You Shall Not Pass

Final Report

Jacqueline Radding | Carter O'Neill

CPE133

Contents

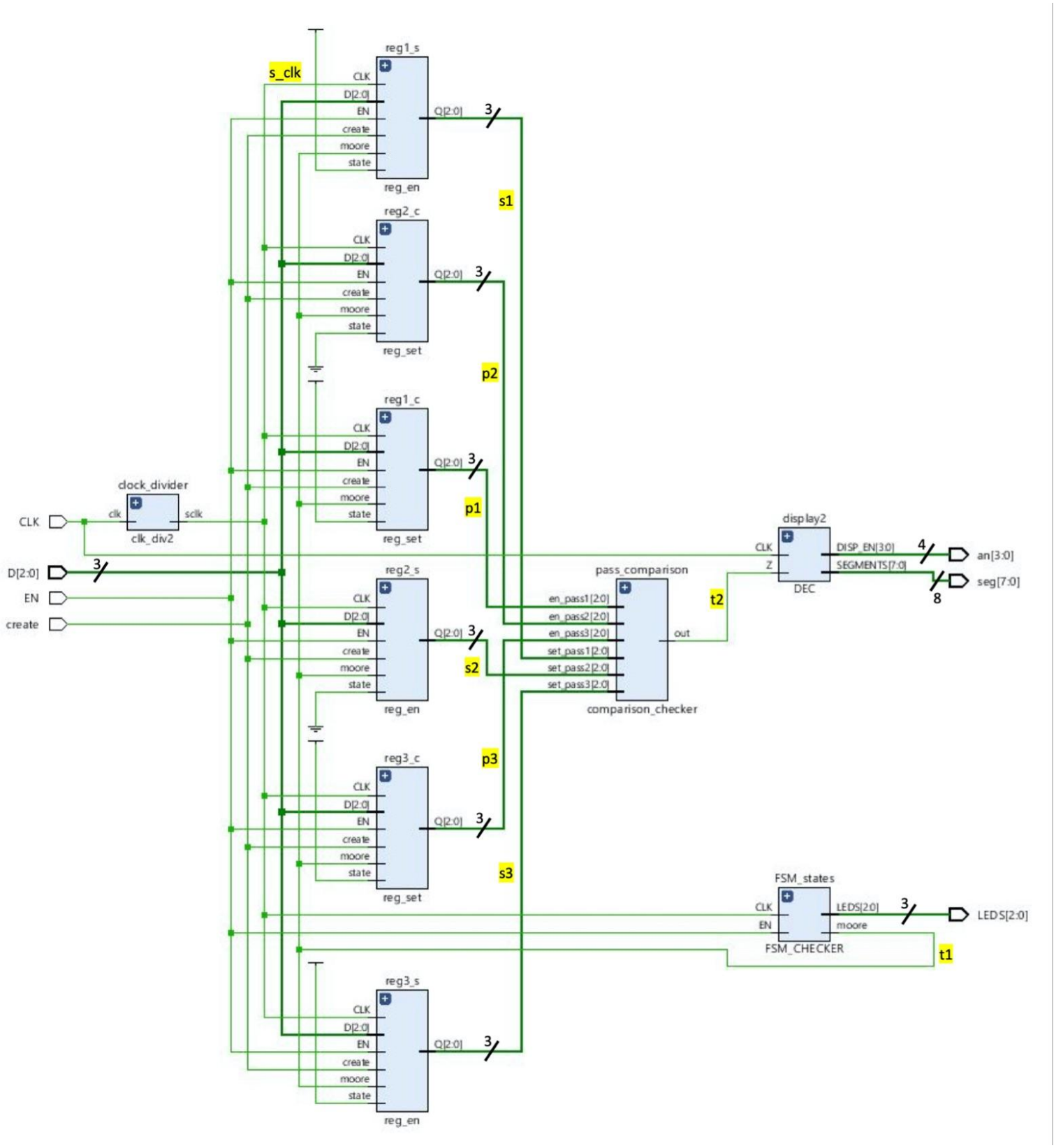
About the Project

You Shall Not Pass is a digital lockbox that accepts inputs ranging from 0-7 from three switches that represent the three bits necessary for inputting the numbers 0-7. It accepts three such 3-bit inputs and saves them as a password. When these three inputs are correct and in the right sequence, the Basys 3 board's seven segment display says "Open." The seven segment will say "Nope" when the three bit passwords are entered incorrectly.

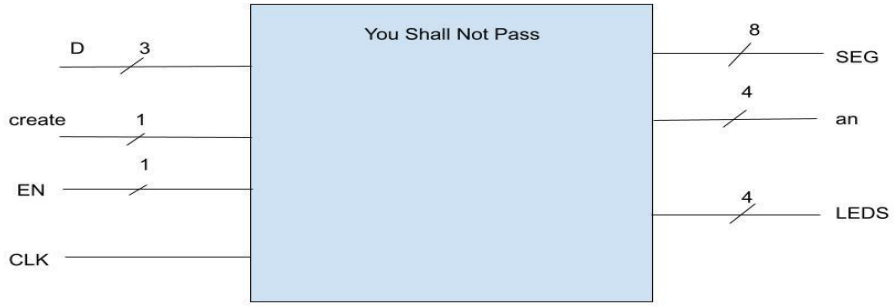
The inputs to the device include D, the input switches; EN, the enter button; and create, the create password switch. Pressing the enter button stores a user's input in a register, then advances the user to the next digit in the password through a finite state machine that activates the registers in succession. LEDs on the Basys board also turn on in sequence as the enter button is pressed to indicate how many password digits the user has inputted. The create switch allows the user to toggle between the two sets of registers: the password-setting registers or the password-guessing registers. When the create switch is on, the password-setting registers (the "set registers") are enabled, allowing the user to set the password on the device. When the create switch is off, the password-guessing registers (the "enter registers") are enabled, which allows the user to enter the predetermined password. The comparator module compares the values of each corresponding pair of enter and set registers, outputting a 1 if all three enter registers match their corresponding set registers and outputting 0 in all other cases. The lockbox will only open if the output value from the comparator is 1, indicating that the user inputted all three password digits correctly.

In a practical application, the "create" switch would be hidden on the inside of the lockbox so that the password can only be changed by a user who knows the current set password.

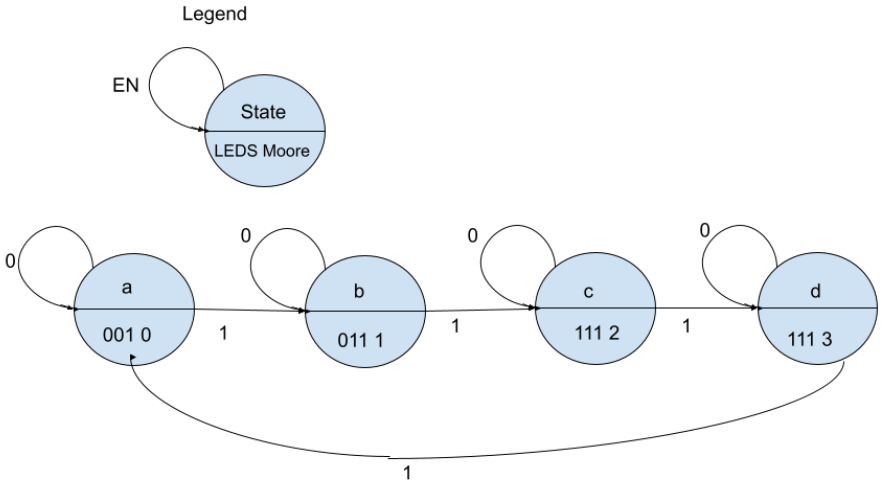
Structural Diagram



High Level Black Box



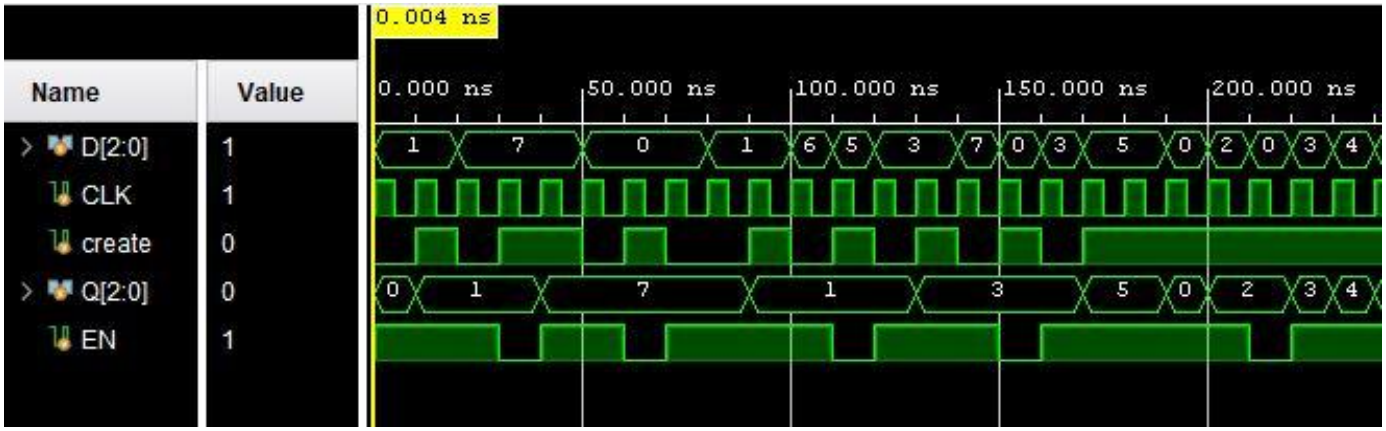
Finite State Machine



The finite state machine is responsible for outputting the number of passwords recorded, which is the moore variable, and the LEDS indicate what state the user is when entering one of the three passwords. When the enter button (EN) is pressed, the FSM progresses to the next state. When the enter button is not pressed, the FSM remains in the same state.

Simulation

Simulation Timing Diagram for reg_en:



The simulation of the set password register properly reacts to the create button, enter button (EN), clock (CLK), and switches (D). The memory register has active low sync load behavior (with create and enable) that saves the three bit password when create mode is on and when state matches FSM pass state.

Sim Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Carter O'Neill
//
// Create Date: 08/21/2021 01:36:36 PM
// Design Name:
// Module Name: reg_setSIM
// Project Name: YSNP
// Target Devices:
// Tool Versions:
// Description: Tests the set register as a simulation.
//
```

```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module reg_setSIM(

);

logic [2:0] D; //SWITCHES
logic CLK;
logic create;
 logic [2:0] Q; // this is Q
logic  EN;// enter button


    reg_set reg_set_inst (.D(D), .CLK(CLK), .create(create), .Q(Q), .EN(EN));

    always
    begin
    CLK = 1;
    #5;
    CLK = 0;
    #5;
    end
    initial
    begin

    EN =1; D = 3'b001; create =0;
      #10
    EN =1; D = 3'b001;  create =1;
         #10
    EN =1; D = 3'b111;  create =0;
            #10
    EN =0; D = 3'b111;  create =1;
            #10
    EN =1; D = 3'b111; create =1;
                #10
    EN =1; D = 3'b000; create =0;
                 #10
```

```
  EN =0; D = 3'b000; create =1;
                  #10
  EN =1; D = 3'b000; create =0;
  #10
    EN =1; D = 3'b001; create =0;
   #10
  EN =1; D = 3'b001;  create =1;
       #10
  EN =1; D = 3'b110;  create =0;
          #10
  EN =0; D = 3'b101;  create =1;
          #10
  EN =1; D = 3'b011; create =0;
              #10
  EN =1; D = 3'b011; create =1;
               #10
  EN =1; D = 3'b111; create =0;
                  #10
  EN =0; D = 3'b000; create =0;
                   #10
   EN =0; D = 3'b000;  create =1;
          #10
  EN =1; D = 3'b011; create =0;
              #10
  EN =1; D = 3'b101; create =1;
               #10
  EN =1; D = 3'b101; create =1;
                  #10
  EN =1; D = 3'b000; create =1;
   #10
  EN =1; D = 3'b010; create =1;
   #10
      EN =0; D = 3'b000;  create =1;
          #10
  EN =1; D = 3'b011; create =1;
              #10
  EN =1; D = 3'b100; create =1;
               #10
  EN =1; D = 3'b101; create =1;
                  #10
  EN =0; D = 3'b000; create =0;
  end
endmodule
```

Final Code

Top Module:

```
timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Cal Poly CPE 133
// Engineer: Jacqueline Radding and Carter O'Neill
//
// Create Date: 08/18/2021 07:57:02 PM
// Design Name:
// Module Name: YSNP_TOPLEVEL
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module YSNP_TOPLEVEL(
input CLK, //clock
    input [2:0]D, // switches
    input EN, // enable button
    input create, // if 1, create password mode, if 0, enter pass mode
    output [7:0]seg, // seven segment out on display
    output [3:0]an, // an out on display
    output [2:0]LEDS // LED output to indicate what state of password entering sequence

    );
    logic t1, t2, s_clk, out;
    logic [2:0] p1, p2, p3;//password created register memory outputs
    logic [2:0]s1,s2,s3; //password entered register memory outputs

    clk_div2 clock_divider(.CLK(CLK), .sclk(s_clk)); // clock dividing module
    FSM_CHECKER FSM_states (.CLK(s_clk), .EN(EN), .LEDS(LEDS), .moore(t1)); // FSM
indicates password recorded and entered state

    reg_set reg1_c(.CLK(s_clk), .D(D), .EN(EN), .state(1), .moore(t1),.create(create), .Q(p1)); //
triggered if create is 1 and FSM is in same password state
```

```
    reg_set reg2_c(.CLK(s_clk), .D(D), .EN(EN), .state(2), .moore(t1),.create(create), .Q(p2)); // outputs a three bit password to be compared to matching en state pass
    reg_set reg3_c(.CLK(s_clk), .D(D), .EN(EN), .state(3), .moore(t1),.create(create),.Q(p3));

    reg_en reg1_s(.CLK(s_clk), .D(D), .EN(EN), .state(1), .moore(t1),.create(create),.Q(s1)); // triggered if create is 0 and FSM is in same password state
    reg_en reg2_s(.CLK(s_clk), .D(D), .EN(EN), .state(2), .moore(t1),.create(create),.Q(s2));
    reg_en reg3_s(.CLK(s_clk), .D(D), .EN(EN), .state(3), .moore(t1),.create(create),.Q(s3));

    comparison_checker pass_comparison(.en_pass1(p1),.set_pass1(s1), .en_pass2(p2),.set_pass2(s2),.en_pass3(p3),.set_pass3(s3), .out(t2)); // compares all equal state created and entered pass

    DEC display2 (.CLK(CLK), .Z(t2), .SEGMENTS(seg), .DISP_EN(an)); // Displays OPEN if all passwords sequences match and NOPE when they do not
Endmodule
```

FSM:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Cal Poly CPE 133
// Engineer: Jacqueline Radding
//
// Create Date: 08/19/2021 01:09:49 PM
// Design Name:
// Module Name: FSM_CHECKER
// Project Name: You Shall Not Pass Digital Lock Box
// Target Devices:
// Tool Versions:
// Description: FSM indicates the state of the password entered, 0-3. This triggers numbered registers to store the entered or created password(s).
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments: works with a seq divider.
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module FSM_CHECKER(
  input EN, // enter button
   input CLK, // clock from clock divider
   output logic [2:0]LEDS, // LED password state indication
   output logic moore  // password recording state
   );
 parameter [1:0] a = 2'd0;
    logic [1:0] b = 2'd1;
    logic [1:0] c = 2'd2;
    logic [1:0] d = 2'd3;
    logic [2:0] e = 3'd4;
   logic [2:0] f = 3'd5;
   logic [1:0] NS;
   logic [1:0] PS = a;
      //sequential logic to change states
   always_ff @ (posedge CLK) // reset add
   begin
       PS = NS;
   end

always_comb
   begin
   moore = 0;
   LEDS = 3'b001; //LEDs indicate password entering sequence one
   case (PS)//This is going to be the case where the button is pressed

      a: // first password entering state
      begin
      LEDS = 3'b001;
      moore = 0; //moore outputs depend only on state. Indicates the number of passwords recorded
to the code to trigger registers
        if (EN ==1) // if enable is pressed
        begin
             NS = b;
              end
         else // if no enable pressed, stay in the same state
        begin
             NS = a;
              end
       end

       b: // second password entering state
       begin
         LEDS = 3'b011; // LED indication of second password entering state
```

```
          moore = 1; // moore indicates the password recording state
          if (EN ==1)
          begin
              NS = c;
          end
           else
         begin
             NS = b;
              end
        end

        c: // third entering state
        begin
          moore = 2; // moore indicates the second password recorded state
          LEDS = 3'b111; // indicates third password entering sequence
          if (EN ==1)
          begin
              NS = d;
          end
          else
         begin
             NS = c;
              end
        end

         d: // finished entering state
        begin
          moore = 3; // moore indicates the third password recorded state
          LEDS = 3'b111; // LED indication of finished entering
          if (EN ==1) // entering to move to a state again
          begin
              NS = a;
          end
         else
         begin
             NS = d;
              end
        end
        default:
          NS = a;
      endcase
      end
Endmodule
`timescale 1ns / 1ps
```

Three Digit Password Comparator:

```
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Jacqueline Radding/Carter O'Neill
//
// Create Date: 08/18/2021 03:37:25 PM
// Design Name:
// Module Name: comparison_checker
// Project Name: You Shall Not Pass Digital Lock Box
// Target Devices: Basys 3
// Tool Versions:
// Description: compares six three bit created and entered passwords. Compares entered and set passwords
// of matching FSM states. Outputs a one when they match to the display module, zero if they don't
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module comparison_checker(
    //input CLK,
    input logic [2:0] set_pass1, // saved set password for first state from register
    input logic [2:0] set_pass2,// saved set password for second state
    input logic [2:0] set_pass3,
    input logic [2:0] en_pass1,// saved entered password for first state
    input logic [2:0] en_pass2,
    input logic [2:0] en_pass3,
    output logic out // outputs a 1 if all passwords match, zero if they don't
   );
   always_comb
  begin
   if ((set_pass1 == en_pass1) && (set_pass2 == en_pass2) && (set_pass3 == en_pass3)) // compares all state matching passwords
    begin
     out = 1; // if matching
    end
```

```
    else
    begin
    out = 0; // if not matching
    end
    end
endmodule
```

Set Password Register:

```
    `timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Cal Poly CPE 133
// Engineer: Jacqueline Radding
//
// Create Date: 08/23/2021 02:48:38 PM
// Design Name:
// Module Name: reg_set
// Project Name: You Shall Not Pass Digital Lock Box
// Target Devices: Basys3
// Tool Versions:
// Description: Memory Register with sync load behavior that saves the three bit password when
create mode is on and when state matches FSM pass state
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments: OPERATES WHEN CREATE IS ON  AND INPUT STATE MATCHES
FSM MOORE STATE
//
//////////////////////////////////////////////////////////////////////////////////



 module reg_set(
input [2:0] D, //switches
 input CLK,
 input state, // state trigger,  from states 1-3
 input moore, // moore state indication from FSM
 input create, // if create mode is on
 output logic [2:0]Q, // this is output 3 bit password
 input EN // enter button
```

```
 );

always_ff @ (posedge CLK) //Register_with_synch_load_behavior
begin
if ((create == 1)&&(EN == 1) && (moore == state)) // if password is in the matching sequence
state, record
 Q = D;
 end
endmodule
```

Enter Password Register:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Cal Poly CPE 133
// Engineer: Jacqueline Radding
//
// Create Date: 08/19/2021 05:27:14 PM
// Design Name:
// Module Name: reg_en
// Project Name: You Shall Not Pass Digital Lock Box
// Target Devices: Basys3
// Tool Versions:
// Description: Memory Register with sync load behavior that saves the three bit password when
create mode is off and when state matches FSM pass state
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments: OPERATES WHEN CREATE IS OFF  AND INPUT STATE MATCHES
FSM MOORE STATE
//
//////////////////////////////////////////////////////////////////////////////////


module reg_en(
input [2:0] D, //switches
 input CLK,
 input state, // state trigger, 1-3
 input moore, // moore state indication from FSM
 input create, // if create mode is on
```

```
 output logic [2:0]Q, // this is output 3 bit password
 input EN // enter button

 ); // created

always_ff @ (posedge CLK) //Register with sync load behavior
begin
if ((create == 0)&&(EN == 1) && (moore == state)) // if password is in the matching sequence
state, record
 Q = D; //saved 3 bit entered password
 end
Endmodule
```

Decoder Display with Open and Nope Output:

```
----------------------------------------------------------------------------------
-- Company: Ratner Engineering
-- Engineer: bryan mealy, edited by Carter O'Neill
--
-- Create Date:    15:27:40 12/20/2010
-- Design Name:
-- Module Name:    DEC
-- Project Name: You Shall Not Pass
-- Target Devices:
-- Tool versions:
-- Description: Special 7-segment display driver (4-letter words only), adapted
-- from Dr. Mealy for CPE133 Final Project
--
--  One Input:  Z
--
--      Z = '1': OPEn
--      Z = '0': nOPE
--
-- Dependencies:
--
-- 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;

----------------------------------------------------------------
-- Two word seven-segment display driver. Outputs are active
-- low and configured ABCEDFG in "segment" output.
----------------------------------------------------------------
entity DEC is
    Port (    CLK,Z : in std_logic;
           DISP_EN : out std_logic_vector(3 downto 0);
          SEGMENTS : out std_logic_vector(7 downto 0));
end DEC;

----------------------------------------------------------------
-- description of ssegment decoder
----------------------------------------------------------------
architecture DEC of DEC is

  component div
     Port (  clk : in std_logic;
             sclk : out std_logic);
  end component;

  -- intermediate signal declaration ----------------------
  signal   cnt_dig : std_logic_vector(1 downto 0);
  signal   digit   : std_logic_vector (3 downto 0);
  signal   sclk    : std_logic;

begin

  my_clk: div
  port map (clk => clk,
            sclk => sclk );

  -- advance the count (used for display multiplexing) -----
  process (SCLK)
  begin
    if (rising_edge(SCLK)) then
       cnt_dig <= cnt_dig + 1;
    end if;
  end process;
```

```
  -- select the display sseg data abcdefg (active low) -----
  segments <= "00000011" when digit = "0000"  else -- O
              "00110001" when digit = "0001"  else -- P
              "01100001" when digit = "0010"  else -- E
              "00010011" when digit = "0011"  else -- n
              "00010011" when digit = "0100"  else -- n
              "00000011" when digit = "0101"  else -- O
              "00110001" when digit = "0110"  else -- P
              "01100001" when digit = "0111"  else -- E
              "11111111";

  -- actuate the correct display --------------------------
  disp_en <= "1110" when cnt_dig = "00" else
             "1101" when cnt_dig = "01" else
             "1011" when cnt_dig = "10" else
             "0111" when cnt_dig = "11" else
             "1111";


  process (cnt_dig,Z)
    begin
    if (Z = '1') then
      case cnt_dig is
        when "00" => digit <= "0000"; -- cool
        when "01" => digit <= "0001";
        when "10" => digit <= "0010";
        when "11" => digit <= "0011";
        when others => digit <= "0000";
      end case;
    else
      case cnt_dig is
        when "00" => digit <= "0100"; -- bad
        when "01" => digit <= "0101";
        when "10" => digit <= "0110";
        when "11" => digit <= "0111";
        when others => digit <= "0000";
      end case;
    end if;
  end process;

end DEC;
```

Clock Divider:

```
----------------------------------------------------------------------------
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------------
-- Module to divide the clock
----------------------------------------------------------------------------
entity div is
    Port (  clk : in std_logic;
            sclk : out std_logic);
end div;

architecture my_clk_div of div is
   constant max_count : integer := (1100);
   signal tmp_clk : std_logic := '0';
begin
   my_div: process (clk,tmp_clk)
      variable div_cnt : integer := 0;
   begin
      if (rising_edge(clk)) then
         if (div_cnt = MAX_COUNT) then
            tmp_clk <= not tmp_clk;
            div_cnt := 0;
         else
            div_cnt := div_cnt + 1;
         end if;
      end if;
      sclk <= tmp_clk;
   end process my_div;
end my_clk_div;
```