Digital Keypad Access Lock System

Erkhembileg Ariunbold U1539956

Nene Ogawa U1522054

5, December 2025

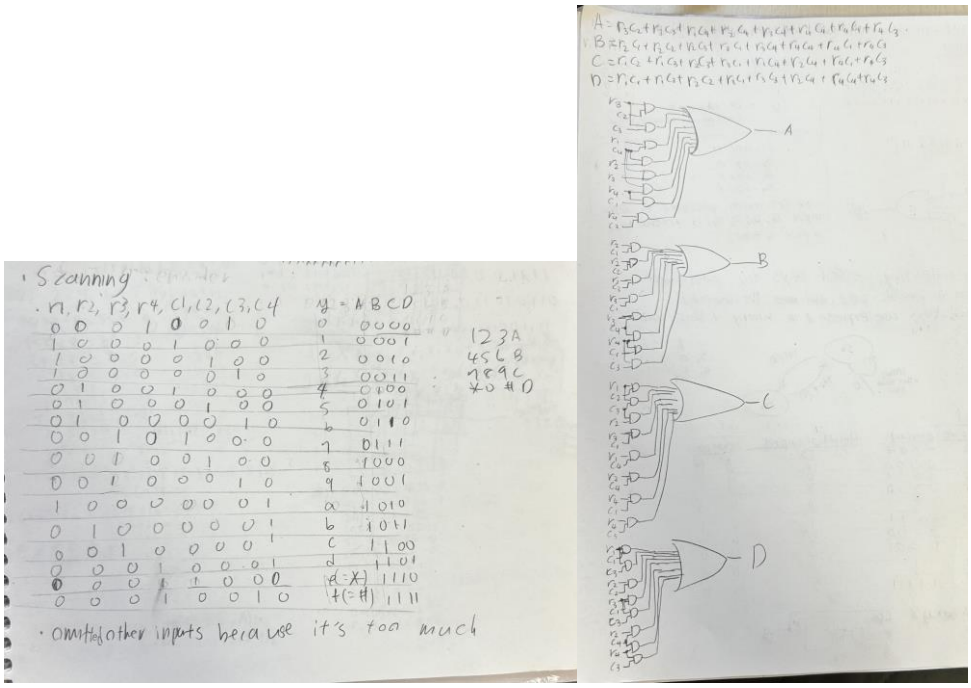
Introduction:

The project goal is to create a digital keypad access lock on an FPGA that has user input of a 5-digit PIN typed from a 4x4 matrix keypad and controls LEDs to indicate if the pin was correct or not. The correct pin is the default, and the code is written based on that pin. If the pin matches, the blue LED turns on (unlocked); otherwise, the red LED turns on (still locked).

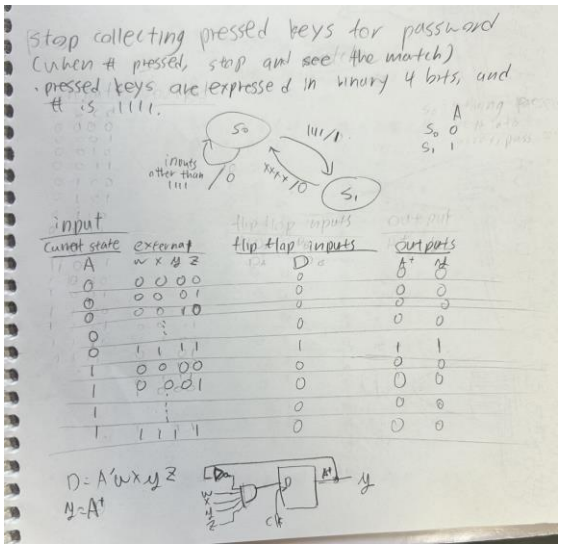
Design of this project:

When the user enters a passcode and "#," the system proceeds to compare the input to the correct passcode.

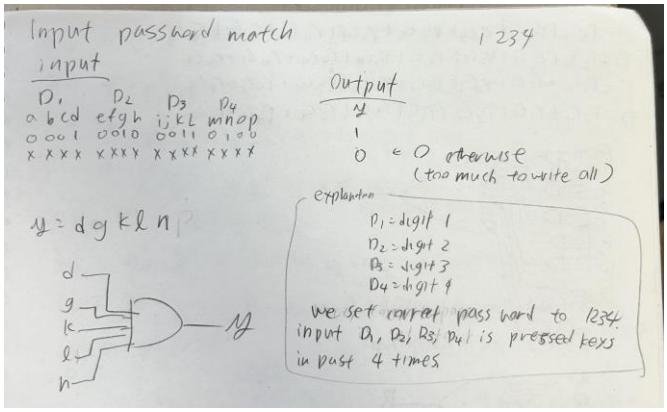
This whole system uses various logic of combination circuits and sequential circuits, but it can be divided into 4 parts. First, column-wise scanning. Here, the logic of the combination circuit is used, where when one column gets voltage from one row, the pressed key can be determined. This is expressed in the pictures below. Here, input is rows and columns, where 1 indicates that they have voltage. The output is binary 4-bit numbers that correspond to keys.



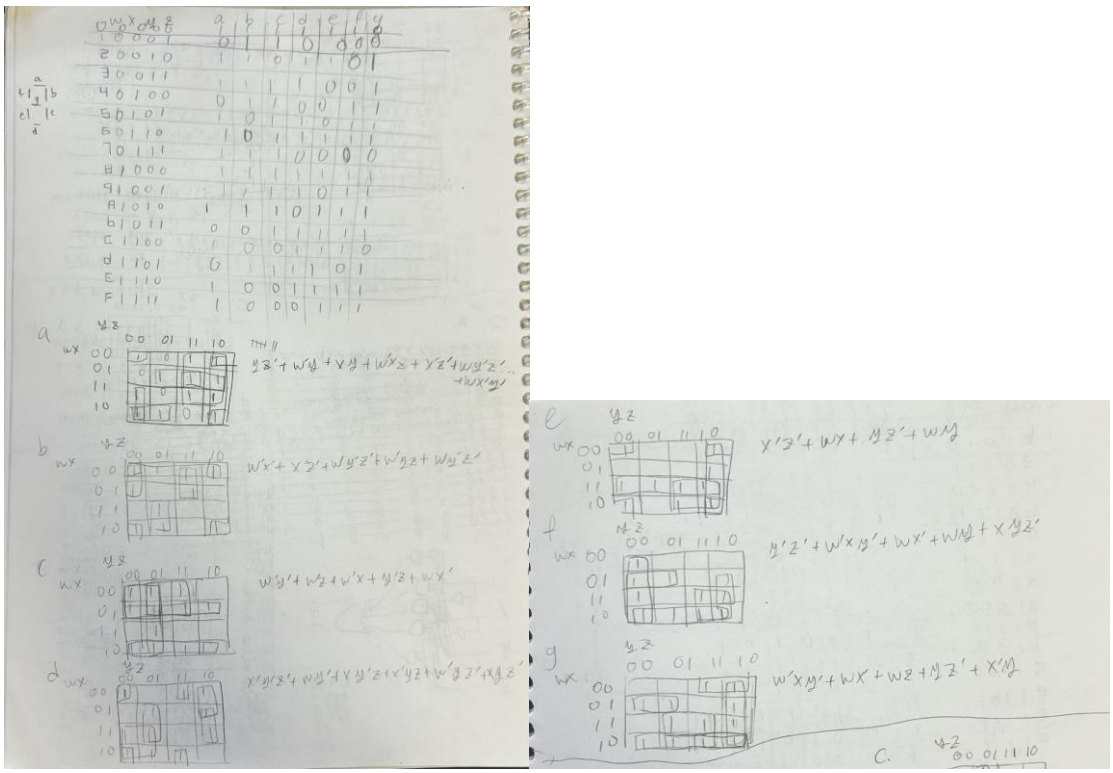
Next, the logic for the state to change from infinitely collecting the input numbers to stopping collecting and seeing a match with the passcode, and after checking, back to an infinite loop of collecting input numbers. From the code, # would stop the infinite collecting session. From this, this logic is a sequential circuit as in the picture below.



Next, logic for checking if the input matches the passcode. Here, input is 4 digits of a number, where each digits are expressed in 4 bits of binary. Output is 1 if the input matches the pass code and 0 otherwise. This logic is a combination circuit as shown below.



Lastly, a combination circuit for a 7-segments display. 7 segments display was used to show which keys were pressed. In this picture, the circuit diagram is not included because it would be too much and meaningless, since the complexity would make it critically difficult to read.



Implementation:

Keypad scanning:

Figure 1: 4×4 Keypad Mechanism

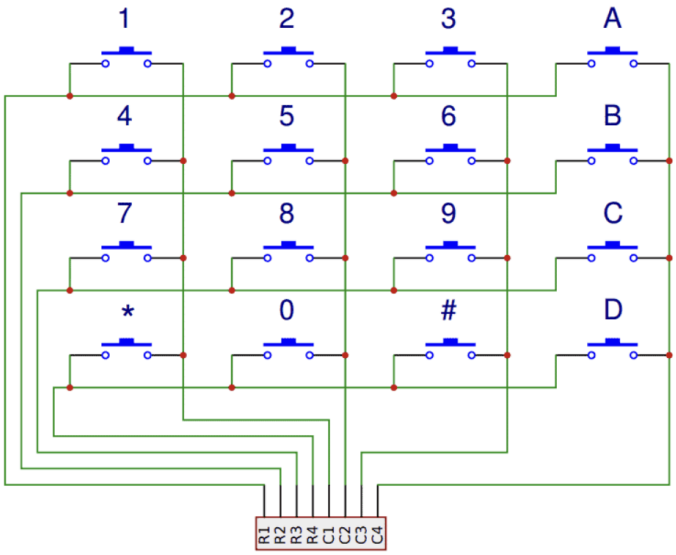
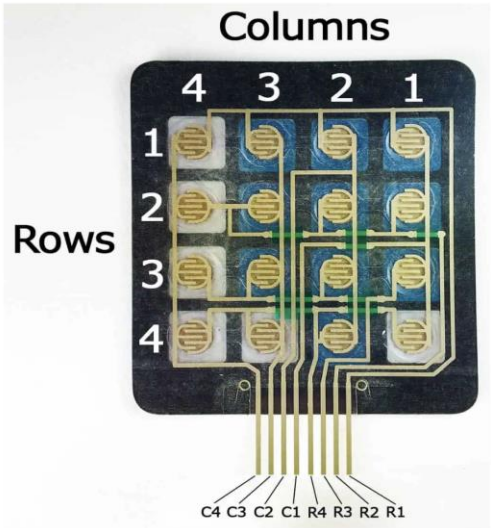


Figure 2: Internal Mechanism of the 4×4 Keypad (Back View)



The images above illustrate the working mechanism of a 4×4 keypad, where each key operates as a simple switch. When a key is not pressed, its switch remains open, preventing any signal from passing through. When a key is pressed, the switch closes and allows the signal to flow. In this project, we use a column-wise scanning method. Columns C1–C4 act as GPIO outputs from the FPGA, supplying voltage and remaining HIGH at all times. Rows

R1–R4 serve as inputs that detect the returning signals. For example, in ***Figure 1***, when the "1" key is pressed, C1 stays active, and the closed switch causes R1 to become active as well. By detecting this C1–R1 combination, we can determine that the number 1 was pressed and further decode the value in the FPGA.

Password storage and display:

Each time a key on the keypad is pressed, the system converts that key into a 4-bit value representing the corresponding digit. These values are stored one by one into a passcode buffer that holds five entries, meaning the system can register a full 5-digit password, with each digit occupying 4 bits of storage.

At the same time, the most recently pressed digit is sent to the seven-segment display so the user receives immediate visual feedback about which key was entered. This same digit is also added to the passcode input sequence, gradually building the full password as each key is pressed.

In this setup, the display and password storage operate together: the display shows the user's input in real time, while the internal register array collects each digit to form the complete passcode for later verification.

Timing adjustment:

When getting the password from the user, it is important to include a debouncing period because mechanical key switches tend to generate multiple rapid on/off signals when pressed. Without this, the FPGA could register a single key press as multiple inputs, leading to errors in the stored password. Therefore, a controlled scanning rate or delay is implemented to ensure that each key press is registered only once, giving the user enough time to press each key and allowing the system to process the input reliably. This timing adjustment ensures accurate password entry and prevents false readings.

Passcode comparison:

When the '#' key is pressed, the system interprets this as the end of the password entry process and initiates the comparison between the entered passcode and the predefined static password. This comparison occurs synchronously with the positive edge of the clock to ensure reliable operation within the FPGA. If the entered password exactly matches the stored static password, a PIN_AA2 provides a high voltage signal (3.3 V) that lights up a green LED, indicating successful entry. Conversely, if the password is incorrect, PIN_AB2 is

activated (3.3V), turning on a red LED to indicate failure. At any given time, only one of these LEDs will be illuminated, giving the user clear and immediate feedback about the result of their input.

Reset:

There is also code for resetting the system, and a reset is necessary to try another passcode input because the variables are initialized only during the reset condition. The reset occurs when the key ‘A’ is pressed. When ‘A’ is pressed, the if statement triggers the reset condition instead of storing the key in the passcode array. After the next positive clock edge, the password array is cleared. During reset, the variables used to store and validate the passcode are reinitialized, and the output voltage is also cleared.

Result:

Pressed keys correctly showed up in the 7 segments display for all the numbers, 0 to 9. The voltages come out from the output pins as expected, and red and green LEDs lights up as expected. Tens of generic pass codes were tested, and when the same pass code is pressed, green LED glowed, and any wrong pass codes glowed red LED. It also works as expected(glows red LED) when pressing ‘#’ with fewer digits input or more digits input, than correct pass code’s digits. FPGA’s built in LED(r4) lights up when the it is ready for next input number(debouncing), and before that LED turns on, pressing any key does not display new number in 7 segments display or count that pressed number toward the passcode, proving that the delay works as expected.

Reference:

[1] Thomas Havy, “SimpleDPCM,” GitHub, November 15, 2025 [Online]. Available: Th-Havy/SimpleDPCM: This repository contains an example of Differential Pulse-Code Modulation (DPCM) written in MATLAB.

[1] K. Pattabiraman, “How to set up a keypad on an Arduino,” Circuit Basics, https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/ (accessed Nov. 29, 2025).

[1] Rsp, “Tutorial: FPGA boards#,” Tutorial: FPGA Boards - IoT Engineering Education, https://iot-kmutnb.github.io/blogs/teaching/digital_logic_lab_2025-1/intro_fpga_boards/

(accessed Nov. 29, 2025).