ELEX 7660: Digital System Design

Project Report

{Remote Door Lock Control System}

Peiman Dadkhah, Julie Lee

Abstract

This document is a report of the digital system design of a modern FPGA-based remote door lock control system using Wi-Fi module and 4x4 keypad.

Table of Contents

Table of Figures

Table of Tables

Proposal

The project that we chose to work on is a modern FPGA-based Remote Door Lock Control system that manages the deadbolt. The door lock can be locked or unlocked through other electronic devices such as phone though Wi-Fi or another alternative to open is entering correct password through the keypad. There is one similar project which used a Bluetooth module to control servo motor/deadbolt. For information on the similar project, the link is provided: https://www.pantechsolutions.net/fpga-projects/bluetooth-based-wireless-home-automation-system-using-spartan3an-fpga-starter-kit.

The reason we chose this project is that when we need a house key to open a door lock, often we got into trouble because we lose it. and, if you want to give access to a friend to your house when you are not close by, it would be convenience when you are able to remotely unlock the door. In case your electronic device has a trouble, we made an alternative way to open a door: keypad. You are still able to open the door with a simple key. The dead bolt will not be held in place tightly with the gears.

Objective

1- With use of servo feedback connection we will identify if the door is locked or not and indicate it through led for the user.

2- From a door contact we will know if door is closed or not and show the user with a led.

3- By the end of project we will be able to open and close a dead bolt lock securely with 6 number passwords on pin pad.

4- Connection through cellphone to use the device through cellphone with the help of a Wi-Fi module.

5- Communication will be with master and slave design also know as Serial Peripheral Interface (SPI).

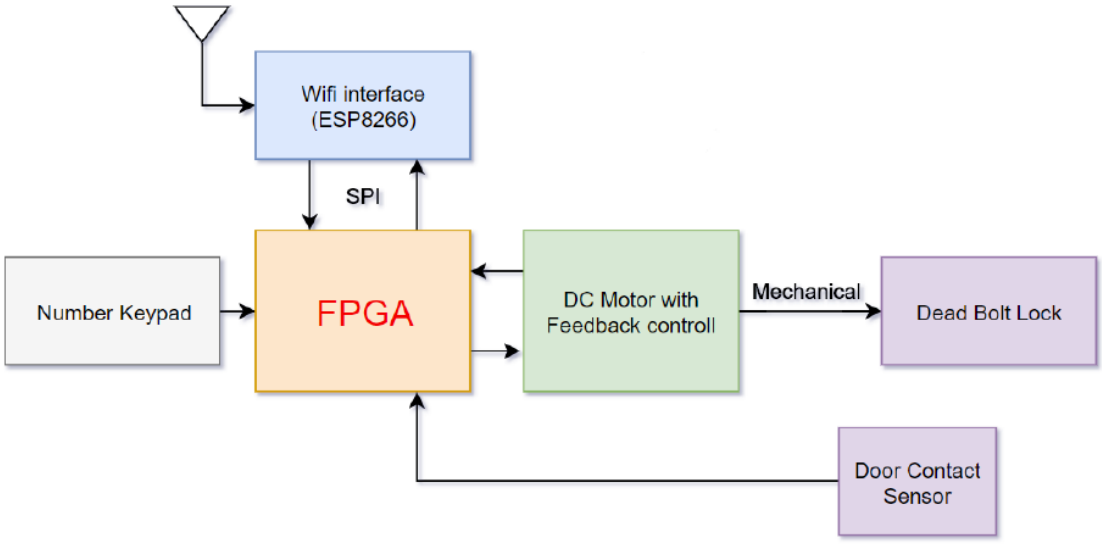


Figure 1: Overall Flow Chart of the Project

Project Plan

In this project we will use the FPGA as the central brain of this automated deadbolt lock. Figure 1 gives a good insight on how the system will be set up. The FPGA will be communicating with Wi-Fi module using SPI, User will know if door is open or close, if door is locked or unlocked. Through a cellphone app the user will be able to lock and unlock the door. However, there is also the option of using the keypad to unlock the door with a password.

The FPGA uses the feedback loop of the servo to know what position the lock in is, locked or not. In addition, the door contact sensor will allow the FPGA to know if door is closed or open. So, in case the door is open you will not be able to lock the door.

In figure 2 the flowchart shows how the software will be set up.

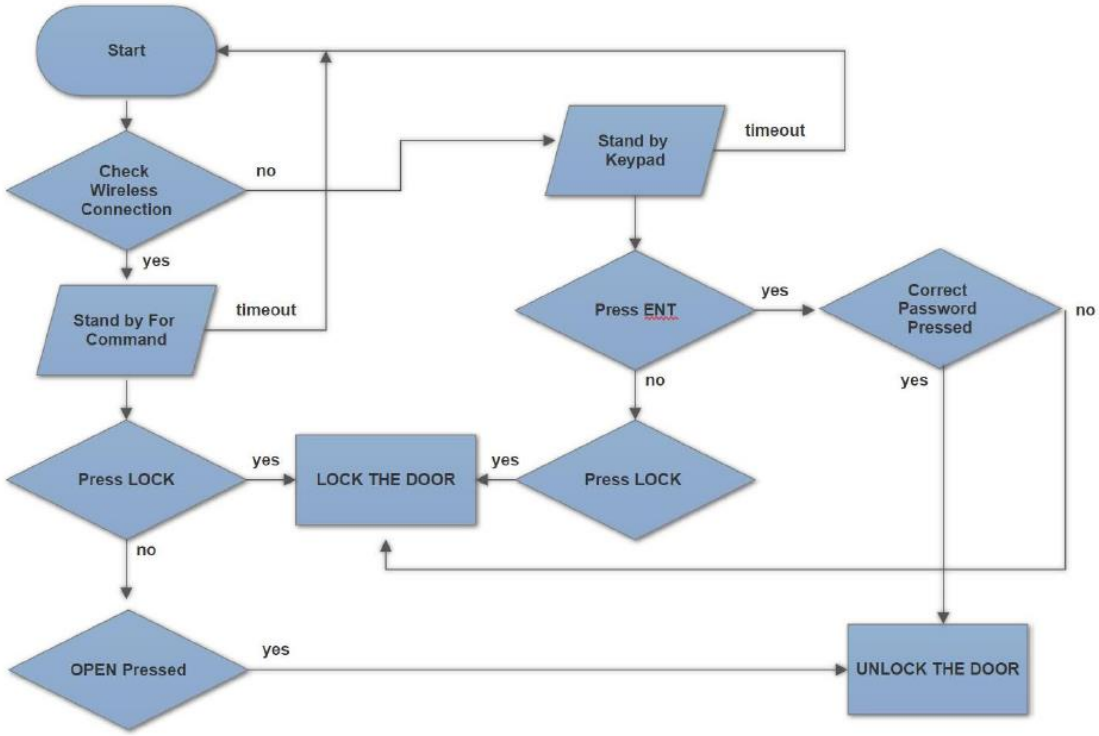


Figure 2: Flow Char of the Process

Parts List

Suggested Supplier					
Description	**Supplier**	**Part URL**	**Quantity**	**Unit Cost**	**Ext. Cost**
Self-contained Wi-Fi networking solution	Adafruit	https://www.adafruit.com/product/3046	1	$ 18.95	$ 18.95
Servomotor Micro Analog Plast	Digikey	https://www.digikey.ca/product-detail/en/adafruit-industries-llc/1449/1528-1089-ND/5154664	1	$ 13.27	$ 13.27
Stepper Motor Driver Breakout Board	Adafruit	https://www.adafruit.com/product/2448	1	$ 4.95	$ 4.95

Table 1: Parts Order List

Schedule

This is the project schedule, agreed by the group members. We hope to reach these milestones throughout the five-six weeks we have for the project.

Date	**Task**
Mar. 05th Week 01	Read the datasheets of the devices and test code each component
Mar. 12th Week 02	Continue to write a code for each component
Mar. 19th Week 03	Combine all the components and start building the project
Mar. 26th Week 04	Troubleshoot any errors and make any changes
April 2nd Week 05	Troubleshoot any errors and make any changes
April 9th Demonstration Week	Demonstrate the completed project to instructor and hand in our final project report

Table 2: Schedule of the Project

1 Background

1.1 4x4 Matrix Keypad

A switch matrix contains push button at each intersection of rows and column. In this project the keypad is implemented decimal values from 0 to 9 which allows user to enter a password and send it to FPGA by pressing the enter. The signals from the keypad are named as shown in below figure:

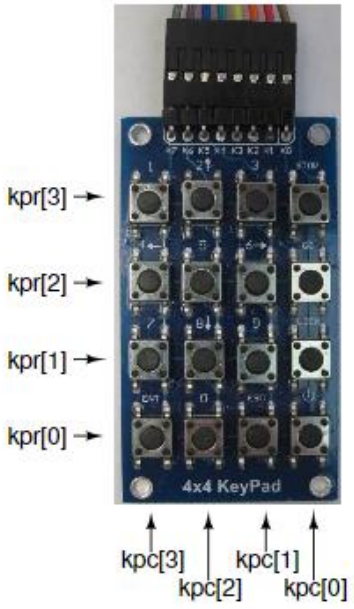


Figure 3: 4x4 Matrix Keypad

1.2 RC Servo

Radio Control (RC) Servos are controlled by sending the pulse of variable width. The angle is determined by the duration of a pulse (PWM). The servo rotates to a position 0° when the pulse width is 1ms, and maximum of 2ms will turn servo angle to 180° conterclockwise from the neutral point.

Batan S1123 Servo is an analog feedback micro servo that returns the position of the RC servo motor though analog input.

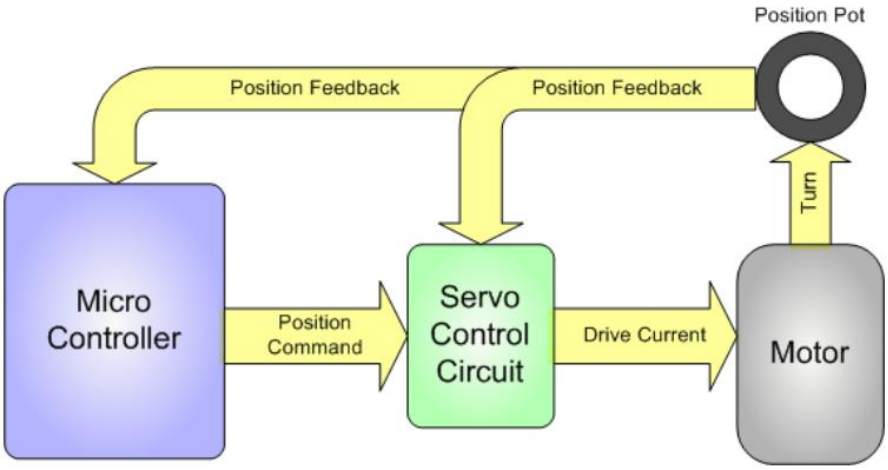
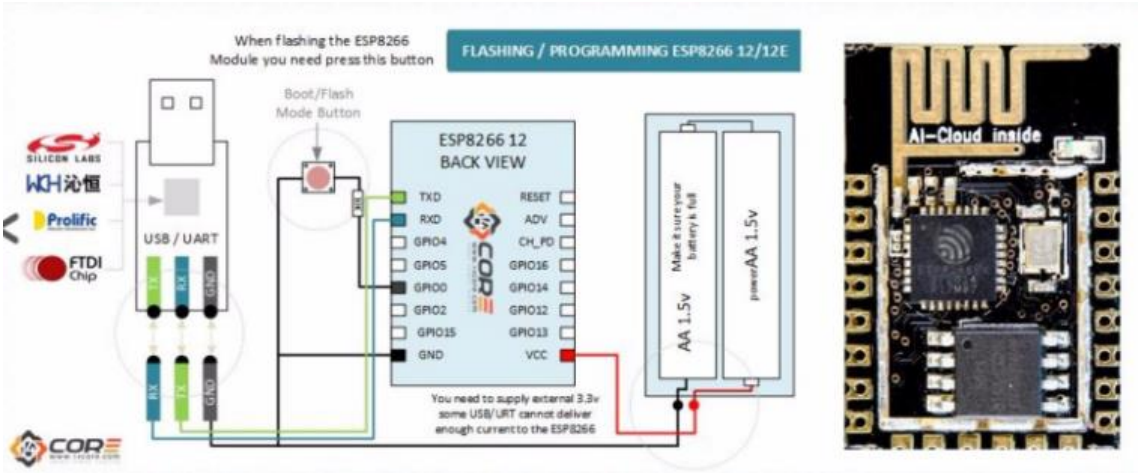


Figure 4: "Close Loop" Feedback System

1.3 ESP8266MOD Wi-Fi

The ESP8266 Wi-Fi module contains SOC with integrated TCP/IP protocol that can access the Wi-Fi network easily. With the circuits connected like below, we are able to program the module with the Arduino.



2 Module Design

2.1 Project.sv

This module is the top-level entity, main module for this project. Following are the inputs and outputs used in this project and their brief description.

Inputs	Description
kpr	Active low with pull up of the 4x4 keypad
Reset_n	Initializes entire program
CLOCK_50	Internal clock from the FPGA, 50MHz
wifi	Digital signal from the ESP8266 MOD
Outputs	**Description**
Lock_state	Digital signal indicating the state of the rotor
codehit	Digital signal connected to the LED to indicate whether the 6-digit long password is pressed
kphit	Digital signal connected to the LED to indicate whether the key it pressed from the keypad
servo	Pwm controlled signal to control position of the rotor
clk	2kHz clock created from 50MHz

Table 3: I/O used in the project

2.2 Keypad Modules

Similar to code from lab2, the keypad modules were kpdecode.sv and colseq.sv. In the kpdecode module, we used a 400Hz clock to solve a problem with debouncing.

2.3 lockpass.sv and Passcheck.sv

The lockpass module saves each digit inputted from the 4x4 keypad into a code array. Based on the 6 digits inputted from the keypad, the module compares entered passcode with the six different passcodes and if there is a match, it changes the state of the lock.

2.4 Pwmcontrol.sv

This module changes the pulse width depending on the state of the lock. If the signal is to open the bolt, it outputs 1ms of a high pulse to change to position of the rotor to 0°. If the signal is to lock the bolt, it outputs 2ms of a high pulse to rotate the rotor to 180°. The width of the pulse is controlled by dividing 2kH clock in to 40 and controlled by using the counter.

2.5 Phone Control

Blynk is a Platform with iOS and Android apps to control microcontrollers over the internet. We chose to use this app to make a simple Platform to lock and unlock the door. It sends out the signal to FPGA through the LOCKED switch, and receives analog data from the feedback servo to indicate to position of the servo to show user the state of the rotor.

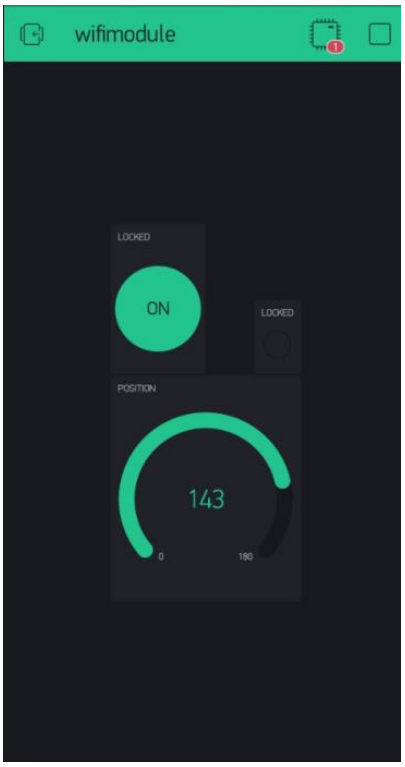


Figure 5: Phone interface of Controlling the Lock.

3 Conclusion

The project successfully completed with some minor issue. Debouncing of the keypad needed more work than we thought moreover, it was difficult to have timing out function programed with the System Verilog. The project required more research, but we gained valuable knowledge on digital hardware systems. Due to the limited knowledge of FGPA and System Verilog, we chose to program ESP8266. For the future work, it would be better to implement a system where the home owner can know who accessed the home and the state of it easily.

4 References

[1] E. Inc., "ESP8266EX Datasheet," 2018. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Accessed 02 April 2018].

[2] B. Earl, "Analog Feedback Servos," Adafruit, 21 Jan. 2018. [Online]. Available: https://learn.adafruit.com/analog-feedback-servos/servos-as-input-devices. [Accessed 01 April 2018].

[3] "Bluetooth Based Wireless Home Automation System Using Spartan3an FPGA Starter Kit," Pantech Solutions, 2017. [Online]. Available: https://www.pantechsolutions.net/fpga-projects/bluetooth-based-wireless-home-automation-system-using-spartan3an-fpga-starter-kit. [Accessed 02 April 2018].

[4] T. Technologies, "Terasic DE0-Nano User Manual," Terasic Technologies, 2012. [Online]. Available: http://mathcs.holycross.edu/~kwalsh/zebra/DE0_Nano_User_Manual_v1.9.pdf. [Accessed 2018].

Appendices

The appendices include only System Verilog code used in this project.

A. Top-Level Module

A.1 Project.sv

```
/*////////////////////////////////////////////////////////
Project.sv This code is a modification of lab 2 to work
as the main module and was modified by Peiman and Julie.
The code calls all the modules and connects them to pins
on the FPGA.

Author: Peiman Dadkhah, Julie Lee          Date: 2018-March-09
*////////////////////////////////////////////////////////]

module Project ( output logic [3:0] kpc, num, // column select, active-low
      output logic lock_state, codehit, kphit, servo, clk,
   (* altera_attribute = "-name WEAK_PULL_UP_RESISTOR ON" *)
      input logic  [3:0] kpr,  // rows, active-low w/ pull-ups
      //output logic [7:0] leds, // active-low LED segments for testing
      input logic  reset_n, CLOCK_50, wifi,
      output logic [3:0] ct) ;

      logic sclk ;         // 400Hz clock for keypad scanning
      logic [3:0] counter ;        // value of pressed key
      logic [5:0] [3:0] code;

      assign clock50 = CLOCK_50 ;


      assign ct = {{2{1'b0}}, 2'b11};


    pll pll0 ( .inclk0(CLOCK_50), .c0(clk) );

      kpdecode    kpdecode_0 ( .kpc, .kpr, .sclk, .num, .kphit );
      myclocks    myclocks_0 (.clock50(clk), .reset_n, .sclk);
      colseq            colseq_0(.reset_n, .clk, .kpr, .kpc);
      lockpass    lockpass_0 (.reset_n, .clk, .kphit, .sclk, .counter, .codehit,
.num, .code);
      passcheck   passcheck_0 (.lock_state, .reset_n, .clk, .codehit, .code);
      pwmcontrol  pwmcontrol_0 ( .state(lock_state), .clk, .reset_n, .servo,
.wifi) ;

      //decode7 decode7_0(.counter, .leds); //Used to test the inputs

endmodule


module myclocks
      #(N = 200)
      (input logic clock50, reset_n, output logic sclk);

      logic [19:0] n, n_next;
      logic sclk_next;
      always_comb begin
            n_next = n ;
```

```
            sclk_next = sclk ;
            if (!reset_n) begin
                  n_next = N - 1 ;
            end
            else begin
                  // sclk 400Hz
                  if ( n )
                        n_next = n - 1 ;
                  else
                        n_next = N - 1 ;

                  if ( n >= N - N/2 )
                        sclk_next = 0 ;
                  else
                        sclk_next = 1 ;
            end
      end
      always_ff @ (posedge clock50 ) begin
            n <= n_next;
            sclk <= sclk_next;
      end
endmodule

module pll ( inclk0, c0);

        input     inclk0;
        output    c0;

        wire [0:0] sub_wire2 = 1'h0;
        wire [4:0] sub_wire3;
        wire  sub_wire0 = inclk0;
        wire [1:0] sub_wire1 = {sub_wire2, sub_wire0};
        wire [0:0] sub_wire4 = sub_wire3[0:0];
        wire  c0 = sub_wire4;

        altpll altpll_component ( .inclk (sub_wire1), .clk
          (sub_wire3), .activeclock (), .areset (1'b0), .clkbad
          (), .clkena ({6{1'b1}}), .clkloss (), .clkswitch
          (1'b0), .configupdate (1'b0), .enable0 (), .enable1 (),
          .extclk (), .extclkena ({4{1'b1}}), .fbin (1'b1),
          .fbmimicbidir (), .fbout (), .fref (), .icdrclk (),
          .locked (), .pfdena (1'b1), .phasecounterselect
          ({4{1'b1}}), .phasedone (), .phasestep (1'b1),
          .phaseupdown (1'b1), .pllena (1'b1), .scanaclr (1'b0),
          .scanclk (1'b0), .scanclkena (1'b1), .scandata (1'b0),
          .scandataout (), .scandone (), .scanread (1'b0),
          .scanwrite (1'b0), .sclkout0 (), .sclkout1 (),
          .vcooverrange (), .vcounderrange ());

        defparam
                altpll_component.bandwidth_type = "AUTO",
                altpll_component.clk0_divide_by = 25000,
                altpll_component.clk0_duty_cycle = 50,
                altpll_component.clk0_multiply_by = 1,
                altpll_component.clk0_phase_shift = "0",
                altpll_component.compensate_clock = "CLK0",
                altpll_component.inclk0_input_frequency = 20000,
                altpll_component.intended_device_family = "Cyclone IV E",
                altpll_component.lpm_hint = "CBX_MODULE_PREFIX=lab1clk",
                altpll_component.lpm_type = "altpll",
```

```
                altpll_component.operation_mode = "NORMAL",
                altpll_component.pll_type = "AUTO",
                altpll_component.port_activeclock = "PORT_UNUSED",
                altpll_component.port_areset = "PORT_UNUSED",
                altpll_component.port_clkbad0 = "PORT_UNUSED",
                altpll_component.port_clkbad1 = "PORT_UNUSED",
                altpll_component.port_clkloss = "PORT_UNUSED",
                altpll_component.port_clkswitch = "PORT_UNUSED",
                altpll_component.port_configupdate = "PORT_UNUSED",
                altpll_component.port_fbin = "PORT_UNUSED",
                altpll_component.port_inclk0 = "PORT_USED",
                altpll_component.port_inclk1 = "PORT_UNUSED",
                altpll_component.port_locked = "PORT_UNUSED",
                altpll_component.port_pfdena = "PORT_UNUSED",
                altpll_component.port_phasecounterselect = "PORT_UNUSED",
                altpll_component.port_phasedone = "PORT_UNUSED",
                altpll_component.port_phasestep = "PORT_UNUSED",
                altpll_component.port_phaseupdown = "PORT_UNUSED",
                altpll_component.port_pllena = "PORT_UNUSED",
                altpll_component.port_scanaclr = "PORT_UNUSED",
                altpll_component.port_scanclk = "PORT_UNUSED",
                altpll_component.port_scanclkena = "PORT_UNUSED",
                altpll_component.port_scandata = "PORT_UNUSED",
                altpll_component.port_scandataout = "PORT_UNUSED",
                altpll_component.port_scandone = "PORT_UNUSED",
                altpll_component.port_scanread = "PORT_UNUSED",
                altpll_component.port_scanwrite = "PORT_UNUSED",
                altpll_component.port_clk0 = "PORT_USED",
                altpll_component.port_clk1 = "PORT_UNUSED",
                altpll_component.port_clk2 = "PORT_UNUSED",
                altpll_component.port_clk3 = "PORT_UNUSED",
                altpll_component.port_clk4 = "PORT_UNUSED",
                altpll_component.port_clk5 = "PORT_UNUSED",
                altpll_component.port_clkena0 = "PORT_UNUSED",
                altpll_component.port_clkena1 = "PORT_UNUSED",
                altpll_component.port_clkena2 = "PORT_UNUSED",
                altpll_component.port_clkena3 = "PORT_UNUSED",
                altpll_component.port_clkena4 = "PORT_UNUSED",
                altpll_component.port_clkena5 = "PORT_UNUSED",
                altpll_component.port_extclk0 = "PORT_UNUSED",
                altpll_component.port_extclk1 = "PORT_UNUSED",
                altpll_component.port_extclk2 = "PORT_UNUSED",
                altpll_component.port_extclk3 = "PORT_UNUSED",
                altpll_component.width_clock = 5;
endmodule
```

B. Keypad Modules

B.1 Kpdecode.sv

```
clear
close all
clc

p = bodeoptions('cstprefs');
p.Xlim = [1, 1e6];
```

```
p.Ylim = [-80, 10];
p.PhaseVisible = 'off';
t = 0:0.001:0.1;

[B, A] = butter(4, 1000, 'low', 's');
[B1, A1] = cheby1(4, 5, 1000, 'low', 's');
[B2, A2] = cheby2(4, 50, 1000, 'low', 's');
[B3, A3] = ellip(4, 5, 50, 1000, 'low', 's');
[Bt, At] = besself(4, 1);
[B4, A4] = lp2lp(Bt, At, 1000);

figure(1);
bodeplot(tf(B,A), tf(B1,A1),tf(B2,A2),tf(B3,A3),tf(B4,A4),p);
legend('Butter', 'cheby1', 'cheby2', 'ellip', 'bessel');
figure(2);
stepplot(tf(B,A), tf(B1,A1),tf(B2,A2),tf(B3,A3),tf(B4,A4));
legend('Butter', 'cheby1', 'cheby2', 'ellip', 'bessel');
```

B.2 Colseq.sv

```
/*////////////////////////////////////////////////
keypad      signal      color conn. GPIO  FPGA
k0          kpr[3]      brown       14          09          D5
k1          kpr[2]      red   16          011         A6
k2          kpr[1]      orange      18          013   D6
k3          kpr[0]      yellow      20          015   C6
k4          kpc[0]      green       22          017   E6
k5          kpc[1]      blue  24          019   D8
k6          kpc[2]      violet      26          021   F8
k7          kpc[3]      gray  28          023   E9
These FPGA pins are defined in the supplied
lab2pins.qsf file as signals named kpr[3] to kpr[0]
(rows, top to bottom) and kpc[3] to 0 (columns, left to right)
Author: Peiman Dadkhah                  Date:2018-1-21
*//////////////////////////////////////////////////
module colseq ( input logic reset_n, input logic clk,
             input logic [3:0] kpr,
             output logic [3:0] kpc);
      logic [3:0] kpc_next;

      always_comb
      begin

            if (!reset_n)
                  kpc_next = 4'b0111;

            else
            begin
                  if (kpr == 4'b1111)
                  begin
                        unique case (kpc)
                              4'b0111: kpc_next = 4'b1011;
                              4'b1011: kpc_next = 4'b1101;
                              4'b1101: kpc_next = 4'b1110;
                              4'b1110: kpc_next = 4'b0111;
                              default: kpc_next = 4'b0111;
                        endcase
                  end
```

```
                  else if ( kpr == 4'b0111 || kpr == 4'b1011 ||
                              kpr == 4'b1101 || kpr == 4'b1110 )
                        kpc_next = kpc;

                  else
                        kpc_next = 4'b0111;
            end

      end
      always_ff @(posedge clk)
      begin
            kpc <= kpc_next;
      end
endmodule
```

B.3 Lockpass.sv

```
module lockpass(input logic reset_n, clk, kphit, sclk,
            output logic [3:0] counter, output logic codehit,
            input logic [3:0] num, output logic [5:0] [3:0]code);

      logic [3:0] counter_next;

      always_ff @ (posedge kphit) begin
            counter_next = counter;
            code [counter] = num;

            if (counter < 6) begin
                  counter_next = counter_next +1;
                  codehit = 0;
            end
            else if (counter == 6) begin
                  codehit =1;
                  counter_next = 0;
            end
      end
      always_ff @ (posedge clk ) begin
            counter <= counter_next ;
      end
endmodule
```

B.4 Passcheck.sv

```
module passcheck (output logic lock_state,
            input logic reset_n, clk, codehit,
            input logic [5:0][3:0]code);

      logic [3:0] index_one, index_two;
      logic [5:0][3:0]passcodes;
      logic [3:0]code_match;
      logic [3:0] i;
      // initialize all the values

      initial begin
            // zero is equal to locked and 1 means lock open
            //passcodes = {4'd4, 4'd4, 4'd4, 4'd4, 4'd4, 4'd4};
```

```
            passcodes[0] = {4'd1,4'd1,4'd1,4'd2,4'd2,4'd2};
            passcodes[1] = {4'h0,4'd0,4'd0,4'd9,4'd9,4'd9};
            passcodes[2] = {4'd9,4'd0,4'd0,4'd9,4'd2,4'hb};
            passcodes[3] = {4'd1,4'd2,4'd3,4'd1,4'd2,4'd3};
            passcodes[4] = {4'd4,4'd4,4'd4,4'd4,4'd4,4'd4};
            passcodes[5] = {4'd1,4'd1,4'd1,4'd1,4'd1,4'd1};
      end

  always_ff @ (posedge codehit) begin
    // if reset is asserted initial everythig
    if (!reset_n) begin
     lock_state = 0;
     // if code is entered check the validity
    end
       else begin
       for (i = 0; i <6; i= i+1) begin
                  if (passcodes[i] == code)
                        lock_state = ~lock_state;
            end
      end

  end

endmodule
```

C. PWM Control

C.1 Pwmcontrol.sv

```
module pwmcontrol( input logic state, clk, reset_n,
              output logic servo) ;

      logic [6:0] count, count_next;
      logic [6:0] max_count = 40;
      logic current_state;
      logic servo_next;


      always_comb begin
            count_next = count;
            current_state = state;
            servo_next = servo;

            if (!reset_n) begin
                  count_next = 0;
                  current_state = state;
                  servo_next = servo;
            end

            if(state) begin //Open
                  if( count <= 2 )
                        servo_next = 1;
                  else
                        servo_next = 0;
            end
```

```
            else if(!state) begin //Locked
                  if( count <= 4 )
                        servo_next = 1;
                  else
                        servo_next = 0;
            end
            if (clk)
                  count_next = count + 1;

            if (count == max_count)
                  count_next = 0;
      end

      always_ff@ (posedge clk) begin
                  servo <= servo_next;
                  count <= count_next ;
            end
endmodule
```

D. Pinouts

D.1 Projectpins.qsf

```
set_location_assignment PIN_R8 -to CLOCK_50
set_location_assignment PIN_T9 -to wifi[0]
set_location_assignment PIN_N12 -to servo[12]
set_location_assignment PIN_A15 -to lock_state
set_location_assignment PIN_A13 -to codehit
set_location_assignment PIN_B13 -to kphit

set_location_assignment PIN_A12 -to ct[0]
set_location_assignment PIN_A5 -to leds[0]
set_location_assignment PIN_C11 -to ct[1]
set_location_assignment PIN_B6 -to leds[1]
set_location_assignment PIN_E11 -to ct[2]
set_location_assignment PIN_B7 -to leds[2]
set_location_assignment PIN_C9 -to ct[3]
set_location_assignment PIN_A7 -to leds[3]
set_location_assignment PIN_C8 -to leds[4]
set_location_assignment PIN_E7 -to leds[5]
set_location_assignment PIN_E8 -to leds[6]
set_location_assignment PIN_F9 -to leds[7]

set_location_assignment PIN_D5 -to kpr[3]
set_location_assignment PIN_A6 -to kpr[2]
set_location_assignment PIN_D6 -to kpr[1]
set_location_assignment PIN_C6 -to kpr[0]
set_location_assignment PIN_E6 -to kpc[0]
set_location_assignment PIN_D8 -to kpc[1]
set_location_assignment PIN_E9 -to kpc[3]
set_location_assignment PIN_F8 -to kpc[2]
# set_location_assignment PIN_J15 -to KEY[0]

set_location_assignment PIN_J15 -to reset_n
```