    END COMPONENT;


   --Inputs
   signal CLK : std_logic := '0';
   signal SWITCHES : std_logic_vector(7 downto 0) := (others => '0');
   signal BUTTONS : std_logic_vector(3 downto 0) := (others => '0');

      --Outputs
   signal digit : std_logic_vector(3 downto 0);
   signal LEDS : std_logic_vector(7 downto 0);
   signal SEVENSEGMENTS : std_logic_vector(7 downto 0);
      signal RdnA :std_logic_vector(1 downto 0);
      signal RdnB :std_logic_vector(1 downto 0);
   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: MainModule PORT MAP (
          CLK => CLK,
          SWITCHES => SWITCHES,
          BUTTONS => BUTTONS,
          digit => digit,
          LEDS => LEDS,
                  RdnA =>RdnA,
                  RdnB =>RdnB,
          SEVENSEGMENTS => SEVENSEGMENTS
        );

   -- Clock process definitions
   CLK_process :process
   begin
           CLK <= '0';
           wait for CLK_period/2;
           CLK <= '1';
           wait for CLK_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
--valid sequence
      BUTTONS(1) <= '1';
        wait for 20 ns;
```

```
  BUTTONS(1) <= '0';
  wait for 80 ns;

  SWITCHES <="00000001";
  wait for 80 ns;
  BUTTONS(1) <= '1';
  wait for 20 ns;
  BUTTONS(1) <= '0';
  wait for 80 ns;

  SWITCHES <="00000010";
  wait for 80 ns;
  BUTTONS(1) <= '1';
  wait for 20 ns;
  BUTTONS(1) <= '0';
  wait for 80 ns;

  SWITCHES <="00000100";
  wait for 80 ns;
  BUTTONS(1) <= '1';
  wait for 20 ns;
  BUTTONS(1) <= '0';
  wait for 80 ns;

  BUTTONS(1) <= '1';
  wait for 20 ns;
  BUTTONS(1) <= '0';

  wait for 100 ns;
---Invalid Sequence
     wait for 50 ns;
     BUTTONS(0) <= '1';
     wait for 10 ns;
     BUTTONS(0) <= '0';

     wait for 100 ns;

     SWITCHES <="00000001";
     wait for 80 ns;
     BUTTONS(1) <= '1';
     wait for 20 ns;
     BUTTONS(1) <= '0';
     wait for 80 ns;

     SWITCHES <="00000001";
     wait for 80 ns;
     BUTTONS(1) <= '1';
     wait for 20 ns;
     BUTTONS(1) <= '0';
     wait for 80 ns;

     SWITCHES <="00000001";
     wait for 80 ns;
     BUTTONS(1) <= '1';
```

```
        wait for 20 ns;
        BUTTONS(1) <= '0';
        wait for 80 ns;

        SWITCHES <="00000001";
        wait for 80 ns;
        BUTTONS(1) <= '1';
        wait for 20 ns;
        BUTTONS(1) <= '0';
        wait for 80 ns;

        BUTTONS(1) <= '1';
        wait for 20 ns;
        BUTTONS(1) <= '0';



        ---Invalid Random Number
        wait for 50 ns;
        BUTTONS(0) <= '1';
        wait for 10 ns;
        BUTTONS(0) <= '0';

        wait for 100 ns;

        BUTTONS(2) <= '1';
        wait for 20 ns;
        BUTTONS(2) <= '0';
        wait for 80 ns;

        SWITCHES <="00000000";
        wait for 80 ns;
        BUTTONS(1) <= '1';
        wait for 20 ns;
        BUTTONS(1) <= '0';
        wait for 80 ns;



        SWITCHES <="00000100";
        wait for 80 ns;
        BUTTONS(1) <= '1';
        wait for 20 ns;
        BUTTONS(1) <= '0';
        wait for 80 ns;

    ---valid random number

        wait for 50 ns;
        BUTTONS(0) <= '1';
        wait for 10 ns;
        BUTTONS(0) <= '0';

        wait for 100 ns;
        BUTTONS(2) <= '1';
        wait for 20 ns;
        BUTTONS(2) <= '0';
        wait for 80 ns;

        CASE RdnA IS
        WHEN  "00"  =>
        SWITCHES <="00000000";

        WHEN  "01"  =>
        SWITCHES <="00000001";

        WHEN  "10"  =>
        SWITCHES <="00000010";
        WHEN  "11"  =>
        SWITCHES <="00000100";
        WHEN OTHERS => SWITCHES <="00000000";
        END CASE;

        SWITCHES <="00000000";
        wait for 80 ns;
        BUTTONS(1) <= '1';
        wait for 20 ns;
        BUTTONS(1) <= '0';
        wait for 80 ns;



        CASE RdnB IS
        WHEN  "00"  =>
        SWITCHES <="00000000";

        WHEN  "01"  =>
        SWITCHES <="00000001";

        WHEN  "10"  =>
        SWITCHES <="00000010";
        WHEN  "11"  =>
        SWITCHES <="00000100";
        WHEN OTHERS => SWITCHES <="00000000";
        END CASE;

        SWITCHES <="00000000";
        wait for 80 ns;
        BUTTONS(1) <= '1';
        wait for 20 ns;
        BUTTONS(1) <= '0';
        wait for 80 ns;
        BUTTONS(1) <= '1';
        wait for 20 ns;
        BUTTONS(1) <= '0';
        wait for 80 ns;
  end process;
```

```
END;
```