



Table of Contents

[Abstract 2](#_30j0zll)

[Introduction 3](#_3znysh7)

[Design 3](#_2et92p0)

[Physical hardware design 3](#_tyjcwt)

[Synthesized hardware design 4](#_1t3h5sf)

[Design Structure 4](#_4d34og8)

[Clock Divider–clk\_dv.v 5](#)

[Lockout–lockout.v 5](#)

[Keypad Decoder–keypad\_decode.v 6](#_wddy84crczvr)

[Input Code–input\_code.v 7](#_g1r18omc74oa)

[Lock Mechanism–lock\_mechanism.v 9](#_888e27x97pe7)

[Seven Segment Display–seg\_7.v 11](#_pea0xwh8q4os)

[Top level–Lab4.v 12](#_7r53t3gz38hb)

[Simulation and Testing 13](#_17dp8vu)

[Problems 13](#_3rdcrjn)

[Results and Conclusion 13](#_26in1rg)

[Appendix A – RTL Diagrams 14](#_lnxbz9)

[Appendix B – Waveforms 17](#_o1liip29khcm)

[Appendix C – State Machine Diagrams 20](#_k4qyftxwhzoe)

[Appendix D – Pin Tables 22](#_z9rzchshg9wc)

[Table of Figures 24](#_35nkun2)

[References 25](#_1ksv4uv)

# Abstract

This lab is to create a digital door lock system. The lock will use a 4x4 membrane keypad to enter a 4-digit code. The code will be checked against valid codes and the door either unlocked or an error LED will be flashed. The system should consist of a couple of LEDs. One will be lit to indicate the pretend door is unlocking. The second will be lit to indicate an error. When either of these LEDs is lit it should remain in that state for a total of 5 seconds. The system should accept a 4-digit code from the keypad. This code will be entered in one digit at a time. The code should be displayed on the DE1-SoC 7-segment displays as it is being entered. To finish the code, the ‘#’ key should be pressed. Once this is pressed the code should be checked against a list of known good codes. If found, the door will be unlocked, otherwise the error will be shown. If the ‘\*’ key is entered at any point in the code entry. The code should be cleared. Also, once started, if the code is not entered within 15 seconds, it should error out and reset. This is broken down into its various modules using hierarchical design.

# Introduction

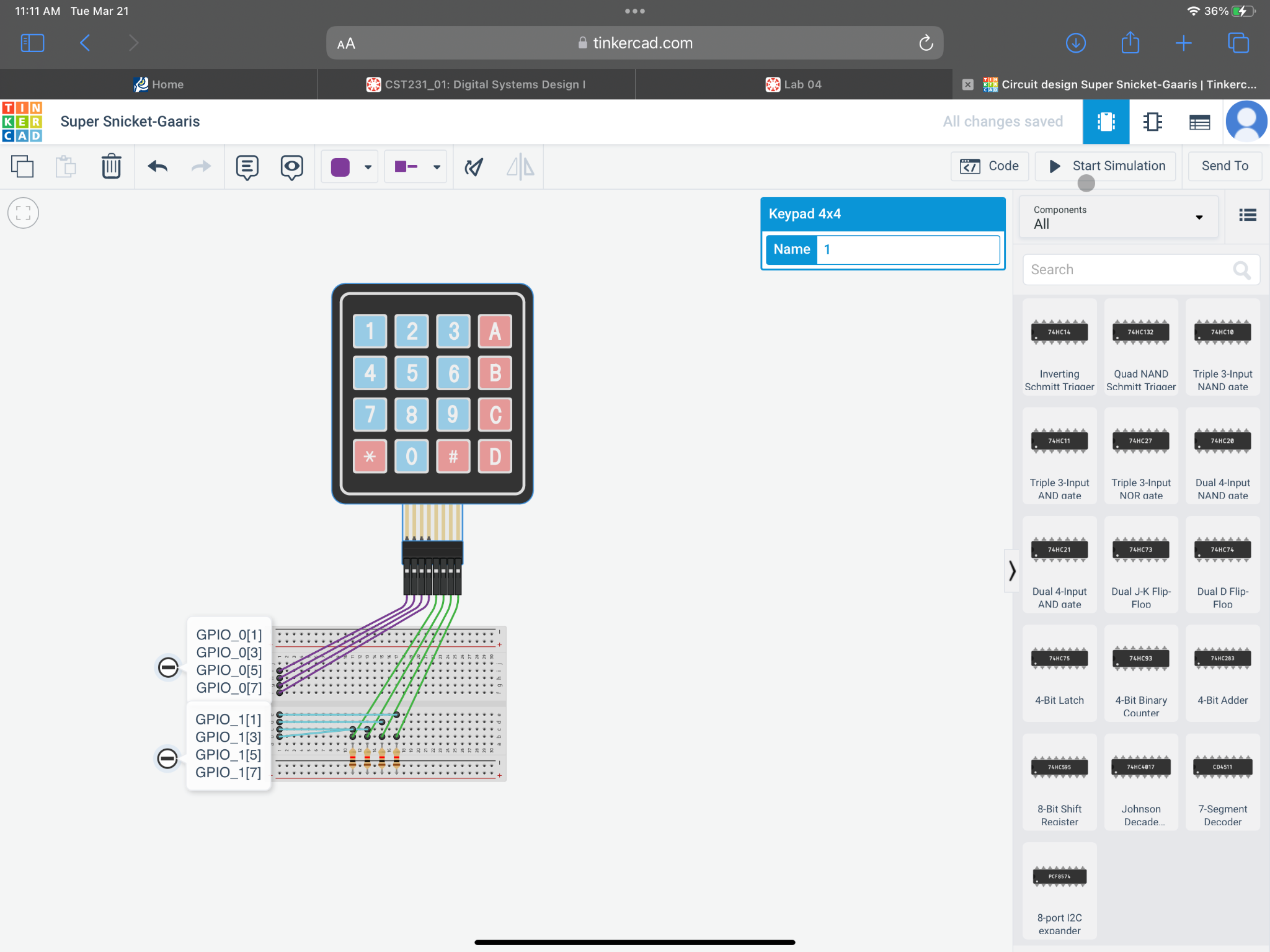
The purpose of this lab is to create the basis for a digital door locking system, practicing interfacing, clock division, and state machines.

# Design

Here, the physical hardware and synthesized hardware designs are discussed.

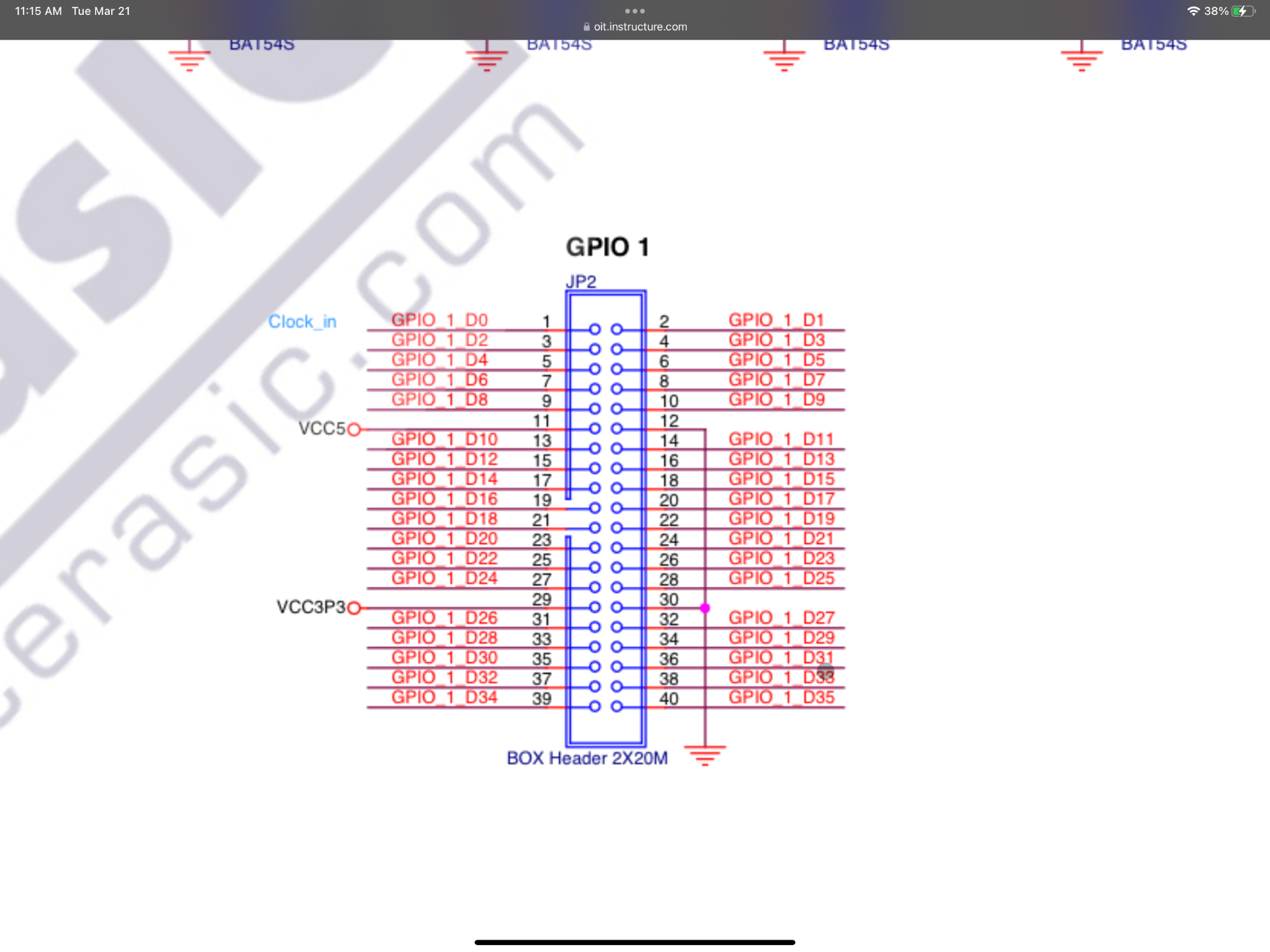
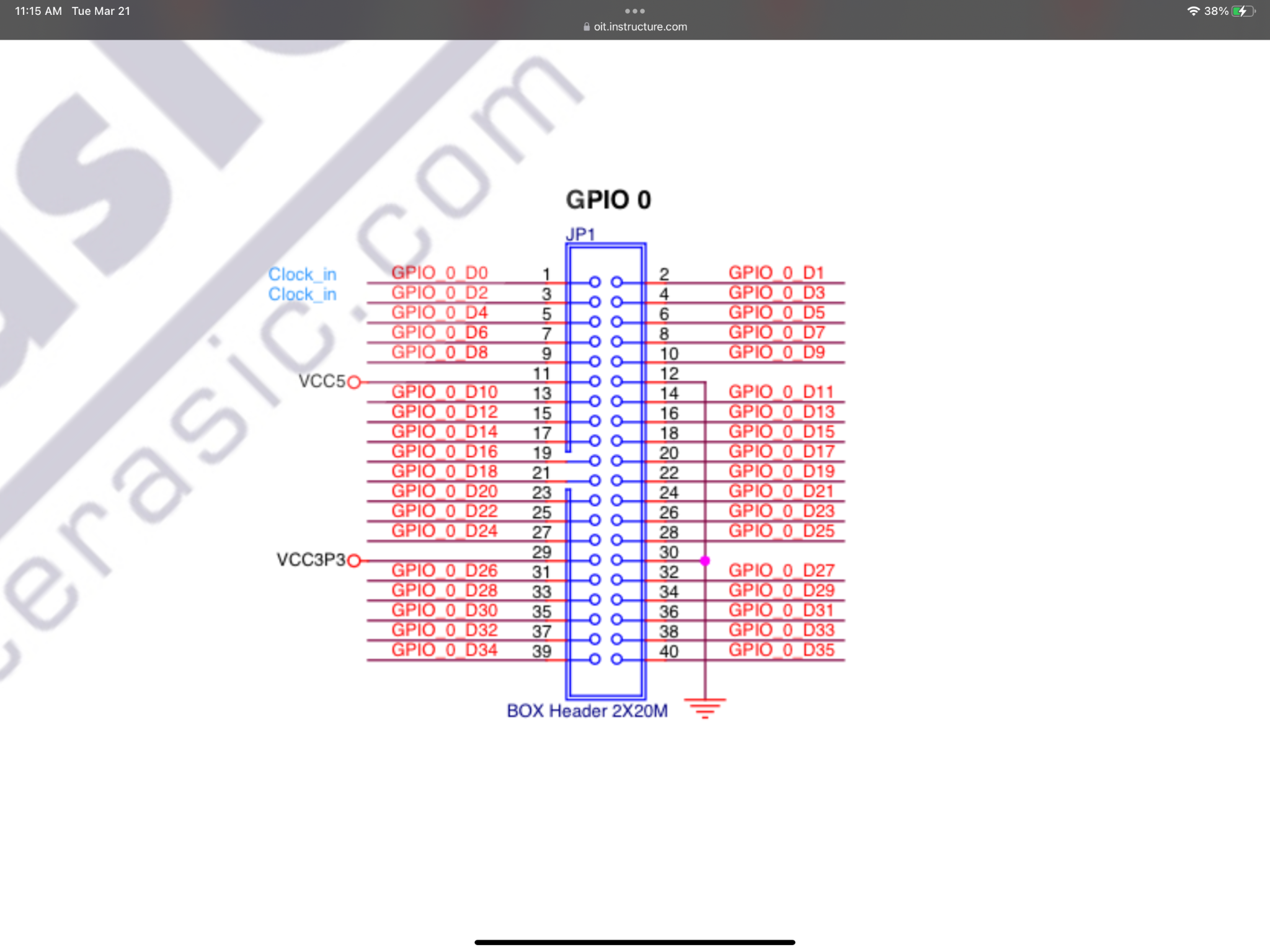
## Physical hardware design

A 4x4 membrane keypad is used for key presses. Four of its pins are connected to GPIO for output (the rows) and four of its pins are connected to the breadboard, and to pull-up resistors, which are also connected to the GPIO as input.



*Figure 1: Hardware Connection Diagram*

The four purple wires are connected to the four GPIO pins listed next to them, and likewise with the blue wires. The purple are row outputs, and the blue are column inputs. The (+) and (-) terminals are powered with the power and ground GPIO pins labeled in the DE1-SoC manual below.



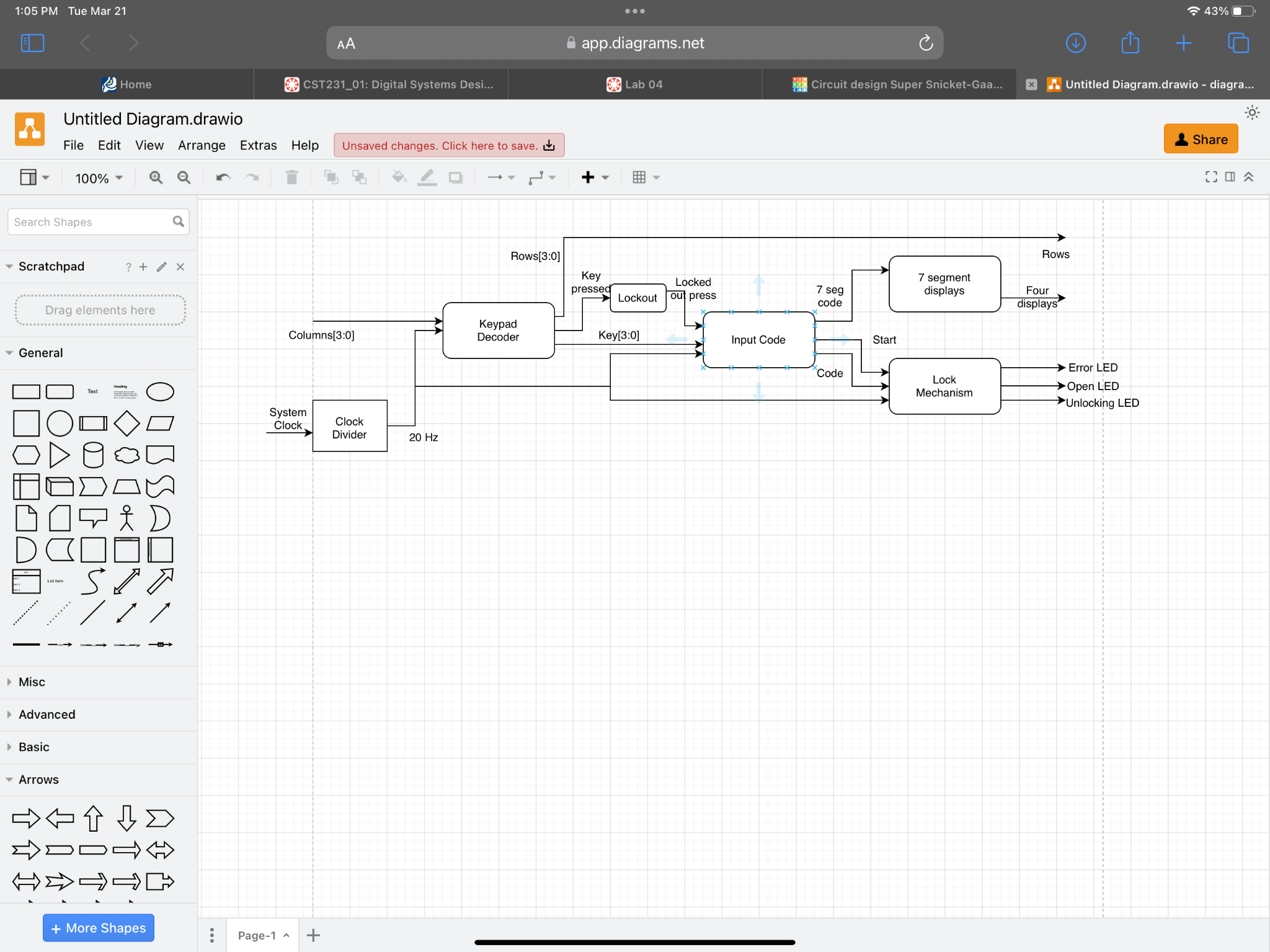
*Figure 2: GPIO pinout diagrams from DE1-SoC User Manual*

## Synthesized hardware design

In this section the major functional modules of the Design hierarchy are discussed.

### Design Structure

Here is the block diagram for the completed project:



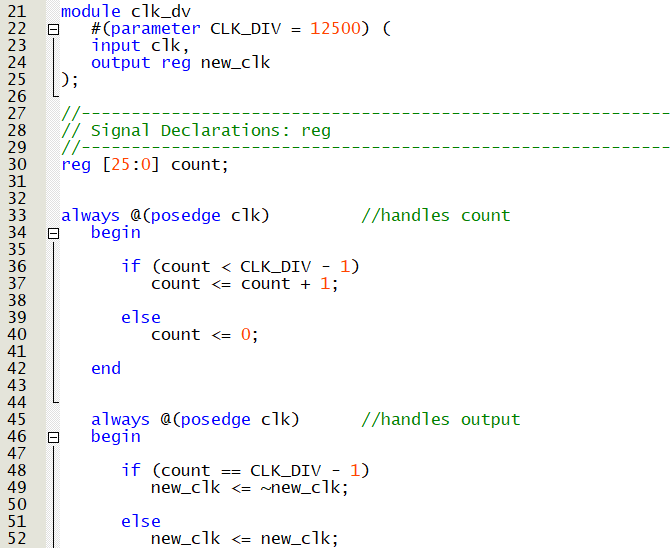
*Figure 3: Hierarchical Block Diagram*

A clock divider takes the system clock and generates a 20 Hz clock used by the decoder, input code state machine, and lock mechanism modules. The lockout module takes the signal from the decoder module indicating a button was pressed and latches it for the input code module. The input code module takes that signal, the key that was pressed from the decoder, and the divided clock, and outputs the code the current pressed keys to seven segments, and once the # key is hit, sends an output signal to start checking the input code, and sends out the full input code. The lock mechanism takes the clock divider, the start signal, and the input code from the input code module and outputs the corresponding LEDs for whether there was an error, the door has been unlocked, or the code is being checked– “unlocking”.

Each module is further described below.

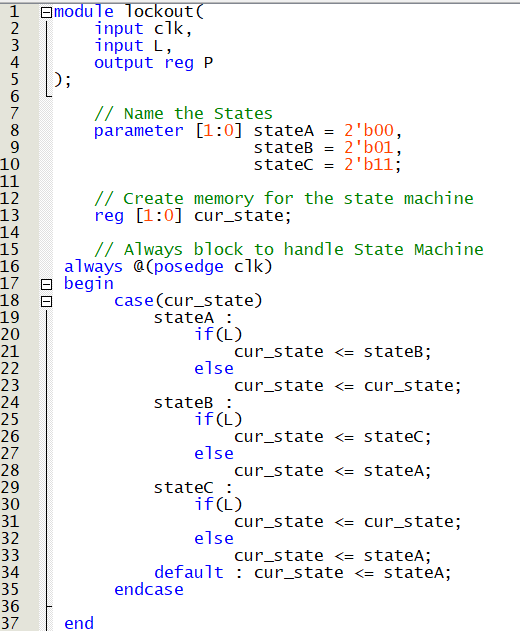
#### Clock Divider–clk\_dv.v

The clock divider takes the system clock as input and modifies its clock output based on a given parameter to produce a new clock of the desired frequency. The RTL for this module can be found in Appendix A. The module code follows.



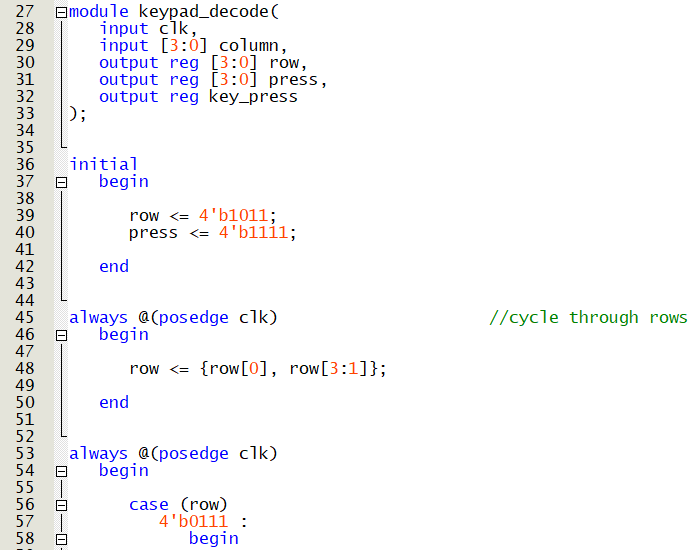
#### Lockout–lockout.v

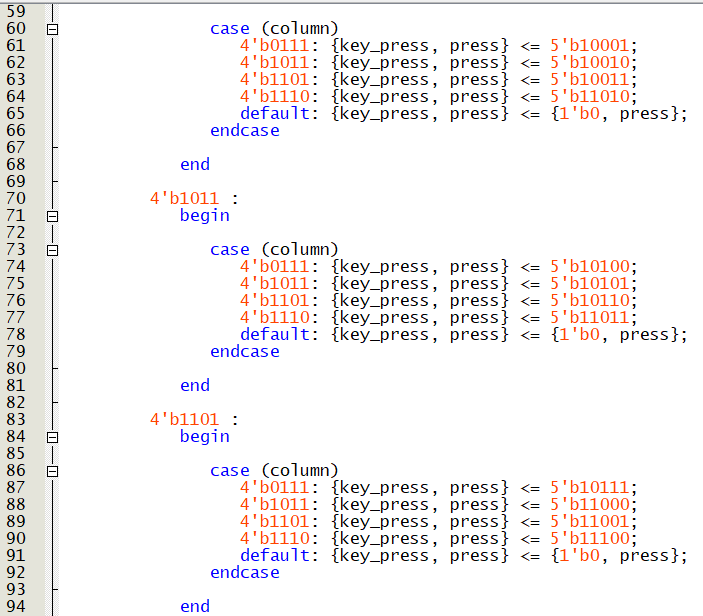
The lockout module takes in a clock, a reset, and an “L” signal. This module is a state machine which kicks off when “L” is high. The state machine defaults to state A. When “L” is high, it transitions to state B on the next positive clock edge. If “L” is high in state B, it transitions to state C, otherwise back to state A. If “L” is high in state C, the machine remains in state C until it goes low, where it transitions back to state A again to wait for the next input change. The module’s sole output register, “P,” is only high in state B, and is only ever in state B for one clock cycle, transforming a level signal to a pulse. The module code is below. The state diagram can be found in Appendix C, and the RTL diagram can be found in Appendix A.

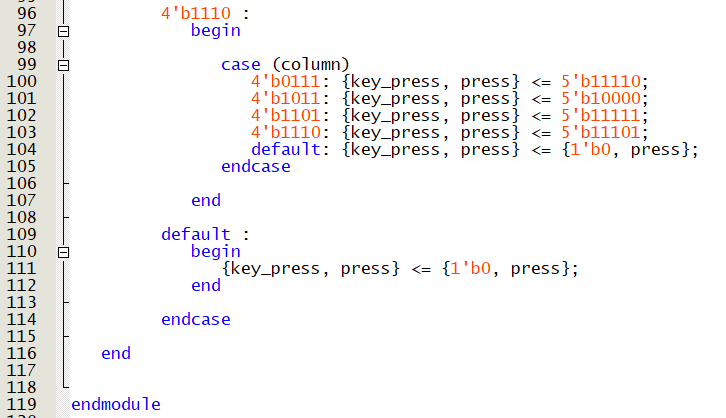


#### Keypad Decoder–keypad\_decode.v

The keypad decoder outputs the rows and inputs the columns to/from the keypad. In one always block, the rows are activated one at a time, cycling through to the next each clock pulse. In another always block, if a key is pressed, the current row and pressed column determine the key that was pressed, and outputs are adjusted appropriately. Key\_press becomes high, and the press output is determined by the column and row. The module’s code is found below. The state machine diagram can be found in Appendix C, and the RTL diagram can be found in Appendix A.

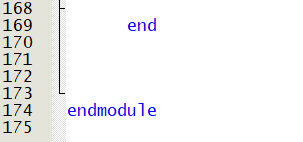
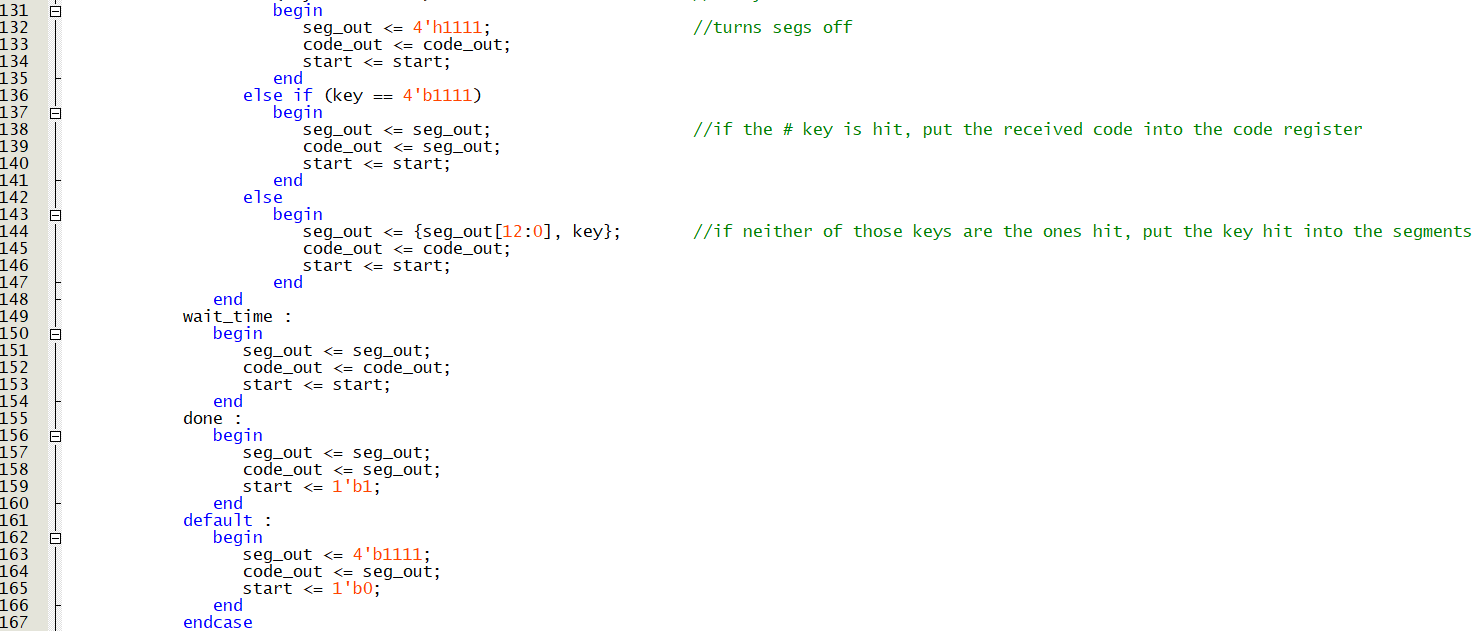
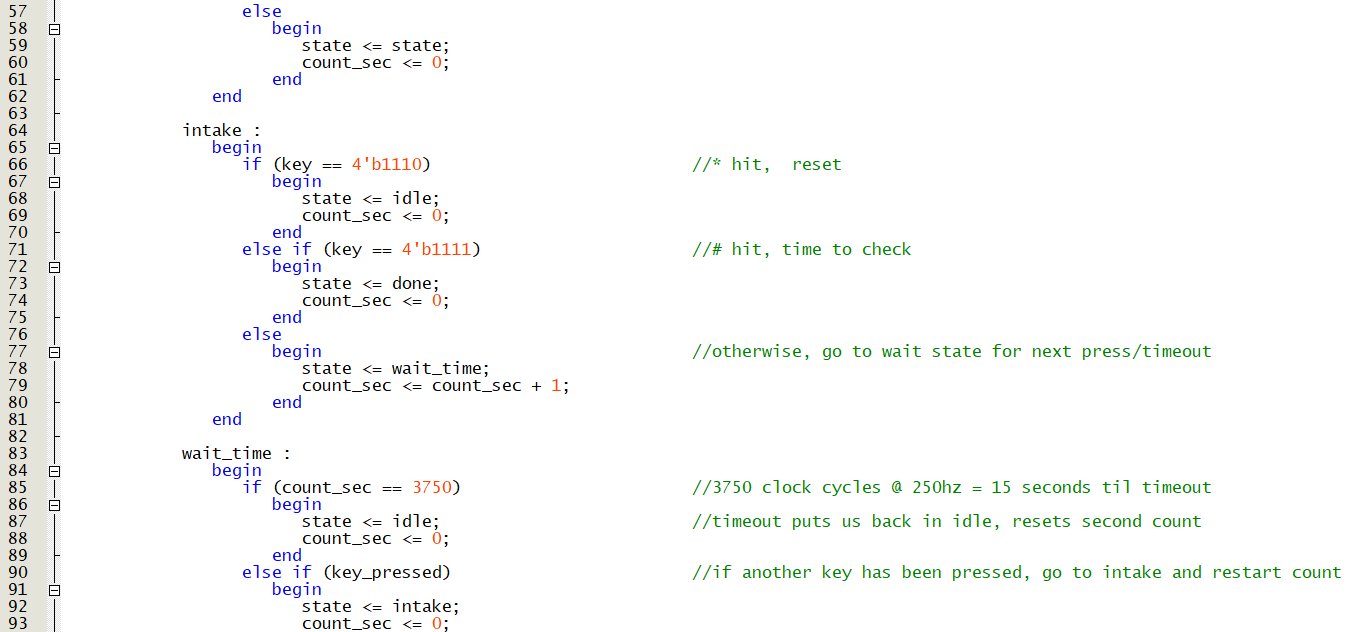
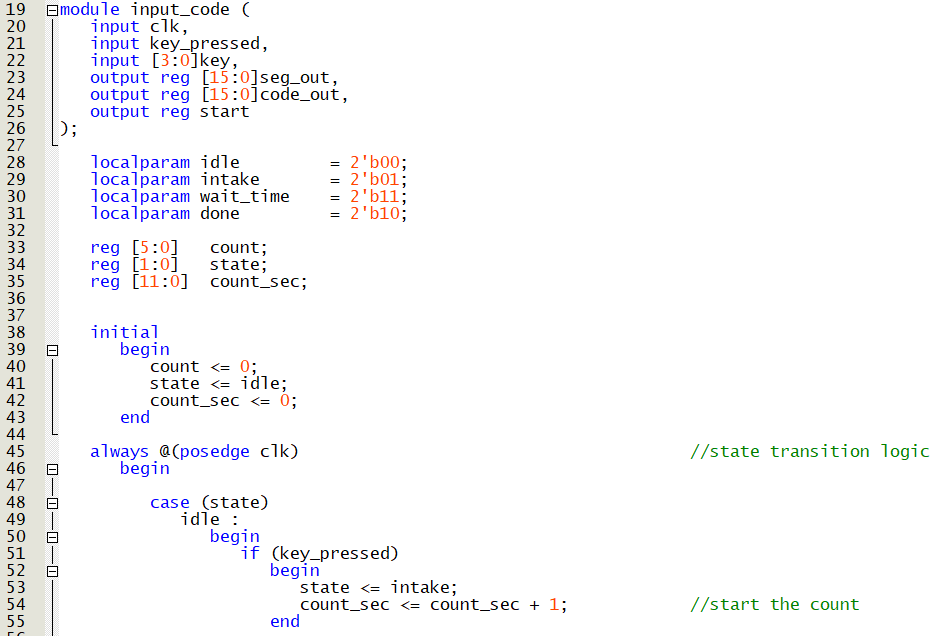






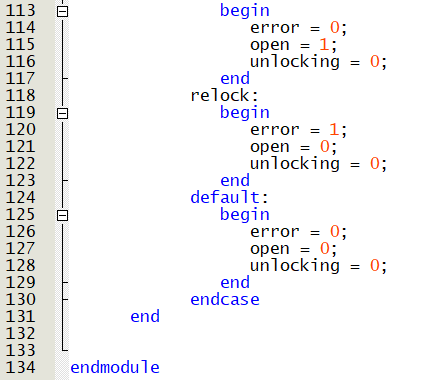
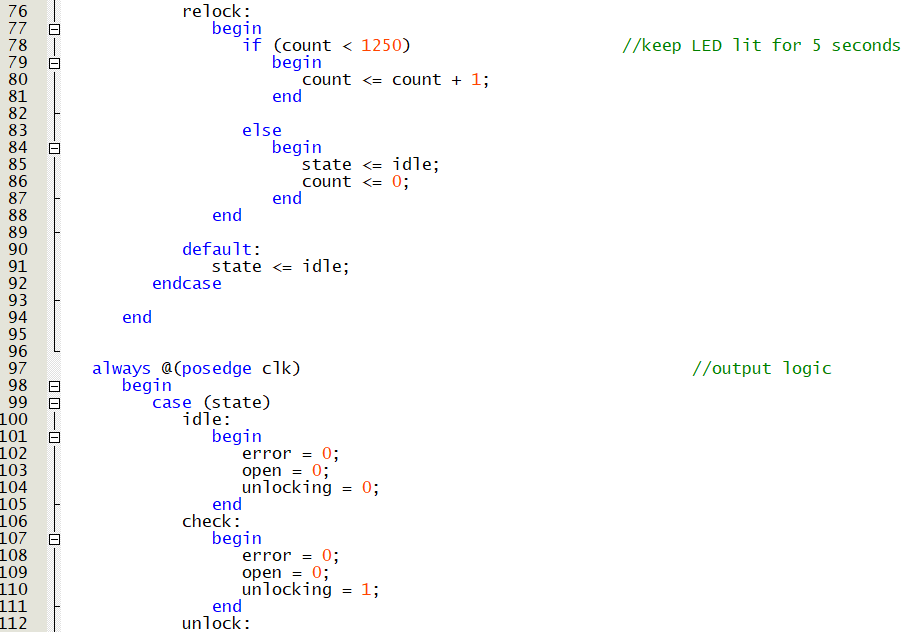
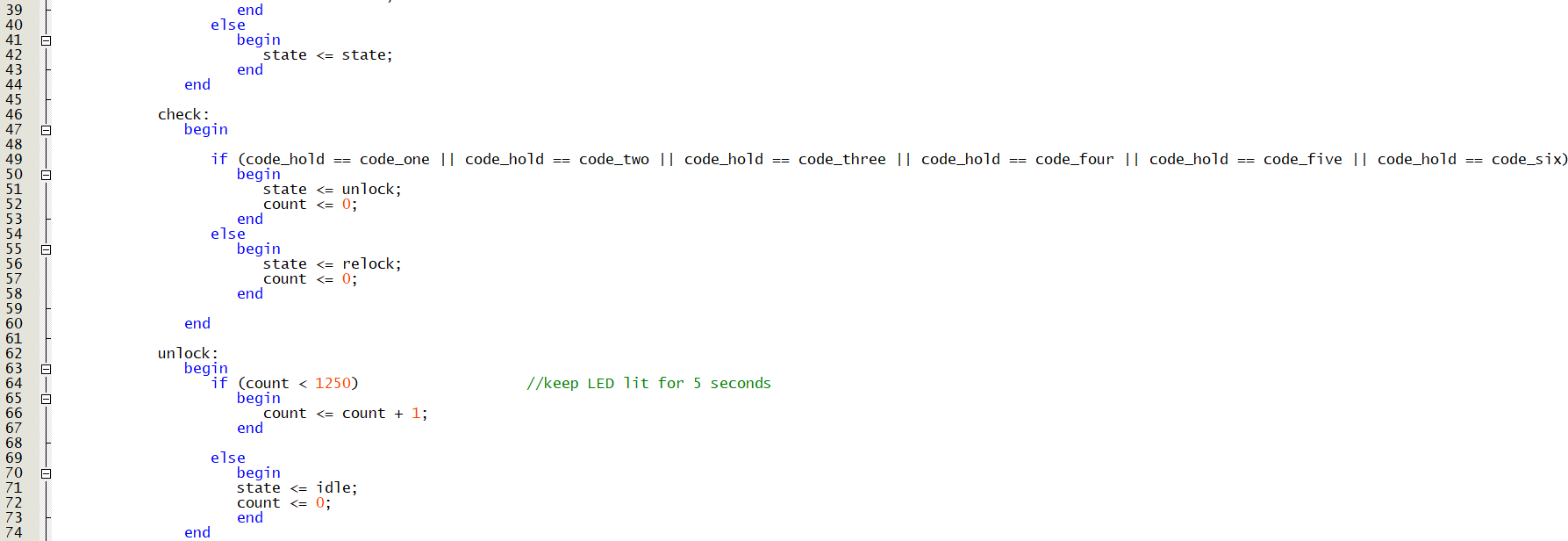
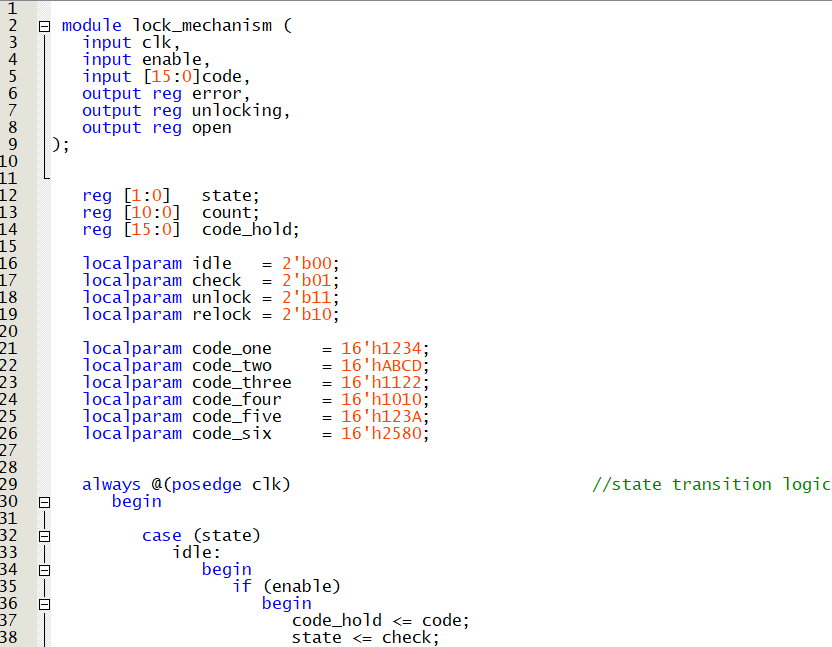
#### Input Code–input\_code.v

The input code module receives the key\_pressed signal, the number of the key that was pressed, and outputs the current code as “seg\_out” and the final code after a # is received as “code\_out,” along with a “start” signal to identify the time to check the code against the current code for the next module. The input code module is a state machine that stays in idle until a key is pressed. In idle, the outputs stay what they were and start is low. Once the key\_pressed signal is received, the state machine kicks off into the “intake” state, and a count starts. If the key was \*, the state machine transitions back to idle. If the key was “#”, the state transitions to done. In the intake state, if a key other than \* or # was pressed, the seg\_out and code\_out outputs are shifted with the key that was pressed. Without a \* or # key pressed, the state transitions to “wait\_time.” In this waiting state, the count continues. If the count reaches 15 seconds (which must be adjusted for clock division), the timeout is reached and the state goes back to idle. If a key is pressed before the 15 seconds mark, the state goes to intake and the count is restarted. Until the 15 second mark is reached or another key is pressed, the state machine stays in the wait stage. In the done stage, the count is restarted, the start output goes high, and the machine transitions back to idle. The code is below. Appendix A contains the RTL for this diagram, Appendix B contains the waveforms from simulation, and Appendix C contains the state machine diagram.



#### Lock Mechanism–lock\_mechanism.v

The lock mechanism module is a state machine to compare the input code with a list of compatible codes. The module begins in the idle state, and stays until the enable signal goes high. In idle, all three output signals are low. If the enable signal is high, the current code input is held in a temporary register, and the state transitions to check. In check, the unlocking signal goes high, and error and open signals stay low. The machine stays in the check state for fifteen seconds. In the check state, the register with the code is checked against a list of codes that will unlock. If the code is found to be an unlocking code, the machine transitions to the unlock state. Otherwise, it transitions to the relock state. The machine stays in the unlock state for five seconds, with the open signal high. The machine stays in the relock state for five seconds, with the error signal high. Both unlock and relock states transition back to idle after five seconds. Appendix A contains the RTL for this diagram, Appendix B contains the waveforms from simulation, and Appendix C contains the state machine diagram.



#### Seven Segment Display–seg\_7.v

The seven segment display module takes in a reset and four bits of data representing a hex number and translates it to a seven bit register output to display the hex number on a seven segment display on the DE1-SoC board, which requires an active low configuration. The module’s code is given below. The module’s RTL diagram can be found in Appendix A, and simulation waveforms can be found in Appendix B.



#### Top level–Lab4.v

The top level module for this design connects all of the given modules together. Its inputs are the system clock and the four columns. Its outputs are four seven segment displays, and three led’s for error, unlocking, and open. The module’s code is given below. The module’s RTL diagram can be found in Appendix A.



# Simulation and Testing

Testing and simulation did not go as planned. The keypad decoder simulates as it should, but never operated properly on the board. Surely it is an issue of wiring, but the problem wasn’t pinpointed before the end of term. Each individual module was simulated and operating properly. Those waveforms can be viewed in Appendix B.

# Problems

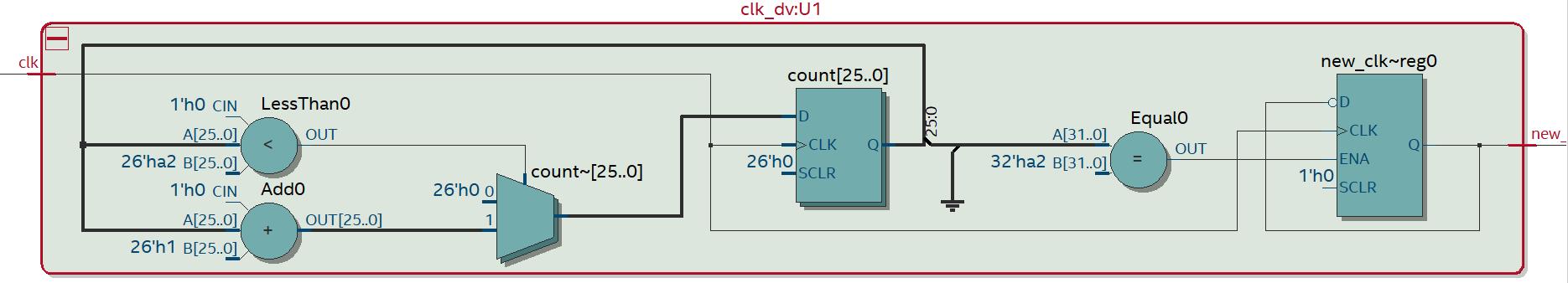
Testing did not go as planned. The keypad decoder simulates as it should, but never operated properly on the board. Surely it is an issue of wiring, but the problem wasn’t pinpointed before the end of term. Therefore, the full project could not be tested on the board, because the preliminary of getting one key press to show up on one 7-segment display failed. The keypad was switched out, different GPIO pins were tried, different clock speeds, all new wires–-none of which solved the issue.

# 

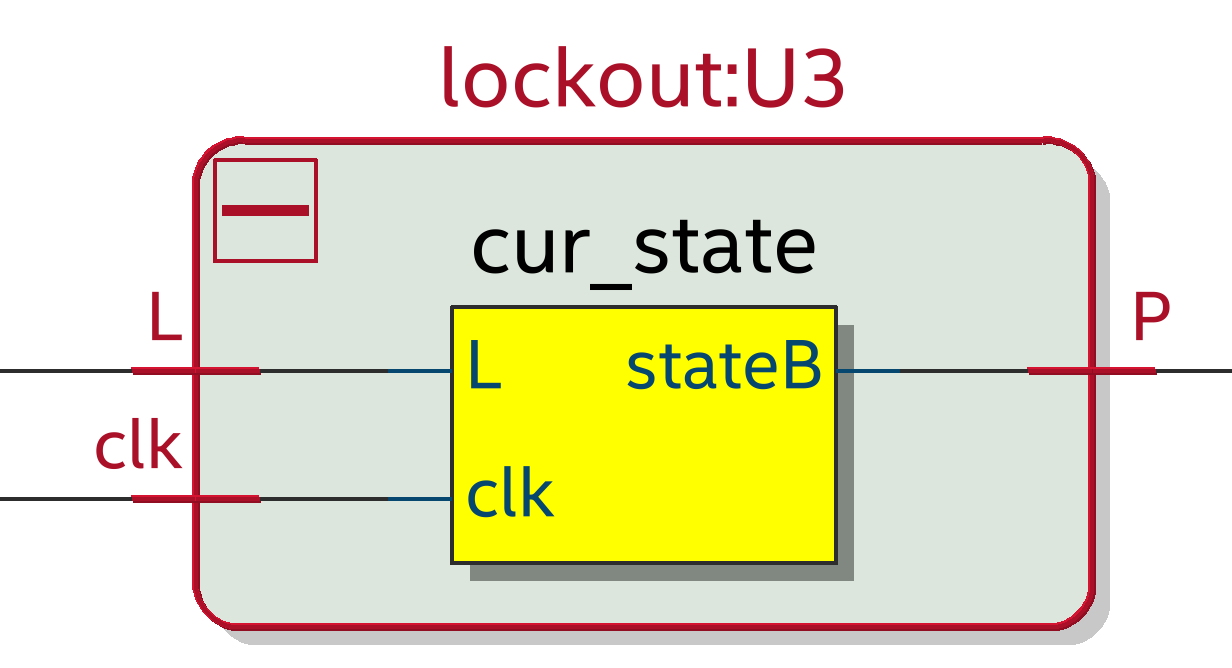
# Results and Conclusion

The result of this lab is a failed attempt at getting a keypad to simulate a digital door lock. Given more time, it may have been fully operational, but the end of the term came swiftly and the project was left incomplete.

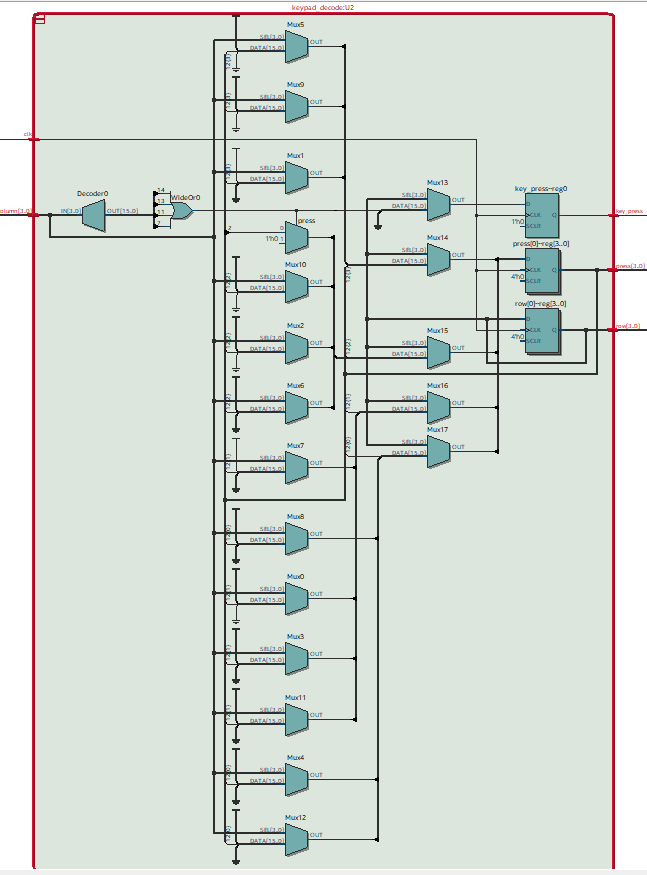
# Appendix A – RTL Diagrams



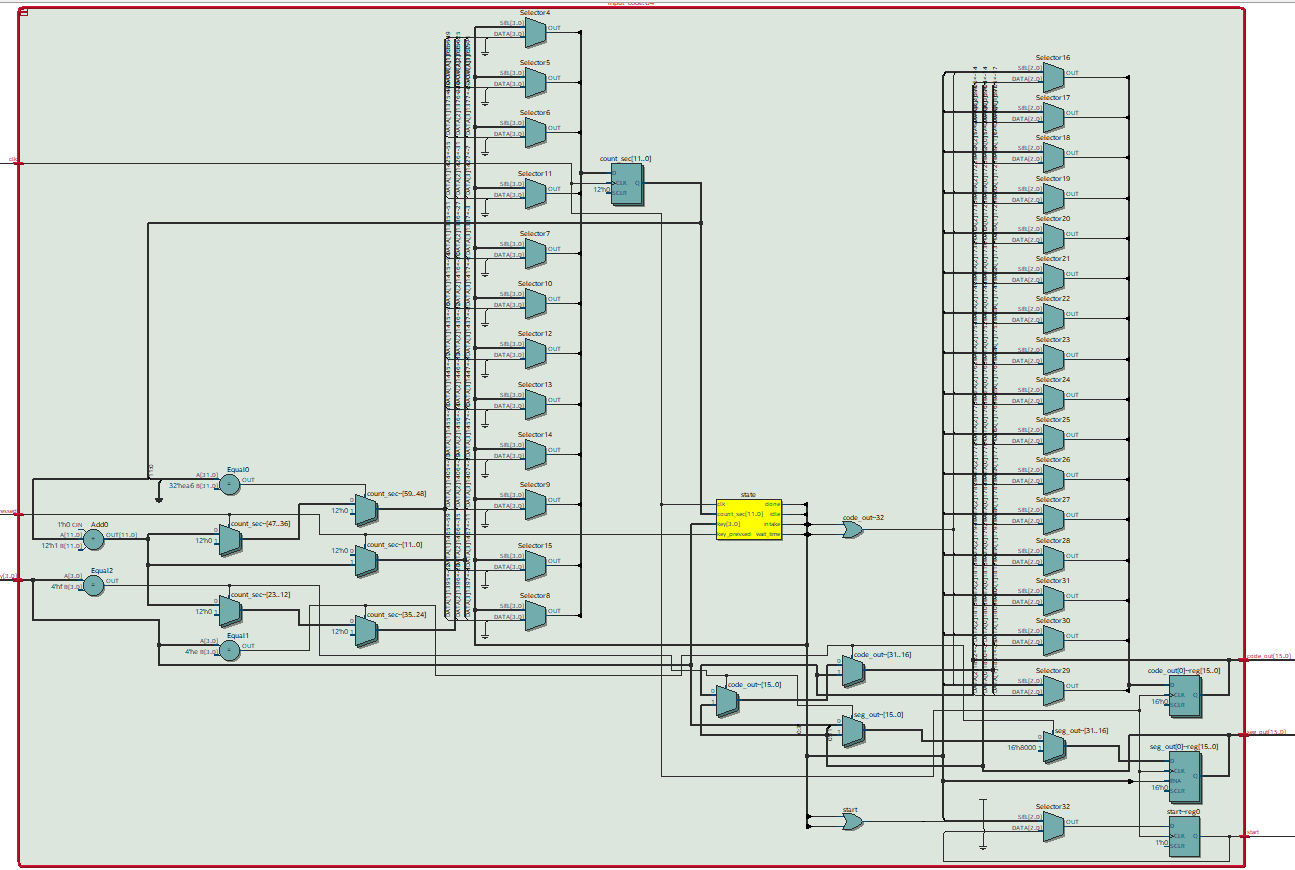
*Figure 4: Clock Divider RTL Diagram*



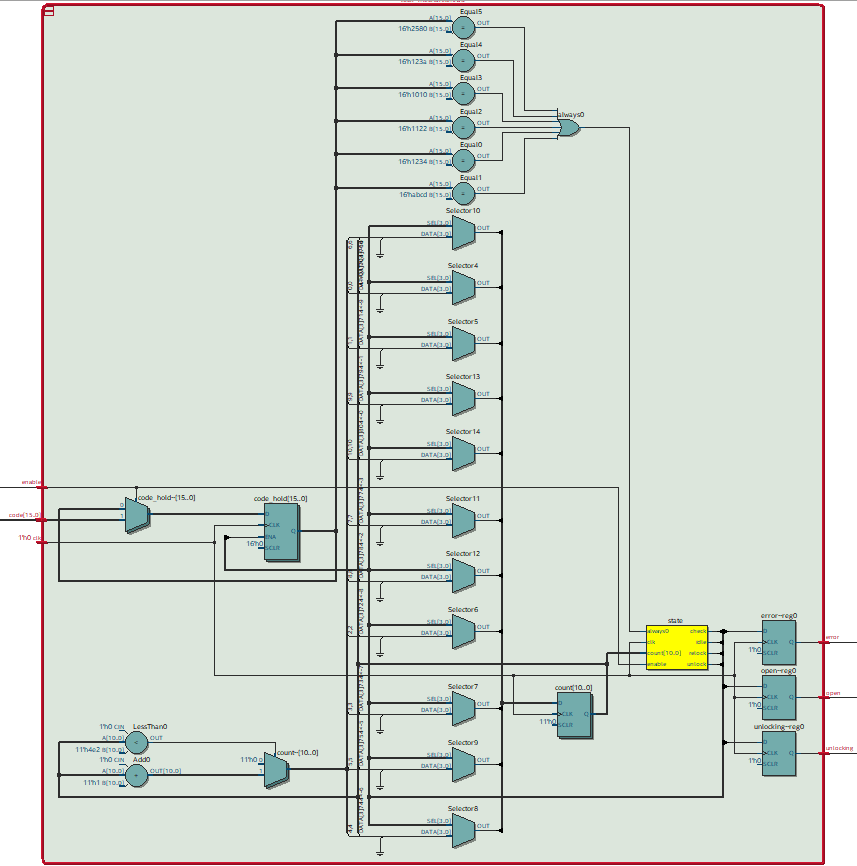
*Figure 5: Lockout RTL Diagram*

**

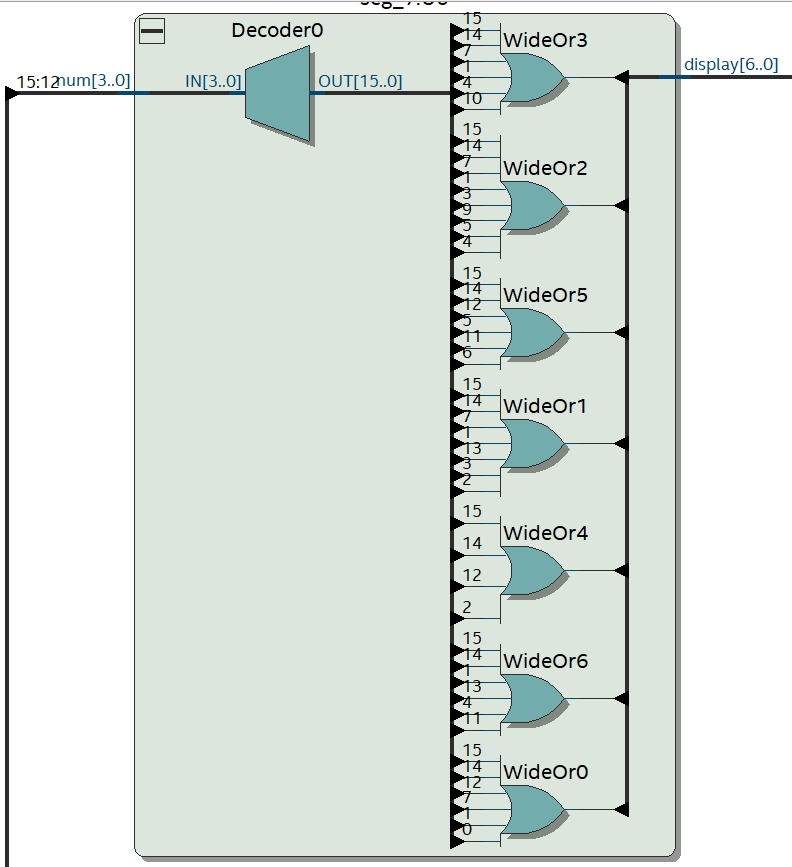
*Figure 6: Keypad Decode RTL Diagram*

**

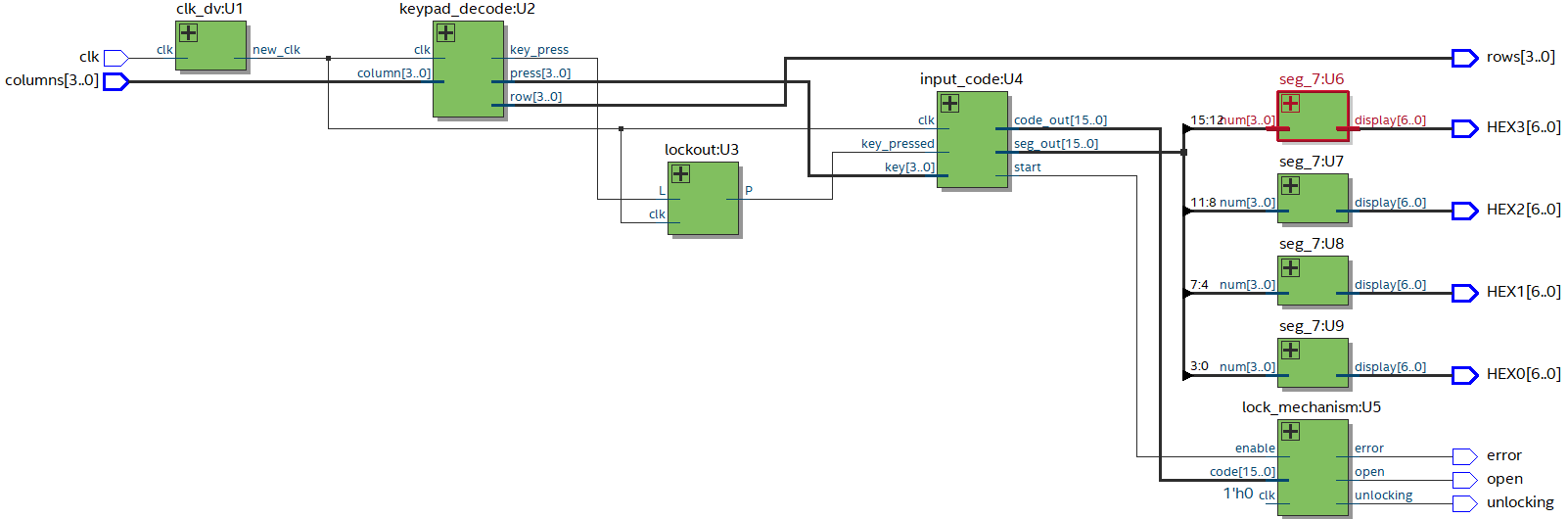
*Figure 7: Input Code RTL Diagram*



*Figure 8: Lock Mechanism RTL Diagram*

**

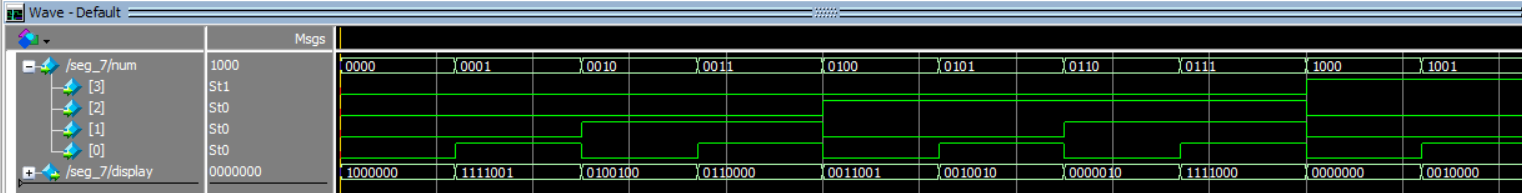
*Figure 9: Seven Segment Display RTL Diagram*

**

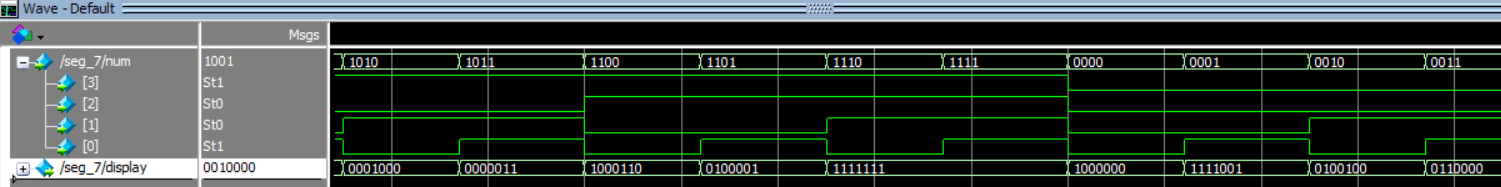
*Figure 10: Top Level RTL Diagram*

# 

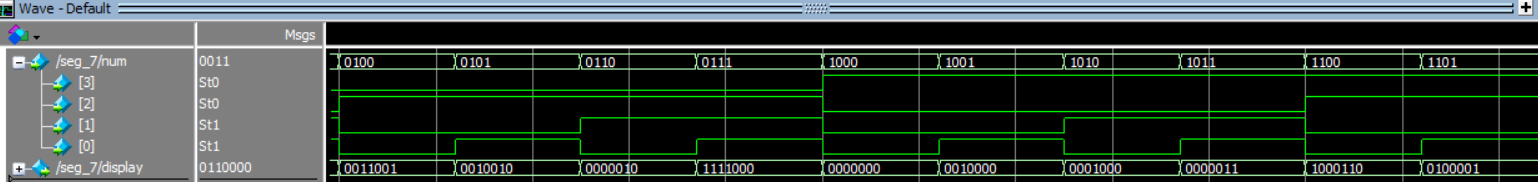
# Appendix B – Waveforms



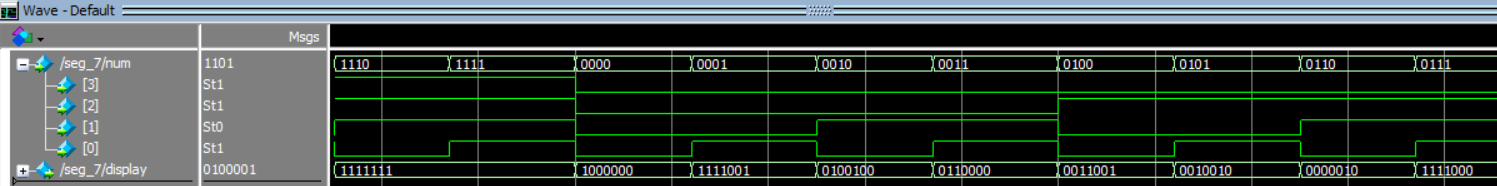
*Figure 11: Seven Segment waveforms part one*



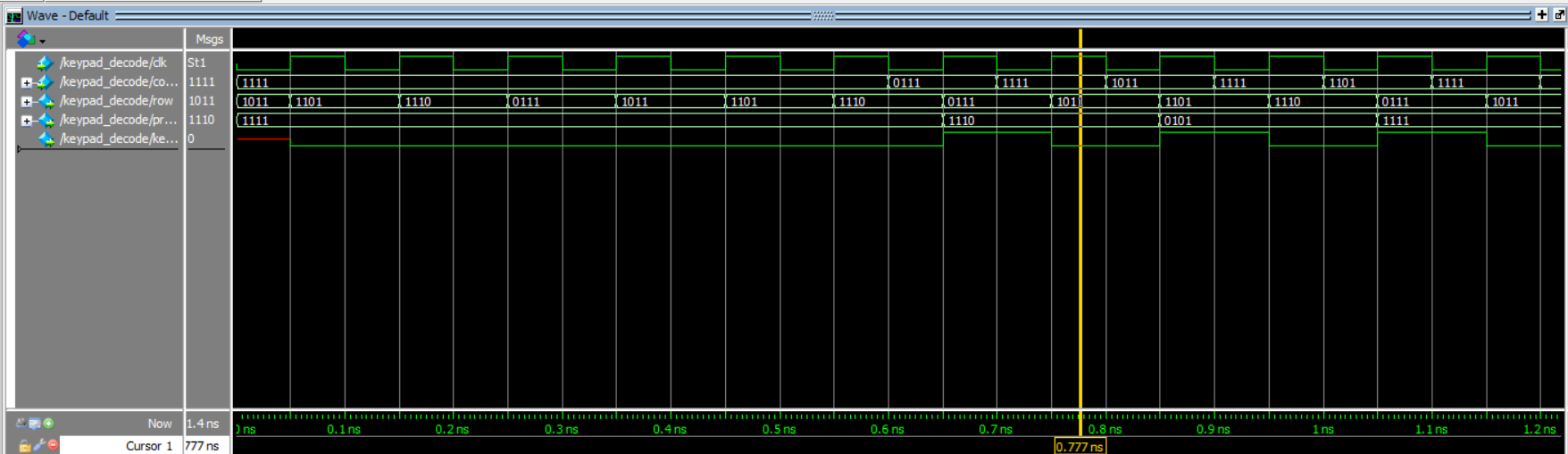
*Figure 12: Seven Segment waveforms part two*



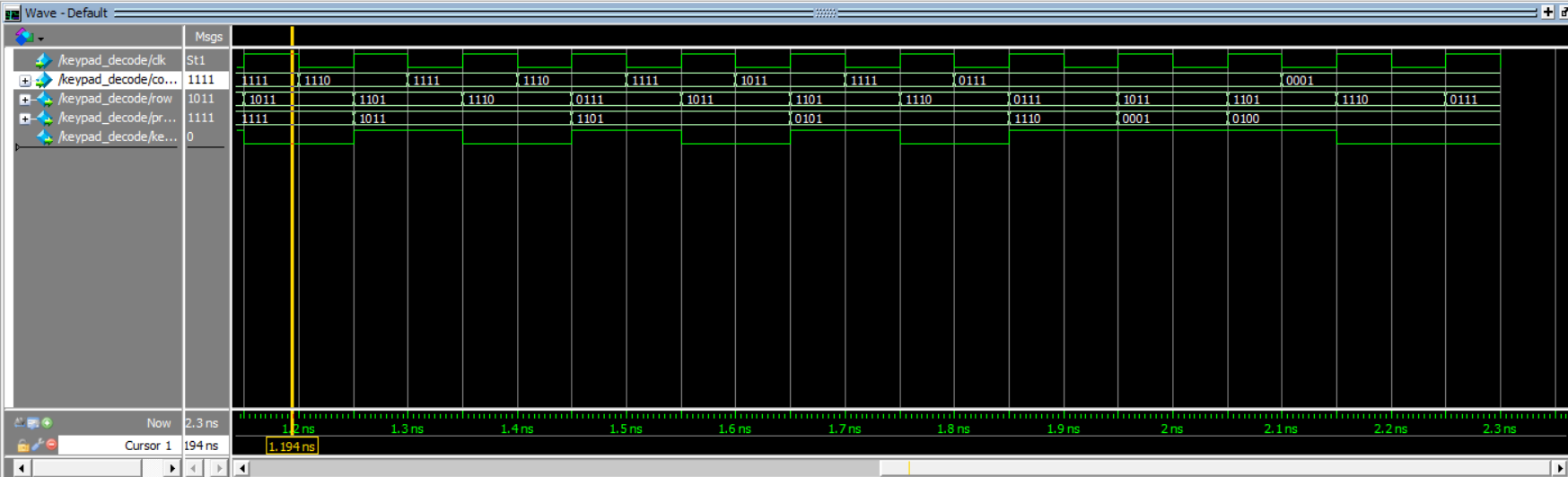
*Figure 13: Seven Segment waveforms part three*



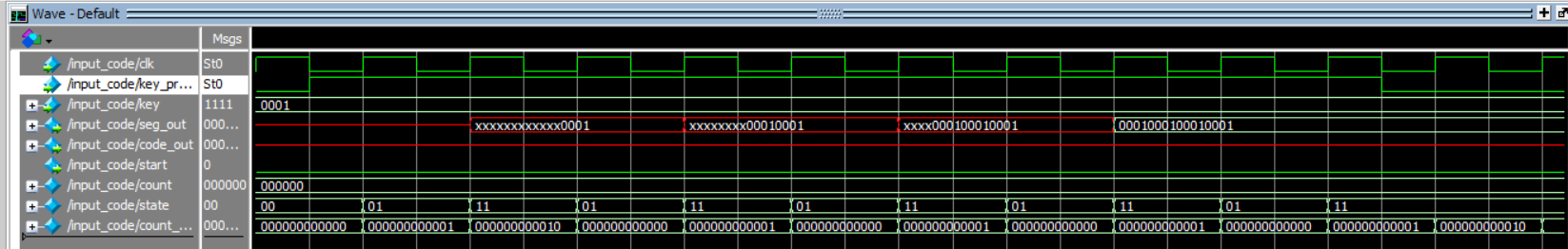
*Figure 14: Seven Segment waveforms part four*

**

*Figure 15: Keypad Decoder waveforms part one*

**

*Figure 16: Keypad Decoder waveforms part two*

**

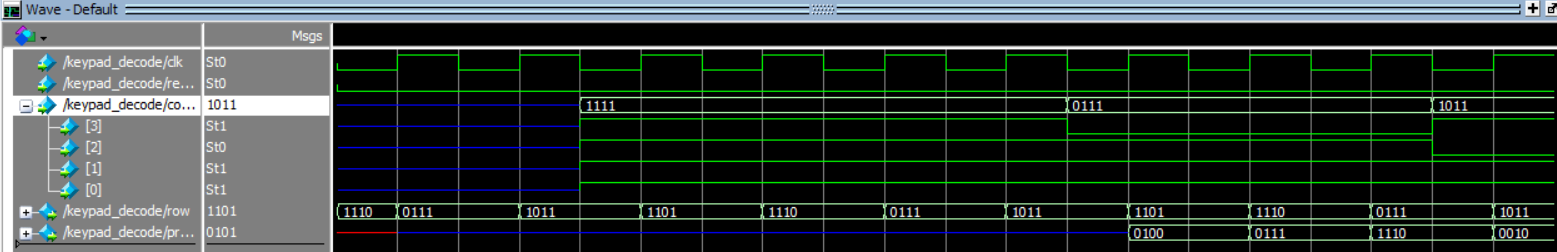
*Figure 17: Input Code waveforms part one*

**

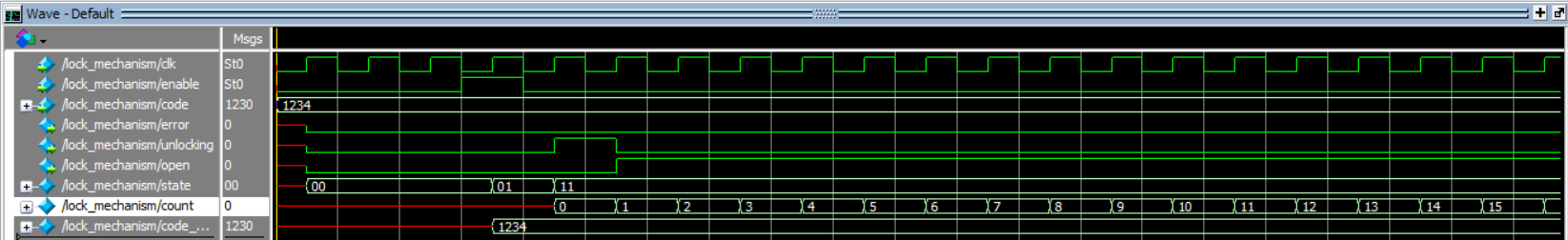
*Figure 18: Input Code waveforms part two*

**

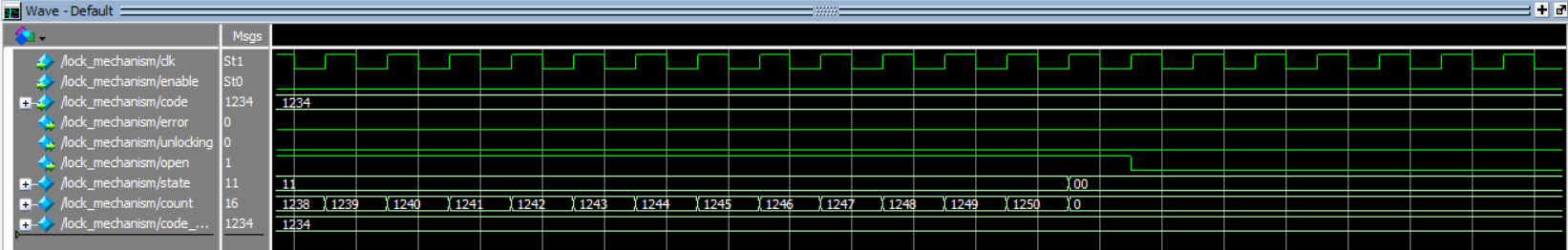
*Figure 19: Input Code waveforms part three*

**

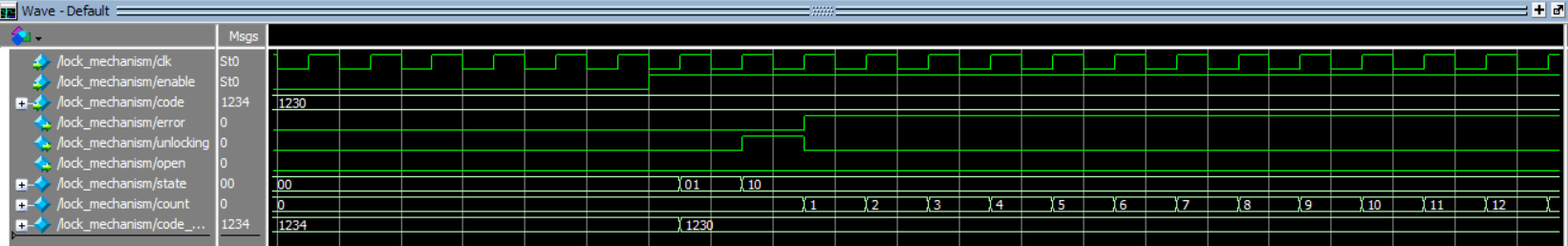
*Figure 20: Input Code waveforms part four*

**

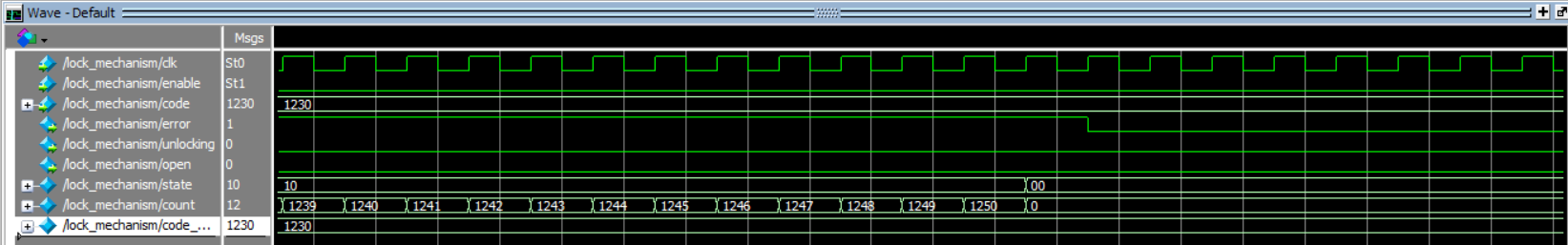
*Figure 21: Lock Mechanism waveforms part one*

**

*Figure 22: Lock Mechanism waveforms part two*

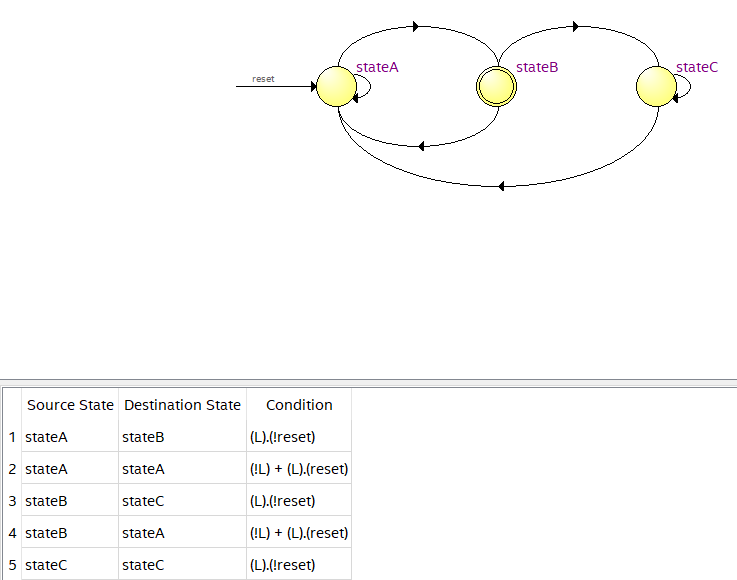
**

*Figure 23: Lock Mechanism waveforms part three*

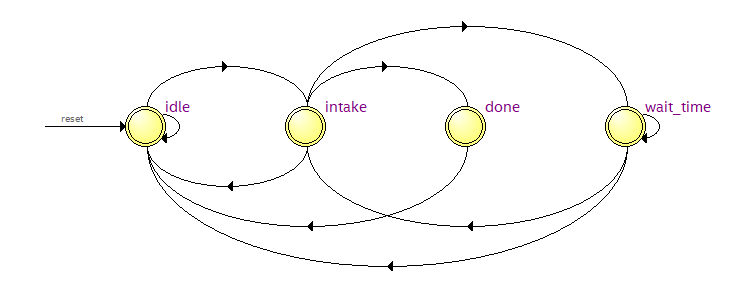
**

*Figure 24: Lock Mechanism waveforms part four*

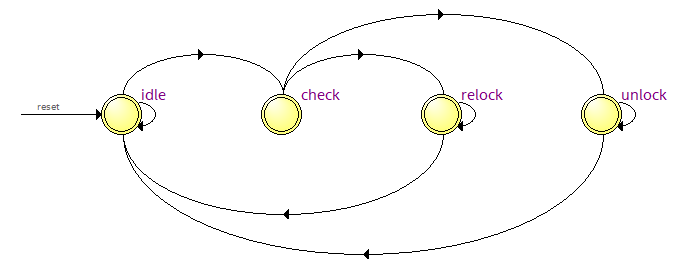
# Appendix C – State Machine Diagrams



*Figure 25: Lockout State Diagram*

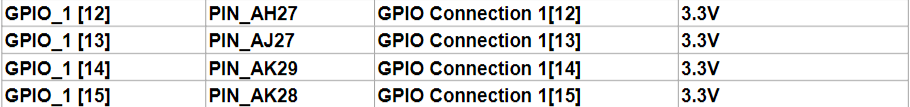
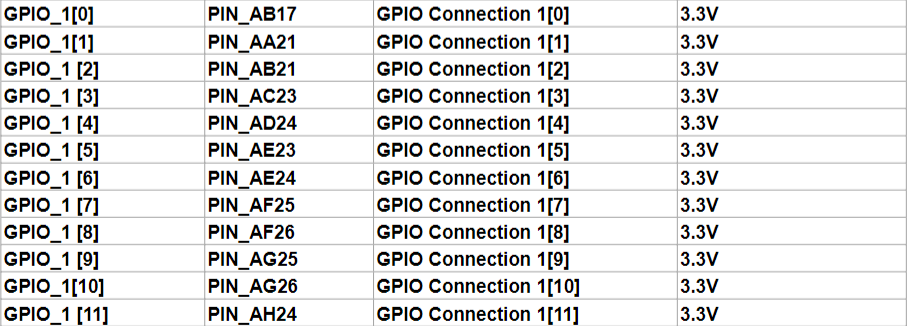
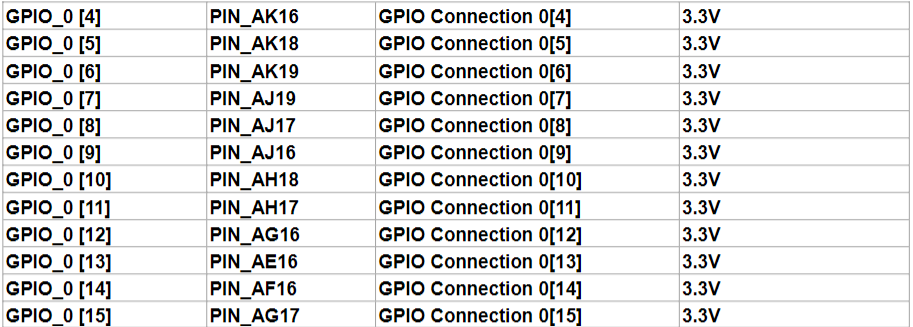
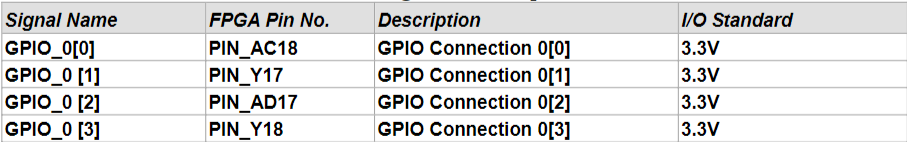
**

*Figure 26: Input Code State Diagram*

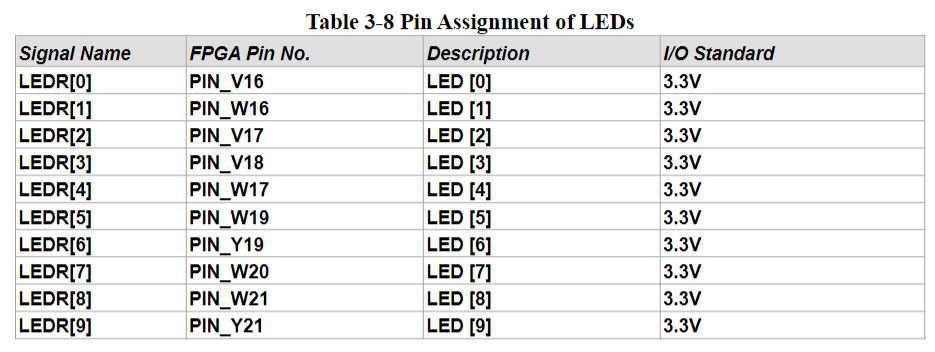
**

*Figure 27: Lock Mechanism State Diagram*

# Appendix D – Pin Tables



*Figure 28: Relevant GPIO Pin Assignment table from DE1-SoC User Manual*



*Figure 29: LED Pin Assignment table from DE1-SoC User Manual*

# Table of Figures

[Figure 1: Hardware Connection Diagram](#_3dy6vkm) 3

Figure 2: GPIO pinout diagrams from DE1-SoC User Manual 4

[Figure 3: Hierarchical Block Diagram](#_3dy6vkm) 4

Figure 4: Clock Divider RTL Diagram 14

[Figure](#_3dy6vkm) 5: Lockout RTL Diagram14

Figure 6: Keypad Decode RTL Diagram 14

[Figure 7:](#_3dy6vkm) Input Code RTL Diagram 15

Figure 8: Lockout Mechanism RTL Diagram 15

[Figure 9:](#_3dy6vkm) Seven Segment Display RTL Diagram 16

Figure 10: Top Level RTL Diagram 16

[Figure](#_3dy6vkm) 11: Seven Segment Display Waveforms part one 17

Figure 12: Seven Segment Display Waveforms part two 17

[Figure 13:](#_3dy6vkm) Seven Segment Display Waveforms part three 17

Figure 14: Seven Segment Display Waveforms part four 17

[Figure 15:](#_3dy6vkm) Keypad Decoder Waveforms part one 17

Figure 16: Keypad Decoder Waveforms part two 18

Figure 17: Input Code Waveforms part one 18

Figure 18: Input Code Waveforms part two 18

Figure 19: Input Code Waveforms part three 18

Figure 20: Input Code Waveforms part four 19

Figure 21: Lock Mechanism Waveforms part one 19

Figure 22: Lock Mechanism Waveforms part two 19

Figure 23: Lock Mechanism Waveforms part three 19

Figure 24: Lock Mechanism Waveforms part four 19

Figure 25: Lockout State Diagram 20

Figure 26: Input Code State Diagram 20

Figure 27: Lock Mechanism State Diagram 21

Figure 28: Relevant GPIO Pin Assignment Table from DE1-SoC User Manual 22

Figure 29: LED Pin Assignment table from DE1-SoC User Manual 23

# References

[1] *DE1-SoC User Manual* (2014). Accessed: January 28, 2023. [Online]. Available: https://inst-fs-iad-prod.inscloudgate.net/files/3607afc4-522a-4f44-a5c1-1580f22dd310/DE1-SoC\_User\_manual.pdf?download=1&token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2NzQxOTc2NTcsInVzZXJfaWQiOiIxMjgzOTAwMDAwMDAwMzc5MjYiLCJyZXNvdXJjZSI6Ii9maWxlcy8zNjA3YWZjNC01MjJhLTRmNDQtYTVjMS0xNTgwZjIyZGQzMTAvREUxLVNvQ19Vc2VyX21hbnVhbC5wZGYiLCJqdGkiOiI4YzdhYjE1ZC03NTY5LTQ1YmUtYTQ4NC0xNzc4YjZmZjI1N2MiLCJob3N0Ijoib2l0Lmluc3RydWN0dXJlLmNvbSIsIm9yaWdpbmFsX3VybCI6Imh0dHBzOi8vYTEyODM5LTMwMjY0ODguY2x1c3RlcjkxLmNhbnZhcy11c2VyLWNvbnRlbnQuY29tL2NvdXJzZXMvMTI4Mzl-MTc2OTUvZmlsZXMvMTI4Mzl-.rAUBUNddvg1bZIEEdqQl46VLR4ma64u9MRfpNPFygfuL28WGrF8huwyLI-6aA0d8GSLMLjCmXed53\_VN1RCaog

[2] *ALTERA Cyclone V SoC Development & Education Board (DE1-SoC)* (2014). Accessed: January 28, 2023. [Online]. Available: https://oit.instructure.com/courses/17695/files/3026486/download?download\_frd=1