ELEX 7660: Digital System Design

Report for Project

Door Security Lock

Prepared by:	Prep Date
Wenjian Chen	Mar 09, 2018
Vasiliy Baryshnikov	Mar 09, 2018

Table of Contents

Introduction 4
1. Background, Modelling and Preliminary Analysis 4
1.1 FPGA 4
1.2 Proximity Sensor 5
1.3 Solenoid 5
1.4 4X4 Matrix Keypad 6
1.5 7 - segment LED Display 6
1.6 Optocoupler 7
1.7 Metal Oxide Semiconductor Field-Effect Transistor 7
1.10 Modelling 9
1.11 Preliminary Analysis 9
2. Equipment 10
3. Procedure 10
4. Results 10
4.1 Block Diagram and Schematic 10
4.2 Real Circuit 11
5. Discussion 12
6. Summary and Conclusions 12
7. References 12
Appendices 13
Datasheet of Optocoupler 4N35 and Power MOSFET IRF540 13
Program listing 13

Table of Figures

Figure 1 - Field Programmable Gate Array of 3.3V DC voltage 4
Figure 2 - Proximity Sensor 5
Figure 3 - Solenoid Pull 12V DC 5
Figure 4 - 4X4 Matrix Keypad 6
Figure 5 - 7 - Segment LED Display 6
Figure 6 - Optocoupler Pinout Figure 7 – Optocoupler 7
Figure 8 - MOSFET IRF540 Figure 9 - MOSFET IRF540 Pinout 7
Figure 10 - Speaker 8
Figure 11 - Digital Multimeter 8
Figure 12 - Schematic of Door Security Lock 9

Figure 13 - Block Diagram of Door Security Lock.. 10
Figure 14 - Schematic of Door Security Lock 11
Figure 15 - Real Circuit of Door Security Lock.. 11
Figure 16 - Datasheet of Optocoupler 4N35 13
Figure 17 - Datasheet of Power MOSFET IRF540 13

Introduction

In this project, the objective is to design, build and test door lock, as security system has become a very important technology in our lives and worldwide. It is always necessary for people to put their personal important stuff/information in a place/system with high and reliable security. For high assurance, systems will logically have security requirement, availability, reliability and robustness requirements, so this kind of proper security controls is used to prevent computer hacker attacks or stealing. A 12V DC solenoid acts as the lock and performs its operation based on the signal received from FPGA, which is an integrated circuit with high speed interfaces and serial communication protocals. The lock comes with a fixed password and could be locked immediately or a certain amount of time after unlocked. Also, when it is left open for a specified amount of time, the alarm will sound to remind people of closing it after leaving. The alarm will go off itself as soon as the door is closed.

1. Background, Modelling and Preliminary Analysis

Since we are designing a security lock made up of FPGA, proximity sensor, solenoid, optocoupler, MOFSET, 4X4 keypad, 7-segment LED display, and speaker, it is quite important to be aware of the power, voltage and current ratings of every component used to prevent damaging the parts, especially expensive components like FPGA. Also, to avoid buying wrong components, it is always good to check the compatibility among these components.

1.1 FPGA

Field Programmable Gate Array is the core component of this project. It can be configured by the designer using Hardware Description Language. FPGA contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be “wire together”, like many logic gates that can be inter-wired in different configurations [1].

Figure 1 - Field Programmable Gate Array of 3.3V DC voltage

1.2 Proximity Sensor

A proximity sensor is a sensor able to detect the presence of nearby objects without any physical contact [2]. With this proximity sensor, we can remind people of closing the door by emitting a sound.

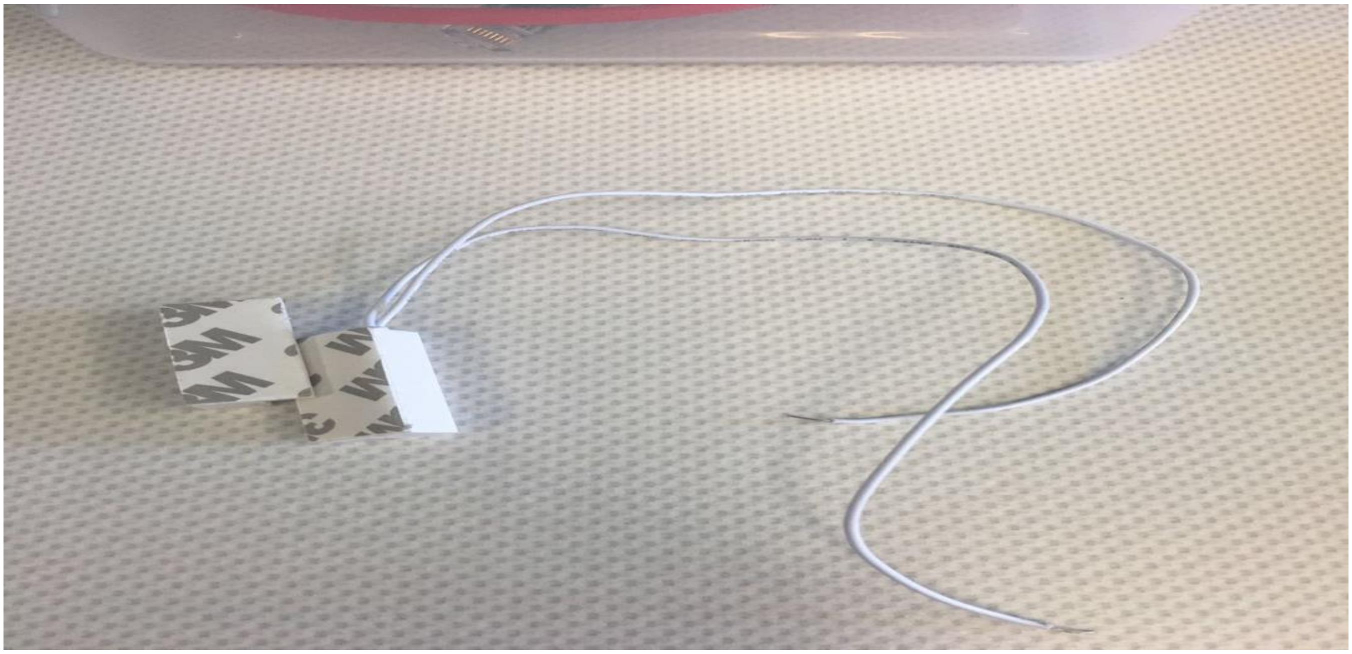


Figure 2 - Proximity Sensor

1.3 Solenoid

Basically solenoids are just electromagnets made of a coil of copper wire. By acquiring a signal from FPGA, the middle slug will be pulled out, hence acting as a security lock. The solenoid we use is 12V DC.

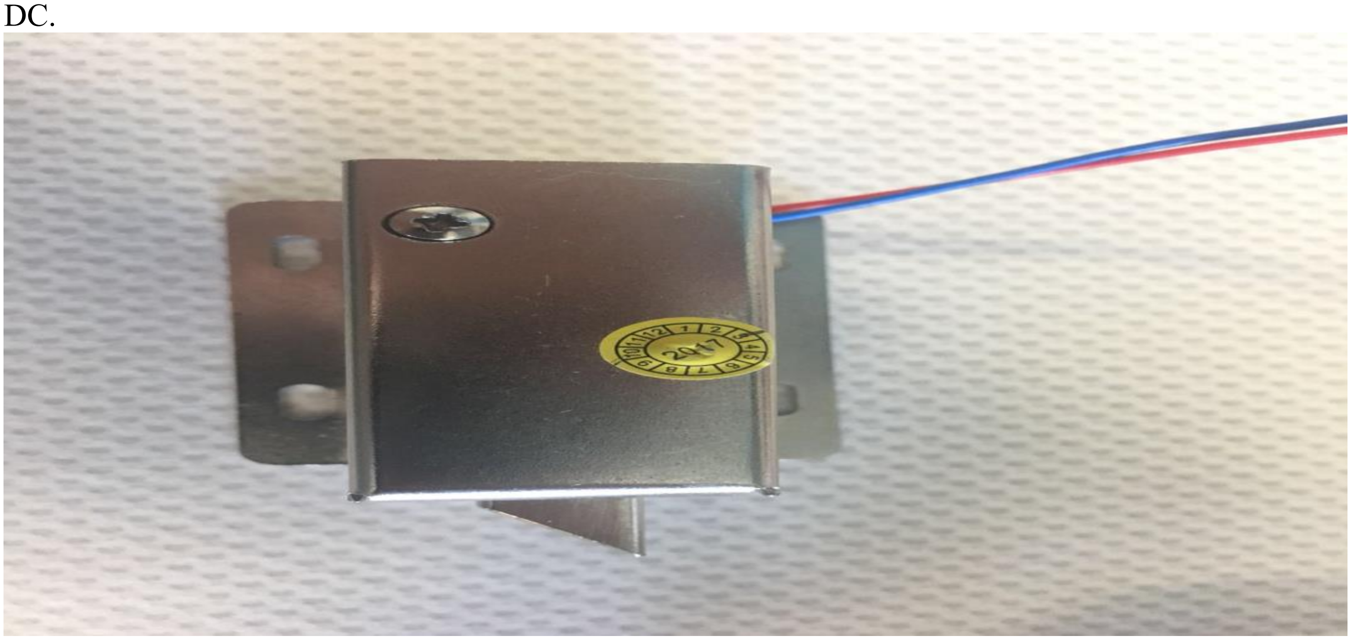


Figure 3 - Solenoid Pull 12V DC

1.4 4X4 Matrix Keypad

The keypad is interfaced to FPGA, will be programmed to output digit from 0 to 9, and also has the unlock and lock functions.

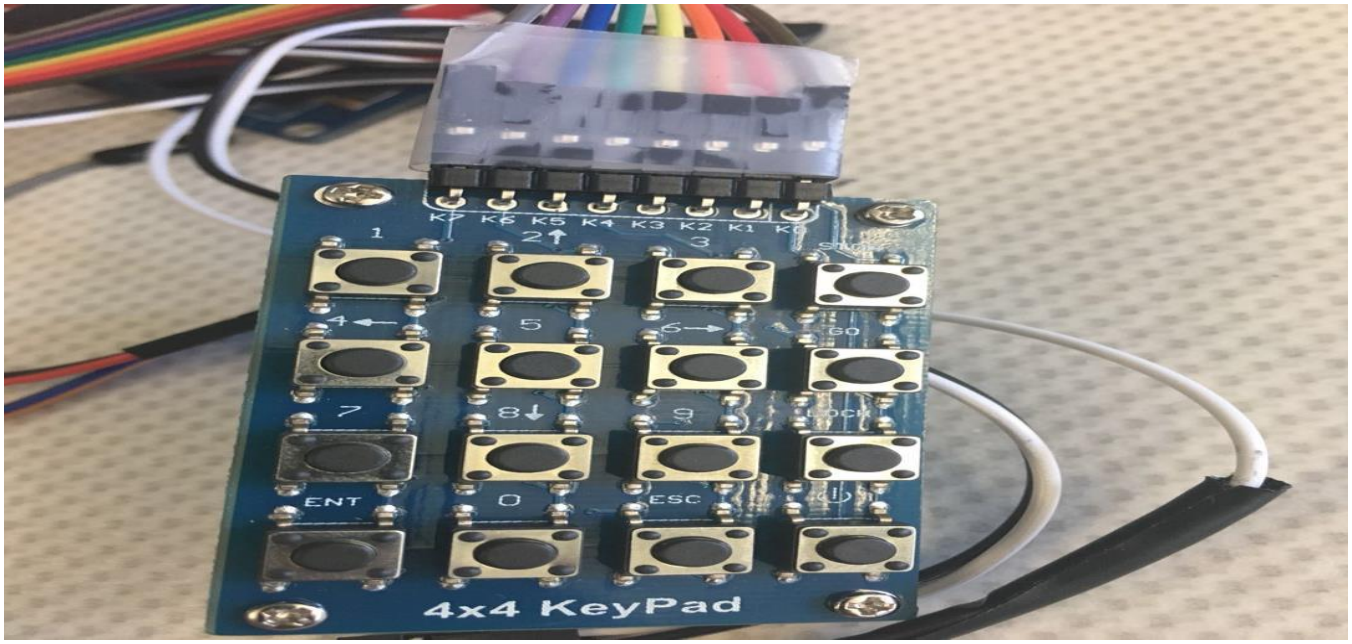


Figure 4 - 4X4 Matrix Keypad

1.5 7 - segment LED Display

This kind of display has been widely used in our lives, such as digital clocks, electronic meters and basic calculators. We programmed this to display the digits we enter so that we can clearly see what we enter.

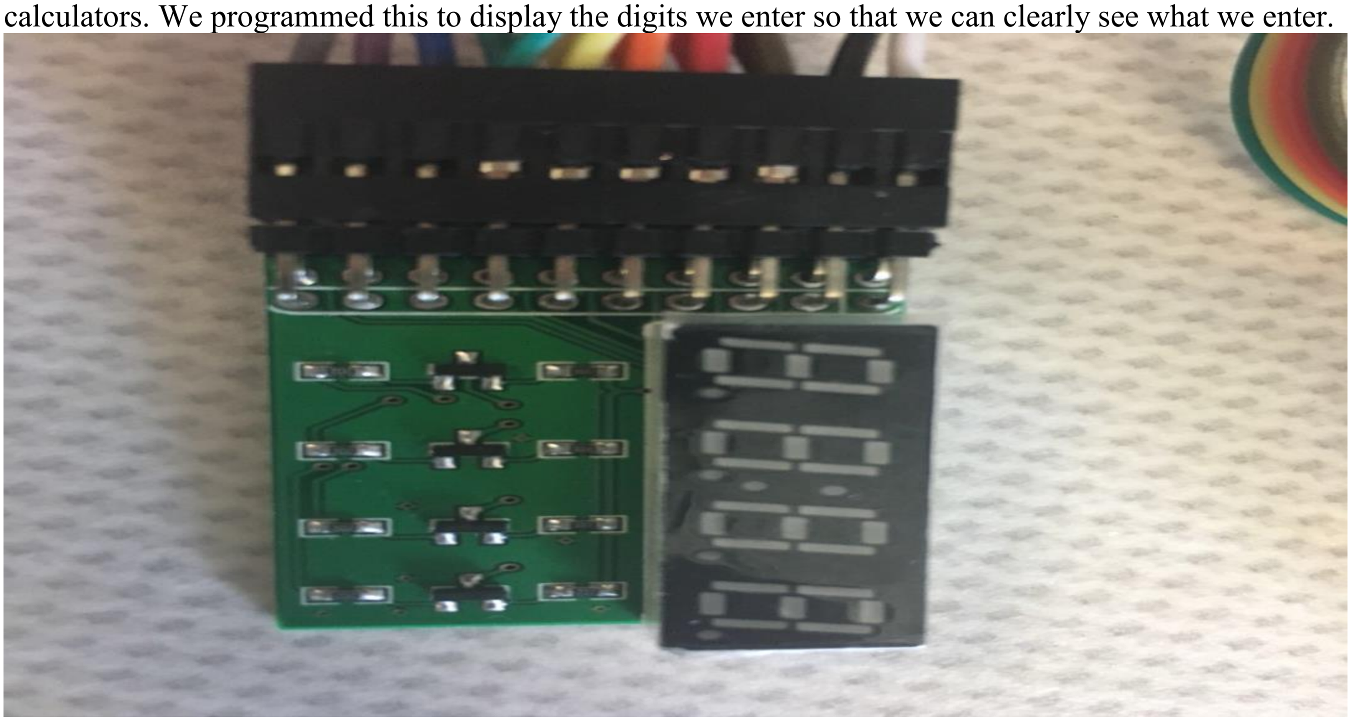
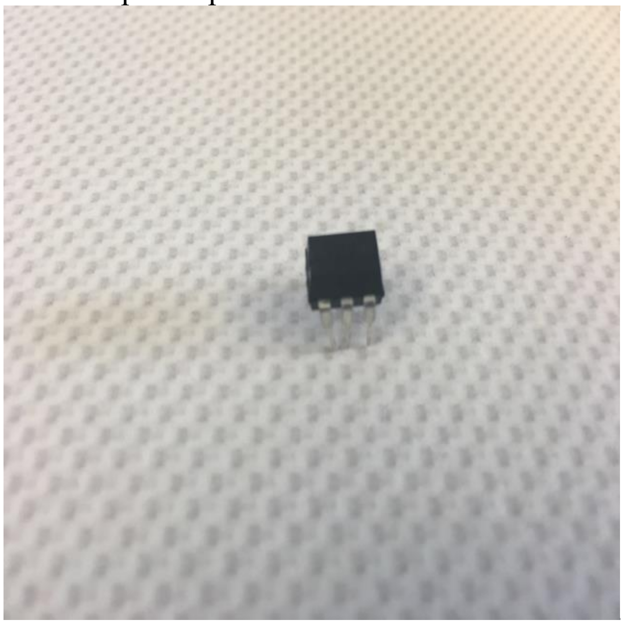


Figure 5 - 7 - Segment LED Display

1.6 Optocoupler

This is Optocoupler 4N35 used to transfer electrical signals between FPGA and solenoid.



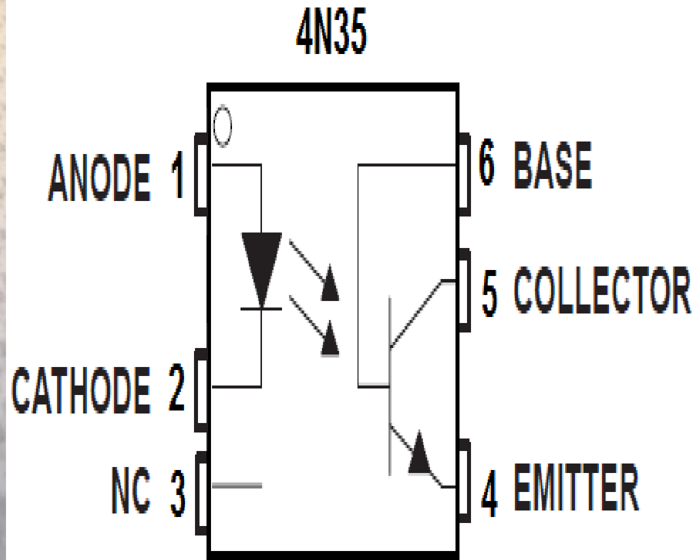
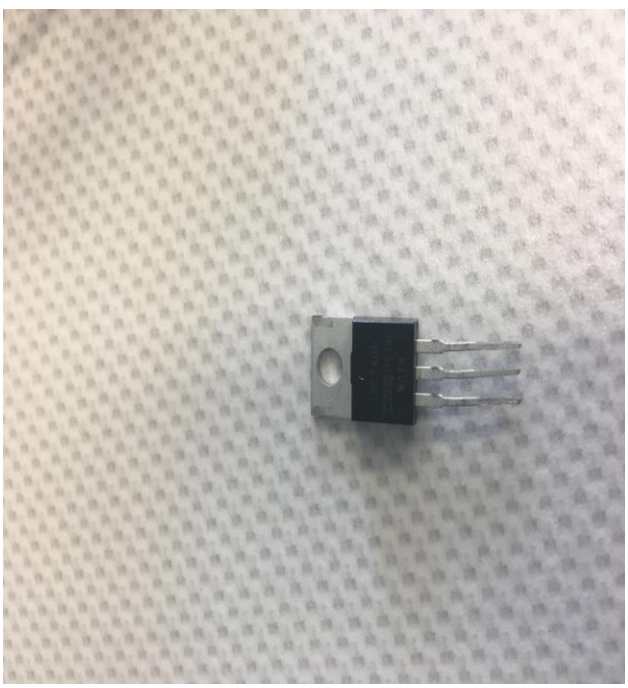


Figure 6 - Optocoupler Pinout   *Figure 7 – Optocoupler*

1.7 Metal Oxide Semiconductor Field-Effect Transistor

In the project, we use this to adjust the voltage between FPGA and solenoid so they can be compatible.



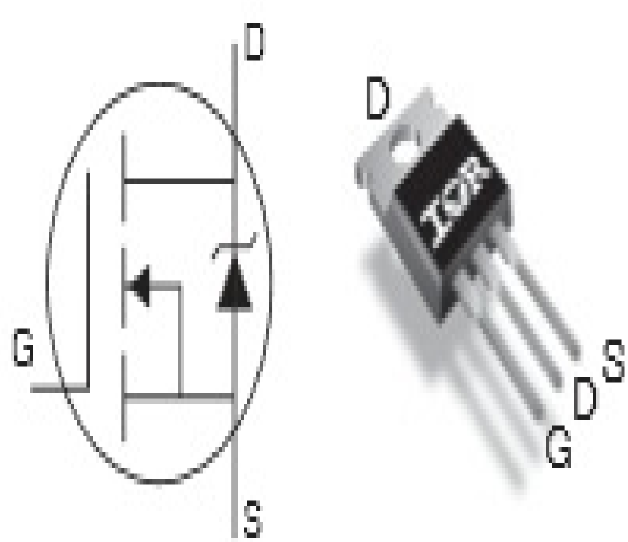


Figure 8 - MOSFET IRF540   *Figure 9 - MOSFET IRF540 Pinout*

1.8 Speaker

This is used to produce a sound when the door is being opened for a relatively long time

Figure 10 - Speaker

1.9 Digital Multimeter

This is used to troubleshoot the circuit when the circuit didn't work after we built it

Figure 11 - Digital Multimeter

1.10 Modelling

In this section, we will analyze FPGA and optocoupler in order to select the correct values for components, i.e. resistors

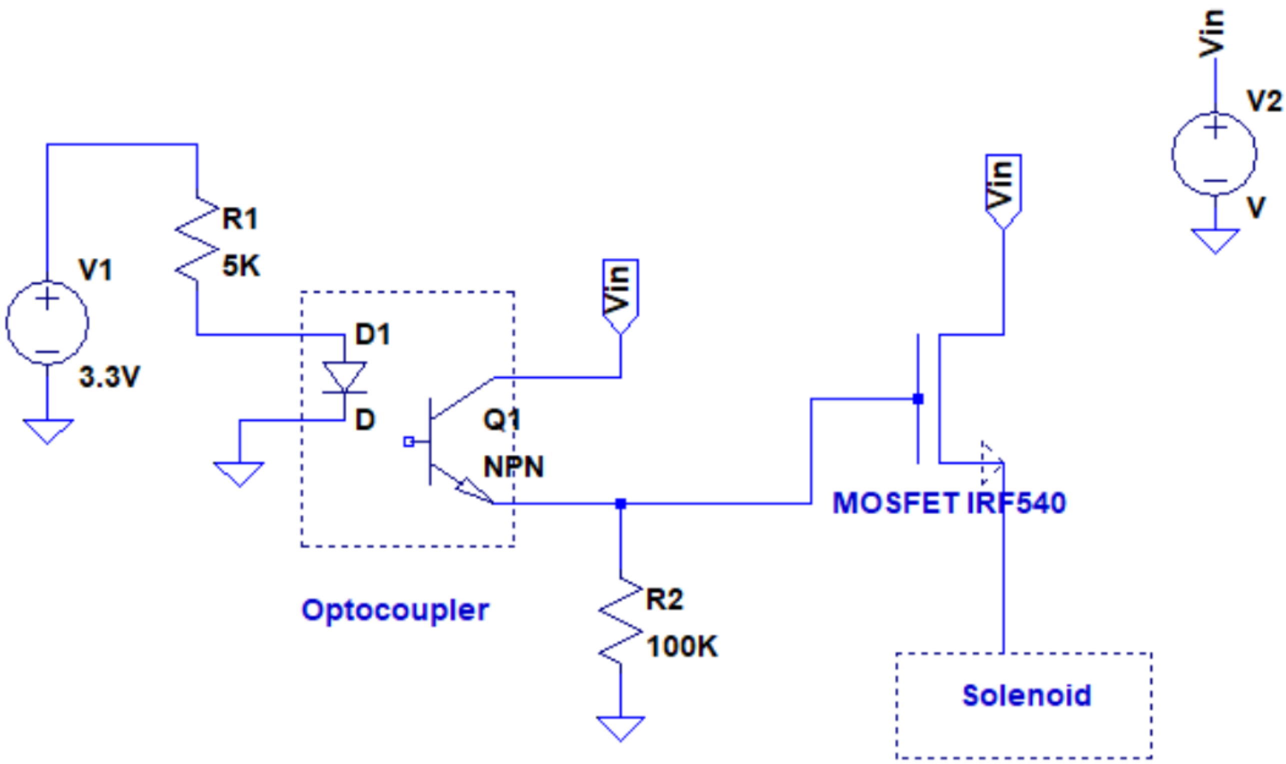


Figure 12 - Schematic of Door Security Lock

The output voltage of the FPGA is 3.3V, also from the datasheet we know that the maximum input voltage and current of optocoupler is 1.7V and 10mA, respectively. As a result, we put a 5K resistor between FPGA and optocoupler.

$$\frac{3.3V - 1.7V}{5K} = 3.3mA < 10mA$$

The reason why wo chose to connect a 100K resistor to the emitter of optocoupler is because the internal output impedance of transistor inside the optocoupler is quite small, a resistor of 100K will not affect its output. For example, assume internal output impedance to be 1K.

$$1K * \frac{100K}{100K + 1K} = 0.99K$$

1.11 Preliminary Analysis

FPGA is a low power circuitry, while the solenoid needs high power. In order to make these two components work well, we came up with a optocoupler and power MOSFET. The optocoupler will physically isolate high power circuitry from FPGA. The output current from optocoupler's emitter is around 50mA and solenoid needs 500mA to be driven, the power MOSFET can help us with this.

2. Equipment

- Quartus Prime
- Oscilloscope and scope, multimeter
- Breadboard
- DC Power supply
- FPGA, solenoid, proximity sensor, optocoupler, power MOSFET, 4X4 Matrix keypad, 7-segment LED display, speaker, resistors

3. Procedure

The objective is to create a door security lock, first of all, we thought out a lock with all the functions we want, and drew a complete block diagram so that we can start to work on one block by one block easily and clearly. Then, we programmed the FPGA on keypad, display, and speaker using Quartus Prime. After this, we started to work on the hardware, i.e. circuit. By looking through the datasheet and pinout of FGPA, optocoupler, and power MOSFET, we know how to connect them together correctly

4. Results

4.1 Block Diagram and Schematic

The followings are the block diagram and circuit we created. With keypad, LED display, speaker, and proximity sensor interfaced to FPGA, the FPGA will send signal to solenoid through optocoupler and MOSFET when any key is pressed. As mentioned above, we put a 5K resistor between FPGA and optocoupler, and 100K to the emitter of optocoupler.

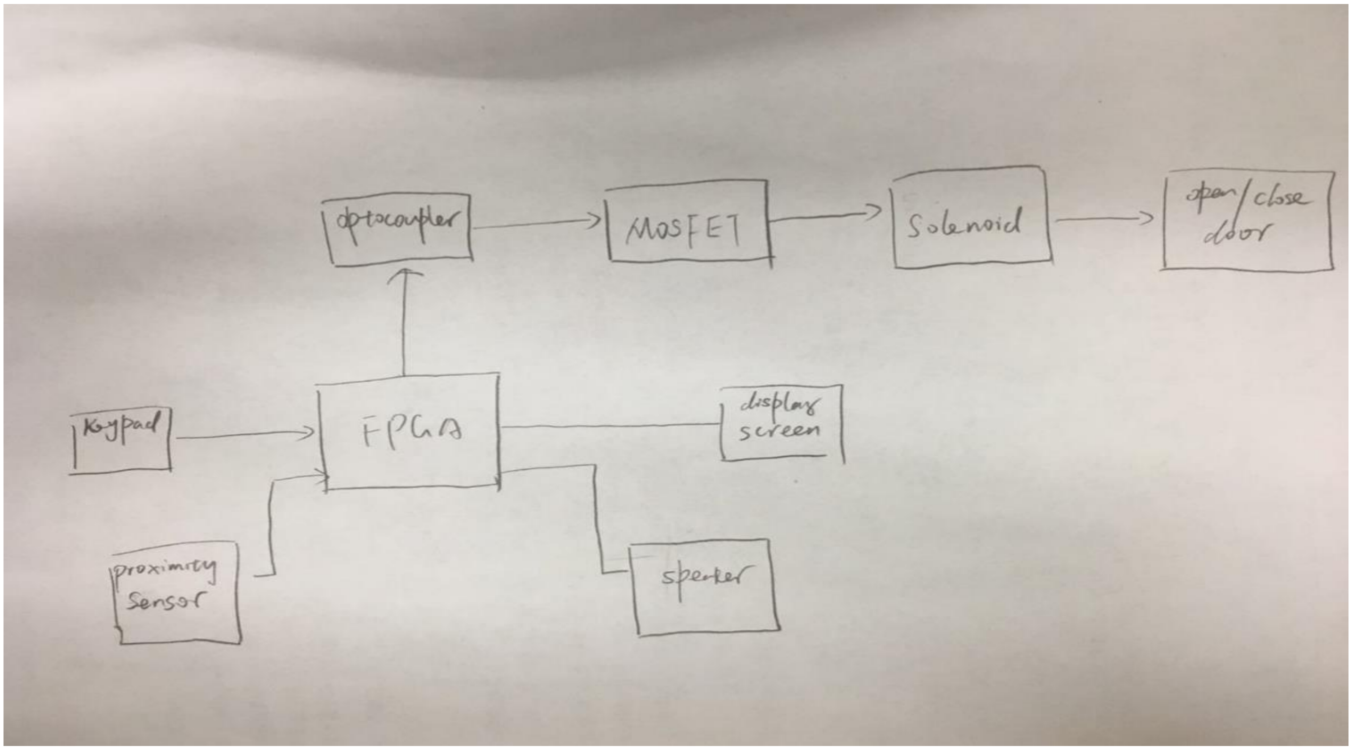


Figure 13 - Block Diagram of Door Security Lock

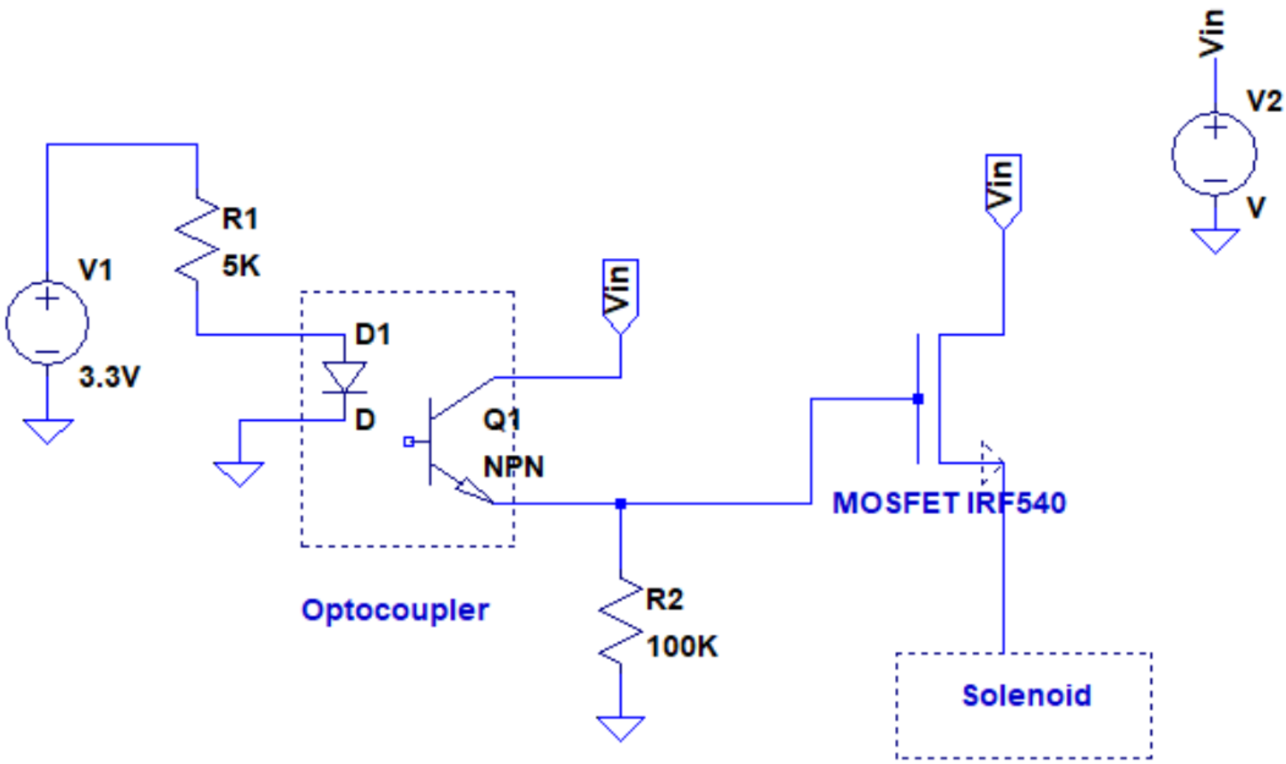


Figure 14 - Schematic of Door Security Lock

4.2 Real Circuit

The following is the real circuit we built.

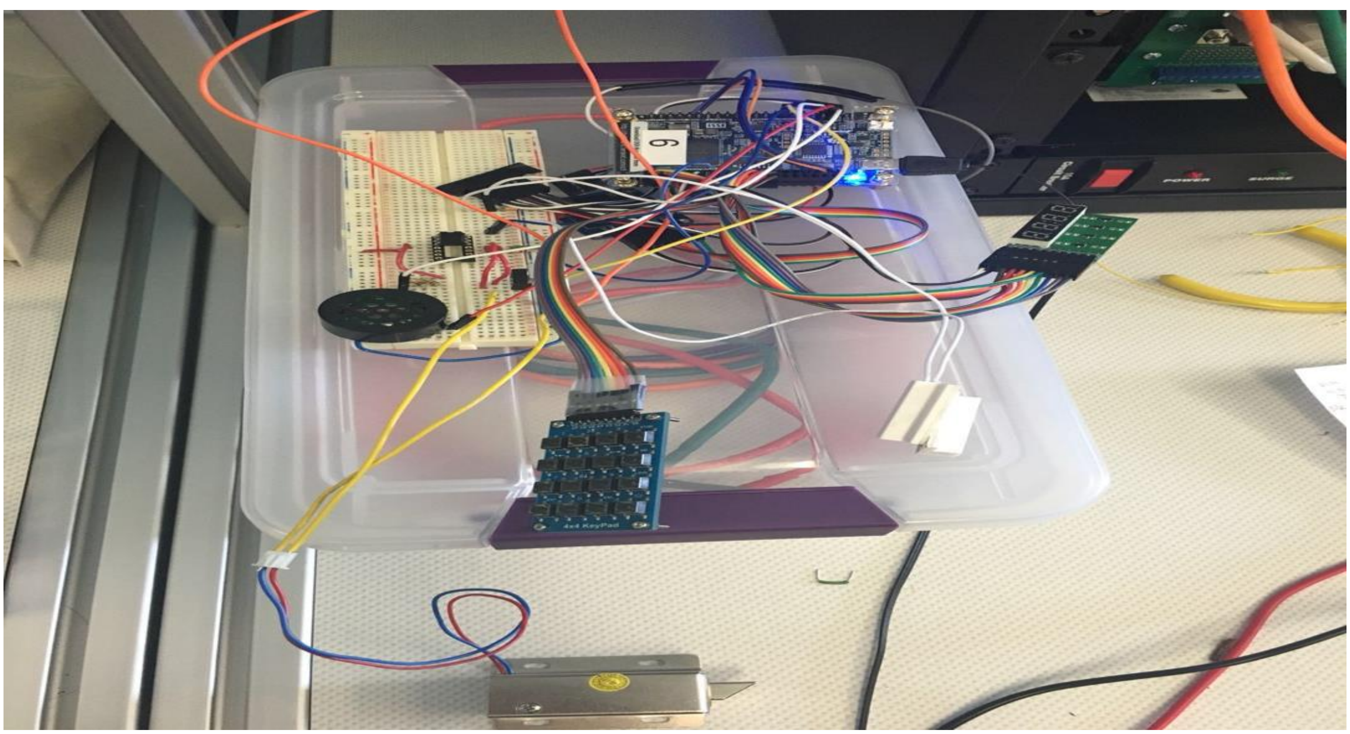


Figure 15 - Real Circuit of Door Security Lock

5. Discussion

We chose the optocoupler and power MOSFET to adjust the power, voltage, and current because it is simpler to implement, also our instructor Ed has these two components, so we don't need to pay extra to buy them. We should really check the circuit carefully or ask the instructor to check when connecting the circuit to FPGA, this can greatly prevent us from blowing the FPGA. One thing to note is when using op-amps, it is better that the resistance connected to input of op-amp should be not large (100K) and the resistance connected to the output of op-amp large enough (100K), so the internal impedance of op-amp won't affect the circuit much.

6. Summary and Conclusions

The primary goal of this lab is to design and build a door lock by programming the FPGA using Verilog language to send the electrical signal to the circuit composed of optocoupler and power MOSFET.

7. References

[1] Wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Field-programmable_gate_array

[2] Wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Proximity_sensor

[3] slideshare, 2018. [Online]. Available: https://www.slideshare.net/aswin5432/smart-door-lock

[4] Vishay, 2017. [Online]. Available: https://www.vishay.com/docs/91021/91021.pdf

[5] Xilinx, 2015. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug385.pdf

[6] ResearchGate, 2018. [Online]. Available: https://www.researchgate.net/publication/272296047

Appendices

Datasheet of Optocoupler 4N35 and Power MOSFET IRF540

Electrical Specifications ($T_A = 25^{\circ}C$)

Parameter	Symbol	Min.	Typ.	Max.	Units	Test Conditions
Forward Voltage	V_F	–	1.2	1.5	V	$I_F = 10$ mA
Reverse Current	I_R	–	–	10	µA	$V_R = 4$ V
Terminal Capacitance	C_t	–	50	–	pF	V = 0, f = 1 KHz
Collector Dark Current	I_{CEO}	–	–	50	nA	$V_{CE} = 10$ V, $I_F = 0$
Collector-Emitter Breakdown Voltage	BV_{CEO}	30	–	–	V	$I_C = 0.1$ mA, $I_F = 0$
Emitter-Collector Breakdown Voltage	BV_{ECO}	7	–	–	V	$I_E = 10$ µA, $I_F = 0$
Collector-Base Breakdown Voltage	BV_{CBO}	70	–	–	V	$I_C = 0.1$ mA, $I_F = 0$
Collector Current	I_C	2	–	–	mA	$I_F = 10$ mA
*Current Transfer Ratio	CTR	20	–	–	%	$V_{CE} = 10$ V
Collector-Emitter Saturation Voltage	$V_{CE(sat)}$	–	0.1	0.5	V	$I_F = 50$ mA, $I_C = 2$ mA
Response Time (Rise)	t_r	–	3	–	µs	$V_{CE} = 10$ V, $I_C = 2$ mA
Response Time (Fall)	t_f	–	3	–	µs	$R_L = 100\ \Omega$
Isolation Resistance	R_{iso}	5×10^{10}	1×10^{11}	–	Ω	DC 500 V 40 ~ 60% R.H.
Floating Capacitance	C_f	–	1	–	pF	V = 0, f = 1 MHz

Figure 16 - Datasheet of Optocoupler 4N35

ABSOLUTE MAXIMUM RATINGS ($T_C = 25$ °C, unless otherwise noted)

PARAMETER			SYMBOL	LIMIT	UNIT
Drain-Source Voltage			V_{DS}	100	V
Gate-Source Voltage			V_{GS}	± 20	
Continuous Drain Current	V_{GS} at 10 V	$T_C = 25$ °C	I_D	28	A
		$T_C = 100$ °C		20	
Pulsed Drain Current[a]			I_{DM}	110	
Linear Derating Factor				1.0	W/°C
Single Pulse Avalanche Energy[b]			E_{AS}	230	mJ
Repetitive Avalanche Current[a]			I_{AR}	28	A
Repetitive Avalanche Energy[a]			E_{AR}	15	mJ
Maximum Power Dissipation	$T_C = 25$ °C		P_D	150	W
Peak Diode Recovery dV/dt[c]			dV/dt	5.5	V/ns
Operating Junction and Storage Temperature Range			T_J, T_{stg}	- 55 to + 175	°C
Soldering Recommendations (Peak Temperature)	for 10 s			300[d]	
Mounting Torque	6-32 or M3 screw			10	lbf · in
				1.1	N · m

Notes

a. Repetitive rating; pulse width limited by maximum junction temperature (see fig. 11).
b. $V_{DD} = 25$ V, starting $T_J = 25$ °C, L = 440 µH, $R_g = 25\ \Omega$, $I_{AS} = 28$ A (see fig. 12).
c. $I_{SD} \leq 28$ A, dI/dt ≤ 170 A/µs, $V_{DD} \leq V_{DS}$, $T_J \leq 175$ °C.
d. 1.6 mm from case.

Figure 17 - Datasheet of Power MOSFET IRF540

Program listing

KEYPAD DECODING

```
module kpdecode ( input logic [3:0] kpc,kpr
                  output logic kphit,
                  output logic [3:0] num);

always_comb begin
        if ( !kpr[0] && !kpc[0])
            kphit = 1'b1;
            num = 8'b1010_0001;
        else if ( !kpr[0] && !kpc[1])
            kphit = 1'b1;
            num = 8'b1000_1110;
        else if ( !kpr[0] && !kpc[2])
            kphit = 1'b1;
            num = 8'b1100_0000;
        else if ( !kpr[0] && !kpc[3])
            kphit = 1'b1;
            num = 8'b1000_0110;
        else if ( !kpr[1] && !kpc[0])
            kphit = 1'b1;
            num = 8'b1100_0110;
        else if ( !kpr[1] && !kpc[1])
            kphit = 1'b1;
            num = 8'b1001_0000;
        else if ( !kpr[1] && !kpc[2])
            kphit = 1'b1;
            num = 8'b1000_0000;
        else if ( !kpr[1] && !kpc[3])
            kphit = 1'b1;
            num = 8'b1111_1000;
        else if ( !kpr[2] && !kpc[0])
            kphit = 1'b1;
            num = 8'b1000_0011;
        else if ( !kpr[2] && !kpc[1])
            kphit = 1'b1;
            num = 8'b1000_0010;
        else if ( !kpr[2] && !kpc[2])
            kphit = 1'b1;
            num = 8'b1001_0010;
        else if ( !kpr[2] && !kpc[3])
            kphit = 1'b1;
            num = 8'b1001_1001;
        else if ( !kpr[3] && !kpc[0])
            kphit = 1'b1;
            num = 8'b1000_1000;
        else if ( !kpr[3] && !kpc[1])
            kphit = 1'b1;
            num = 8'b1011_0000;
        else if ( !kpr[3] && !kpc[2])
            kphit = 1'b1;
            num = 8'b1010_0100;
        else if ( !kpr[3] && !kpc[3])
            kphit = 1'b1;
            num = 8'b1111_1001;

        else
            kphit = 1'b0;
```

```
    end
endmodule
```

7-Segment LED DISPLAY DECODER

```
// Course Name: Digital System Design (ELEX 7660 )
// Instructor Name: Ed Casar
// Student Name: Wenjian Chen
// Student Number: A00893336
// Set: 6S

module decode7 (input logic [3:0] num,
        output logic [7:0] leds); // create 4-digit input and 8-digit output


   always_comb begin
      unique case(num)          // use of verilog case statement
    0: leds = 8'b1100_0000; // corresponding output to the first input
    1: leds = 8'b1111_1001;
    2: leds = 8'b1010_0100;
    3: leds = 8'b1011_0000;
    4: leds = 8'b1001_1001;
    5: leds = 8'b1001_0010;
    6: leds = 8'b1000_0010;
    7: leds = 8'b1111_1000;
    8: leds = 8'b1000_0000;
    9: leds = 8'b1001_0000;
    A: leds = 8'b1000_1000;
    B: leds = 8'b1000_0011;
    C: leds = 8'b1100_0110;
    D: leds = 8'b1010_0001;
    E: leds = 8'b1000_0110;
    F; leds = 8'b1000_1110;


    default: leds = 8'b0000_0000; // set the default for bits not used
      endcase // end case statement
   end
endmodule // end module
```

SPEAKER

```
module buzz
(
   input logic  clk,                  // 2kHz clock for keypad scanning
   input logic  unlock, proximity,               // a key is pressed
   output logic speaker
);

  reg [2:0] state;
  reg [2:0] next_state;

  logic [15:0] t_counter;
  // Period is 0.005
  // 8000 click to make 5 sec
```

```
  localparam s_reset=3'b000,
             s_on=3'b001,
             s_off=3'b010;




  always @ (posedge clk) begin
      state <= next_state;
  end

  always @ (posedge clk) begin
    case(state)
      s_reset : if ((unlock==0)&&(proximity==0)) begin
                  next_state = s_on;
                end
                else next_state = s_reset;


      s_on      : if((unlock==0)&&(proximity==0)) begin
                  next_state = s_off;
                  speaker = 1;
                end
                else begin
                  next_state = s_reset;
                  speaker = 0;
                end

      s_off     : if((unlock==0)&&(proximity==0)) begin
                  next_state = s_on;
                  speaker = 0;
                end
                else begin
                  next_state = s_reset;
                  speaker = 0;
                end




      default : begin
                  next_state = s_reset;
                  speaker = 0;
                end
    endcase
  end

endmodule
```

FPGA PINOUT

```
set_location_assignment PIN_R8 -to CLOCK_50
set_location_assignment PIN_A15 -to LED[0]
set_location_assignment PIN_A13 -to LED[1]
set_location_assignment PIN_B13 -to LED[2]
```

```
set_location_assignment PIN_A11 -to LED[3]
set_location_assignment PIN_D1 -to LED[4]
set_location_assignment PIN_F3 -to LED[5]
set_location_assignment PIN_B1 -to LED[6]
set_location_assignment PIN_L3 -to LED[7]
# set_location_assignment PIN_J15 -to KEY[0]

set_location_assignment PIN_J15 -to reset_n

set_location_assignment PIN_E1 -to KEY[1]
set_location_assignment PIN_M1 -to SW[0]
set_location_assignment PIN_T8 -to SW[1]
set_location_assignment PIN_B9 -to SW[2]
set_location_assignment PIN_M15 -to SW[3]
set_location_assignment PIN_P2 -to DRAM_ADDR[0]
set_location_assignment PIN_N5 -to DRAM_ADDR[1]
set_location_assignment PIN_N6 -to DRAM_ADDR[2]
set_location_assignment PIN_M8 -to DRAM_ADDR[3]
set_location_assignment PIN_P8 -to DRAM_ADDR[4]
set_location_assignment PIN_T7 -to DRAM_ADDR[5]
set_location_assignment PIN_N8 -to DRAM_ADDR[6]
set_location_assignment PIN_T6 -to DRAM_ADDR[7]
set_location_assignment PIN_R1 -to DRAM_ADDR[8]
set_location_assignment PIN_P1 -to DRAM_ADDR[9]
set_location_assignment PIN_N2 -to DRAM_ADDR[10]
set_location_assignment PIN_N1 -to DRAM_ADDR[11]
set_location_assignment PIN_L4 -to DRAM_ADDR[12]
set_location_assignment PIN_M7 -to DRAM_BA[0]
set_location_assignment PIN_M6 -to DRAM_BA[1]
set_location_assignment PIN_L7 -to DRAM_CKE
set_location_assignment PIN_R4 -to DRAM_CLK
set_location_assignment PIN_P6 -to DRAM_CS_N
set_location_assignment PIN_G2 -to DRAM_DQ[0]
set_location_assignment PIN_G1 -to DRAM_DQ[1]
set_location_assignment PIN_L8 -to DRAM_DQ[2]
set_location_assignment PIN_K5 -to DRAM_DQ[3]
set_location_assignment PIN_K2 -to DRAM_DQ[4]
set_location_assignment PIN_J2 -to DRAM_DQ[5]
set_location_assignment PIN_J1 -to DRAM_DQ[6]
set_location_assignment PIN_R7 -to DRAM_DQ[7]
set_location_assignment PIN_T4 -to DRAM_DQ[8]
set_location_assignment PIN_T2 -to DRAM_DQ[9]
set_location_assignment PIN_T3 -to DRAM_DQ[10]
set_location_assignment PIN_R3 -to DRAM_DQ[11]
set_location_assignment PIN_R5 -to DRAM_DQ[12]
set_location_assignment PIN_P3 -to DRAM_DQ[13]
set_location_assignment PIN_N3 -to DRAM_DQ[14]
set_location_assignment PIN_K1 -to DRAM_DQ[15]
set_location_assignment PIN_R6 -to DRAM_DQM[0]
set_location_assignment PIN_T5 -to DRAM_DQM[1]
set_location_assignment PIN_L1 -to DRAM_CAS_N
set_location_assignment PIN_L2 -to DRAM_RAS_N
set_location_assignment PIN_C2 -to DRAM_WE_N
set_location_assignment PIN_F2 -to I2C_SCLK
set_location_assignment PIN_F1 -to I2C_SDAT
set_location_assignment PIN_G5 -to G_SENSOR_CS_N
set_location_assignment PIN_M2 -to G_SENSOR_INT
set_location_assignment PIN_A14 -to GPIO_2[0]
set_location_assignment PIN_B16 -to GPIO_2[1]
```

```
set_location_assignment PIN_C14 -to GPIO_2[2]
set_location_assignment PIN_C16 -to GPIO_2[3]
set_location_assignment PIN_C15 -to GPIO_2[4]
set_location_assignment PIN_D16 -to GPIO_2[5]
set_location_assignment PIN_D15 -to GPIO_2[6]
set_location_assignment PIN_D14 -to GPIO_2[7]
set_location_assignment PIN_F15 -to GPIO_2[8]
set_location_assignment PIN_F16 -to GPIO_2[9]
set_location_assignment PIN_F14 -to GPIO_2[10]
set_location_assignment PIN_G16 -to GPIO_2[11]
set_location_assignment PIN_G15 -to GPIO_2[12]
set_location_assignment PIN_E15 -to GPIO_2_IN[0]
set_location_assignment PIN_E16 -to GPIO_2_IN[1]
set_location_assignment PIN_M16 -to GPIO_2_IN[2]
set_location_assignment PIN_A8 -to GPIO_0_IN[0]
set_location_assignment PIN_D3 -to GPIO_0[0]
set_location_assignment PIN_B8 -to GPIO_0_IN[1]
set_location_assignment PIN_C3 -to GPIO_0[1]
set_location_assignment PIN_A2 -to GPIO_0[2]
set_location_assignment PIN_A3 -to GPIO_0[3]
set_location_assignment PIN_B3 -to GPIO_0[4]
set_location_assignment PIN_B4 -to GPIO_0[5]
set_location_assignment PIN_A4 -to GPIO_0[6]
set_location_assignment PIN_B5 -to GPIO_0[7]
set_location_assignment PIN_A5 -to GPIO_0[8]
set_location_assignment PIN_D5 -to GPIO_0[9]
set_location_assignment PIN_B6 -to GPIO_0[10]
set_location_assignment PIN_A6 -to GPIO_0[11]
set_location_assignment PIN_B7 -to GPIO_0[12]
set_location_assignment PIN_D6 -to GPIO_0[13]
set_location_assignment PIN_A7 -to GPIO_0[14]
set_location_assignment PIN_C6 -to GPIO_0[15]
set_location_assignment PIN_C8 -to GPIO_0[16]
set_location_assignment PIN_E6 -to GPIO_0[17]
set_location_assignment PIN_E7 -to GPIO_0[18]
set_location_assignment PIN_D8 -to GPIO_0[19]
set_location_assignment PIN_E8 -to GPIO_0[20]
set_location_assignment PIN_F8 -to GPIO_0[21]
set_location_assignment PIN_F9 -to GPIO_0[22]
set_location_assignment PIN_E9 -to GPIO_0[23]
set_location_assignment PIN_C9 -to GPIO_0[24]
set_location_assignment PIN_D9 -to GPIO_0[25]
set_location_assignment PIN_E11 -to GPIO_0[26]
set_location_assignment PIN_E10 -to GPIO_0[27]
set_location_assignment PIN_C11 -to GPIO_0[28]
set_location_assignment PIN_B11 -to GPIO_0[29]
set_location_assignment PIN_A12 -to GPIO_0[30]
set_location_assignment PIN_D11 -to GPIO_0[31]
set_location_assignment PIN_D12 -to GPIO_0[32]
set_location_assignment PIN_B12 -to GPIO_0[33]
set_location_assignment PIN_T9 -to GPIO_1_IN[0]
set_location_assignment PIN_F13 -to GPIO_1[0]
set_location_assignment PIN_R9 -to GPIO_1_IN[1]
set_location_assignment PIN_T15 -to GPIO_1[1]
set_location_assignment PIN_T14 -to GPIO_1[2]
set_location_assignment PIN_T13 -to GPIO_1[3]
set_location_assignment PIN_R13 -to GPIO_1[4]
set_location_assignment PIN_T12 -to GPIO_1[5]
set_location_assignment PIN_R12 -to GPIO_1[6]
```

```
set_location_assignment PIN_T11 -to GPIO_1[7]
set_location_assignment PIN_T10 -to GPIO_1[8]
set_location_assignment PIN_R11 -to GPIO_1[9]
set_location_assignment PIN_P11 -to GPIO_1[10]
set_location_assignment PIN_R10 -to GPIO_1[11]
set_location_assignment PIN_N12 -to GPIO_1[12]
set_location_assignment PIN_P9 -to GPIO_1[13]
set_location_assignment PIN_N9 -to GPIO_1[14]
set_location_assignment PIN_N11 -to GPIO_1[15]
set_location_assignment PIN_L16 -to GPIO_1[16]
set_location_assignment PIN_K16 -to GPIO_1[17]
set_location_assignment PIN_R16 -to GPIO_1[18]
set_location_assignment PIN_L15 -to GPIO_1[19]
set_location_assignment PIN_P15 -to GPIO_1[20]
set_location_assignment PIN_P16 -to GPIO_1[21]
set_location_assignment PIN_R14 -to GPIO_1[22]
set_location_assignment PIN_N16 -to GPIO_1[23]
set_location_assignment PIN_N15 -to GPIO_1[24]
set_location_assignment PIN_P14 -to GPIO_1[25]
set_location_assignment PIN_L14 -to GPIO_1[26]
set_location_assignment PIN_N14 -to GPIO_1[27]
set_location_assignment PIN_M10 -to GPIO_1[28]
set_location_assignment PIN_L13 -to GPIO_1[29]
set_location_assignment PIN_J16 -to GPIO_1[30]
set_location_assignment PIN_K15 -to proximity
set_location_assignment PIN_J13 -to unlock
set_location_assignment PIN_J14 -to speaker

set_location_assignment PIN_A2 -to qspb
set_location_assignment PIN_A8 -to qsa
set_location_assignment PIN_B8 -to qsb
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_D9 -to rgb_din
set_location_assignment PIN_E10 -to rgb_clk
set_location_assignment PIN_B11 -to rgb_cs
set_location_assignment PIN_D11 -to rgb_dc
set_location_assignment PIN_B12 -to rgb_res
```

```
set_location_assignment PIN_G15 -to jstk_sel
set_location_assignment PIN_A10 -to adc_cs_n
set_location_assignment PIN_B10 -to adc_saddr
set_location_assignment PIN_A9 -to adc_sdat
set_location_assignment PIN_B14 -to adc_sclk
set_location_assignment PIN_B3 -to spkr
set_location_assignment PIN_D12 -to point

set_instance_assignment -name WEAK_PULL_UP_RESISTOR ON -to jstk_sel
```

LOCK

```
module lock
(
    input logic     clk,                    // 2kHz clock for keypad scanning
   input logic  kphit,                  // a key is pressed
   input logic [3:0] num,             // value of pressed key
   output logic     unlock
);

  reg [2:0] state;
  reg [2:0] next_state;

  logic [15:0] t_counter;
  // Period is 0.005
  // 8000 click to make 5 sec

  localparam s_reset=3'b000,
             s1=3'b001,
             s2=3'b010,
             s3=3'b011,
             s4=3'b100,
             s5=3'b101,
             open=3'b110;



  localparam key1=4'h2,
             key2=4'h0,
             key3=4'h1,
             key4=4'h8,
             key5=4'hb,
                 keylock=4'hc;






  always @ (posedge clk) begin
      state <= next_state;
  end

  always @ (posedge clk) begin
    case(state)
      s_reset : begin
                  next_state = s1;
```

```
              unlock = 0;
                    t_counter = 0;
            end

      s1      : if(num==key1) begin
              next_state = s2;
              unlock = 0;
            end
            else begin
              next_state = s1;
              unlock = 0;
            end

      s2      : if(num==key2) begin
              next_state = s3;
              unlock = 0;
            end
            else begin
              next_state = s2;
              unlock = 0;
            end

      s3      : if(num==key3) begin
              next_state = s4;
              unlock = 0;
            end
            else begin
              next_state = s3;
              unlock = 0;
            end

      s4      : if(num==key4) begin
              next_state = s5;
              unlock = 0;
            end
            else begin
              next_state = s4;
              unlock = 0;
            end

      s5      : if(num==key5) begin
              next_state = open;
              t_counter = 0;
                    unlock = 0;
                    t_counter = 8000;
            end
            else begin
              next_state = s5;
              unlock = 0;
            end

      open    : if((num==keylock)||(t_counter==0)) begin
              next_state = s_reset;
              unlock = 0;
            end
            else begin
              next_state = open; // Keep the door open
              unlock = 1;
                    t_counter = t_counter - 1;
```

```
                end


        default : begin
                  next_state = s_reset;
                  unlock = 0;
                end
      endcase
    end

endmodule
```