DIGITAL LOCK

A digital lock designed using VHDL
Created By : Basant Saad Eldin

The system is a digital lock with keypad input ranging from 0 to 9.

A dedicated button located inside the house enables the user to change the secret code

lets to navigate at the VHDL 🛳️:

🔐 Digital Lock System - VHDL Design Overview

This VHDL code defines a **digital lock** system using a **keypad input**, with functionality for:

- Verifying a **password** ,
- Allowing the user to **change the password**,
- **Locking/unlocking** based on correct input,
- Tracking **failed attempts** and disabling the system after too many failures

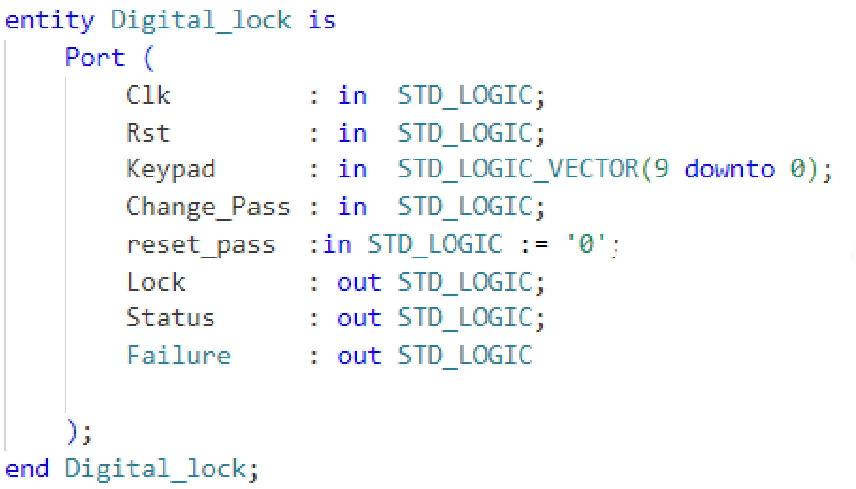
🧪 Initial Values

- **Default stored password: "2345" (hex: x"2345" = 0010 0011 0100 0101).**
- **The system starts in a locked state (Lock = '0').**

Code : Drive

VHDL of Main code and Testbench

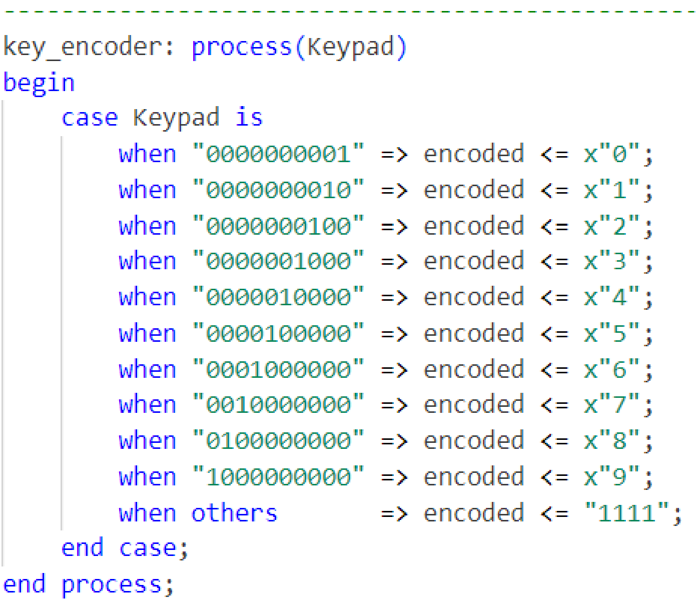
📦 Entity Ports



```
entity Digital_lock is
    Port (
        Clk         : in  STD_LOGIC;
        Rst         : in  STD_LOGIC;
        Keypad      : in  STD_LOGIC_VECTOR(9 downto 0);
        Change_Pass : in  STD_LOGIC;
        reset_pass  :in STD_LOGIC := '0';
        Lock        : out STD_LOGIC;
        Status      : out STD_LOGIC;
        Failure     : out STD_LOGIC

    );
end Digital_lock;
```

Signal	Direction	Description
Clk	in	⌚ Clock signal.
Rst	in	®Reset signal.
Keypad	in	⌨️10-bit input representing the keypad (1-hot encoded: one bit per digit 0–9).
Change_Pass	in	🔄When high, enables password change mode.
Lock	out	🔐Indicates whether the lock is opened (1) or closed (0).
Status	out	Indicates a successful login. ▶Blue LED
Failure	out	Indicates system lock due to repeated failures. ❌ Red LED
reset_pass	inout	🧹For clear Entered Storge , Rewrite Password

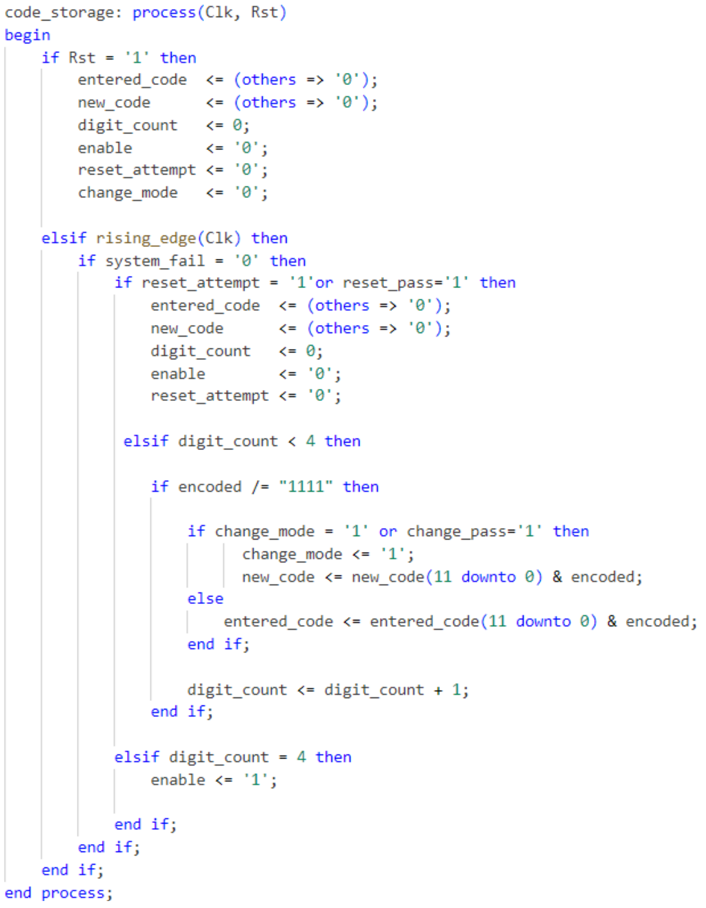
🧠 Main Functional Blocks



```
------------------------------------------------
key_encoder: process(Keypad)
begin
    case Keypad is
        when "0000000001" => encoded <= x"0";
        when "0000000010" => encoded <= x"1";
        when "0000000100" => encoded <= x"2";
        when "0000001000" => encoded <= x"3";
        when "0000010000" => encoded <= x"4";
        when "0000100000" => encoded <= x"5";
        when "0001000000" => encoded <= x"6";
        when "0010000000" => encoded <= x"7";
        when "0100000000" => encoded <= x"8";
        when "1000000000" => encoded <= x"9";
        when others       => encoded <= "1111";
    end case;
end process;
```

🔢 1. Keypad Encoder

- Converts the 10-bit keypad input into a Hexadecimal Number (4-bit binary) value (0–9).
- If no key is pressed or multiple keys are pressed, sets encoded value to "1111" (invalid).



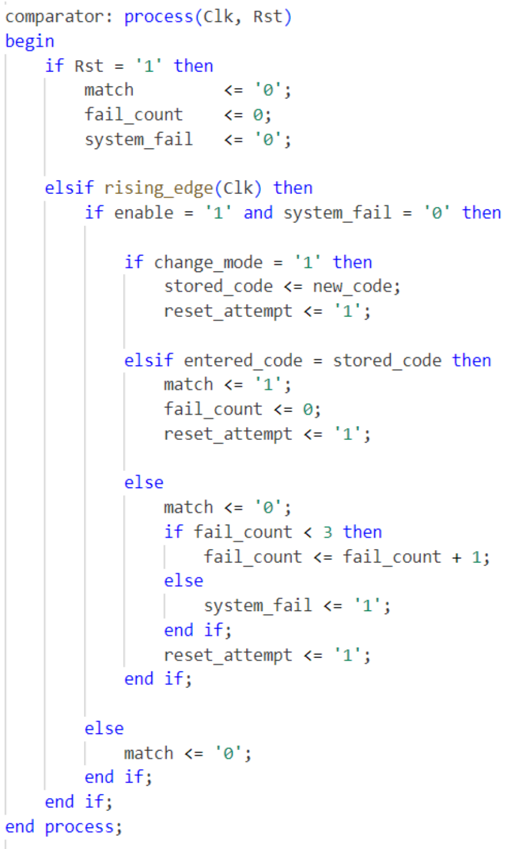
```
code_storage: process(Clk, Rst)
begin
    if Rst = '1' then
        entered_code  <= (others => '0');
        new_code      <= (others => '0');
        digit_count   <= 0;
        enable        <= '0';
        reset_attempt <= '0';
        change_mode   <= '0';

    elsif rising_edge(Clk) then
        if system_fail = '0' then
            if reset_attempt = '1'or reset_pass='1' then
                entered_code  <= (others => '0');
                new_code      <= (others => '0');
                digit_count   <= 0;
                enable        <= '0';
                reset_attempt <= '0';

             elsif digit_count < 4 then

                if encoded /= "1111" then

                    if change_mode = '1' or change_pass='1' then
                          change_mode <= '1';
                          new_code <= new_code(11 downto 0) & encoded;
                    else
                        entered_code <= entered_code(11 downto 0) & encoded;
                    end if;

                    digit_count <= digit_count + 1;
                end if;

            elsif digit_count = 4 then
                enable <= '1';

            end if;
        end if;
    end if;
end process;
```

💾 2. Code Entry and Storage

- Uses a shift register logic to build a 4-digit code:
    - If Change_Pass = '1', the user is entering a new password.
    - Otherwise, the user is entering the current password.
- After 4 digits, it enables password comparison (enable <= '1').

✅ 3. Password Comparator & Fail Counter

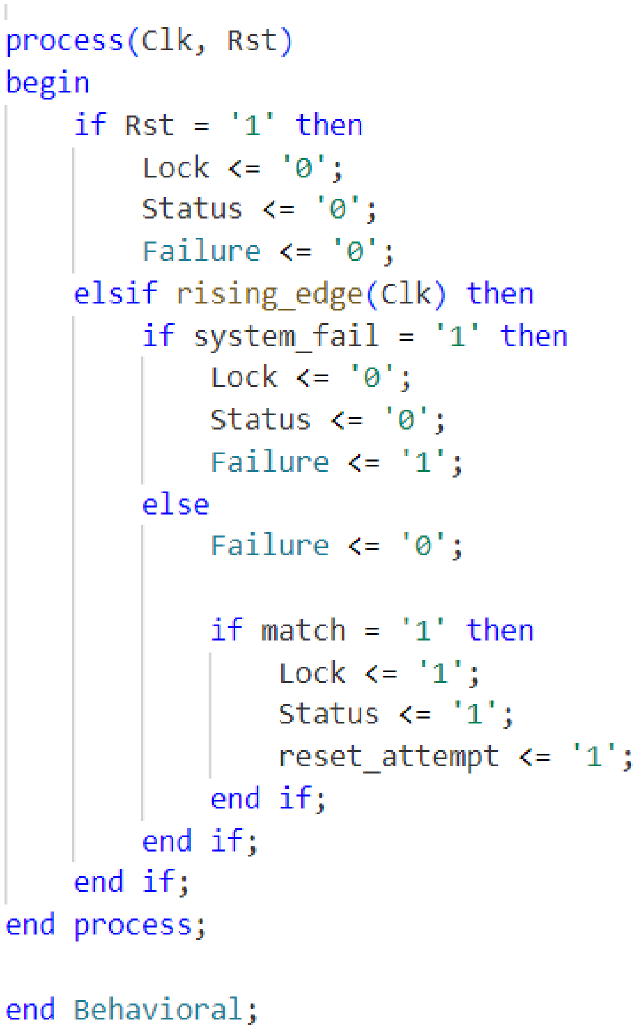
- Compares the entered password to the stored password (stored_code).
- If matched:
  - Sets match = '1', resets fail count.
  - Unlocks the system (Lock = '1').
- If incorrect:
  - Increments fail counter.
  - If 3 wrong attempts → system_fail = '1' (locks the system permanently).
- In change mode:
  - Stores the new code into stored_code.



```vhdl
comparator: process(Clk, Rst)
begin
    if Rst = '1' then
        match        <= '0';
        fail_count   <= 0;
        system_fail  <= '0';

    elsif rising_edge(Clk) then
        if enable = '1' and system_fail = '0' then

            if change_mode = '1' then
                stored_code <= new_code;
                reset_attempt <= '1';

            elsif entered_code = stored_code then
                match <= '1';
                fail_count <= 0;
                reset_attempt <= '1';

            else
                match <= '0';
                if fail_count < 3 then
                    fail_count <= fail_count + 1;
                else
                    system_fail <= '1';
                end if;
                reset_attempt <= '1';
            end if;

        else
            match <= '0';
        end if;
    end if;
end process;
```

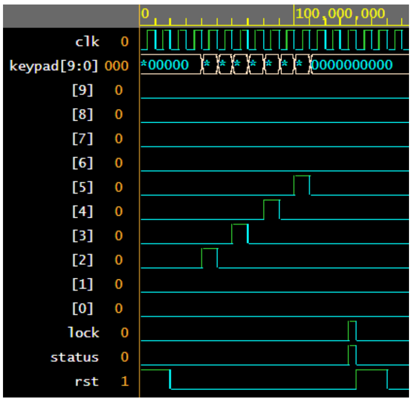
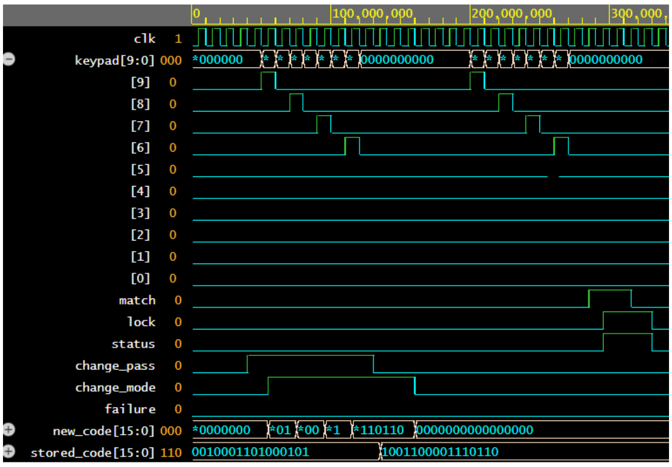
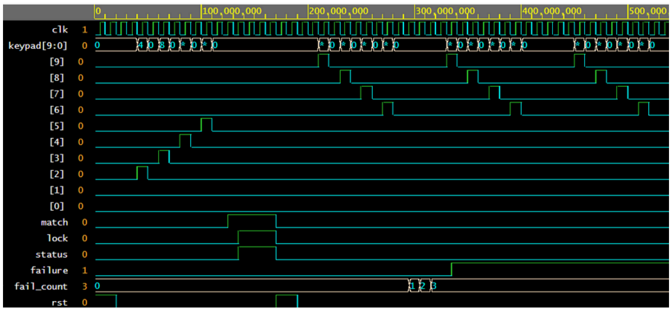
🚦 4. Output Logic

- If system is failed (system_fail = '1'):
  - Sets Failure = '1', keeps Lock = '0', and disables Status.
- If successful login:
  - Lock = '1', Status = '1'.
  - reset_attempt = '1'. for the next new enters



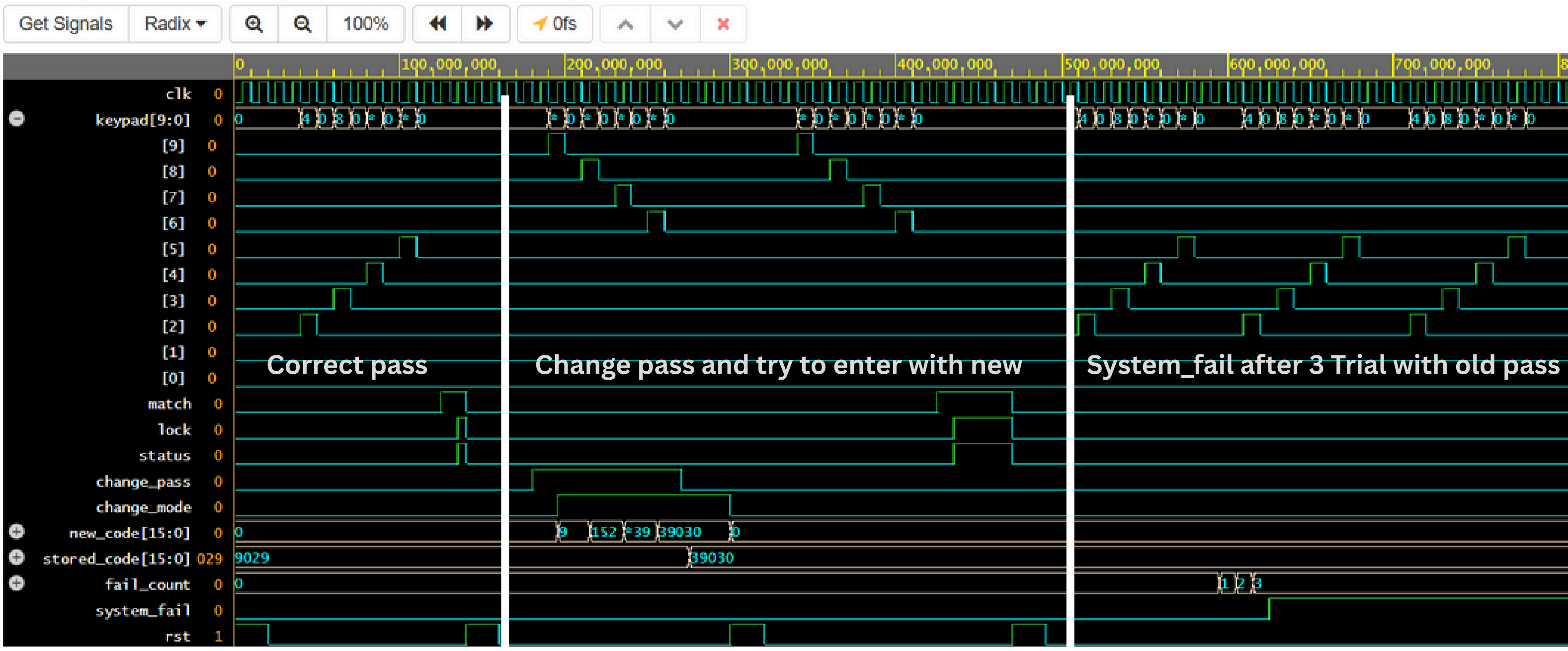
```vhdl
process(Clk, Rst)
begin
    if Rst = '1' then
        Lock <= '0';
        Status <= '0';
        Failure <= '0';
    elsif rising_edge(Clk) then
        if system_fail = '1' then
            Lock <= '0';
            Status <= '0';
            Failure <= '1';
        else
            Failure <= '0';

            if match = '1' then
                Lock <= '1';
                Status <= '1';
                reset_attempt <= '1';
            end if;
        end if;
    end if;
end process;

end Behavioral;
```

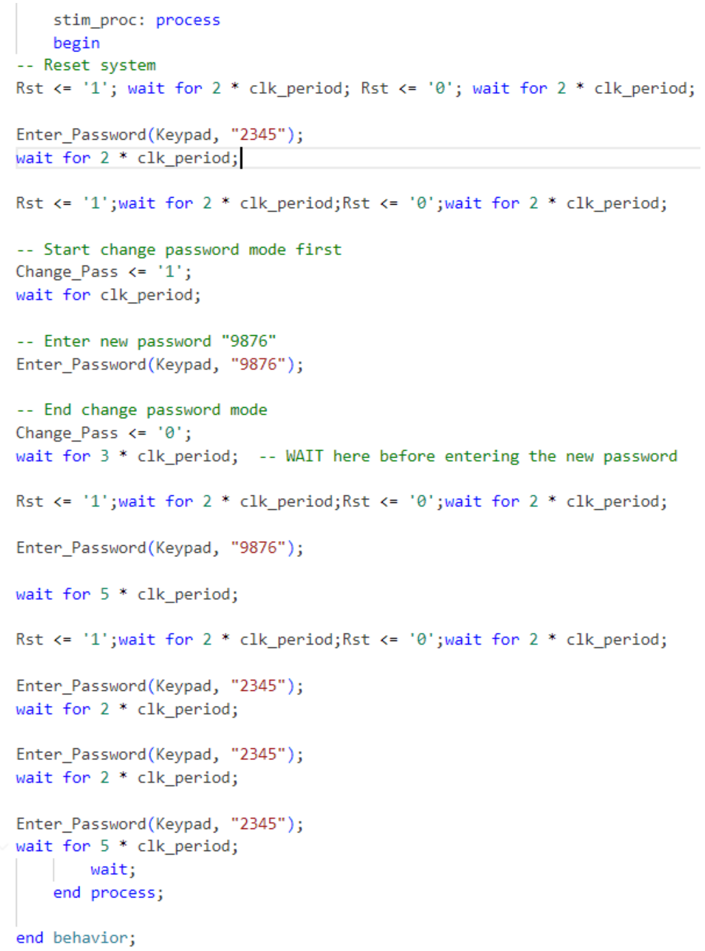
🧷 Summary of Features

Feature	Behavior
Password verification Enter 4 digits, compare to stored code , ▶ Statue	 0 100,000,000 clk 0 keypad[9:0] 000 *00000 0000000000 [9] 0 [8] 0 [7] 0 [6] 0 [5] 0 [4] 0 [3] 0 [2] 0 [1] 0 [0] 0 lock 0 status 0 rst 1
Password change Triggered by Change_Pass = '1', stores new code	 0 100,000,000 200,000,000 300,000 clk 1 keypad[9:0] 000 *000000 0000000000 0000000000 [9] 0 [8] 0 [7] 0 [6] 0 [5] 0 [4] 0 [3] 0 [2] 0 [1] 0 [0] 0 match 0 lock 0 status 0 change_pass 0 change_mode 0 failure 0 new_code[15:0] 000 *0000000 *01 *00 *1 *110110 0000000000000000 stored_code[15:0] 110 0010001101000101 1001100001110110
Lock control	Unlocks only when password matches,✅ Match
Fail counter Counts wrong attempts (max 3), then system locks , ❌SastemPstill high	 0 100,000,000 200,000,000 300,000,000 400,000,000 500,000 clk 1 keypad[9:0] 0 [9] 0 [8] 0 [7] 0 [6] 0 [5] 0 [4] 0 [3] 0 [2] 0 [1] 0 [0] 0 match 0 lock 0 status 0 failure 1 fail_count 3 rst 0
Reset	Rst = '1' resets all internal signals

🧠 Goal of the Testbench

The testbench simulates how your digital lock system behaves in several scenarios:



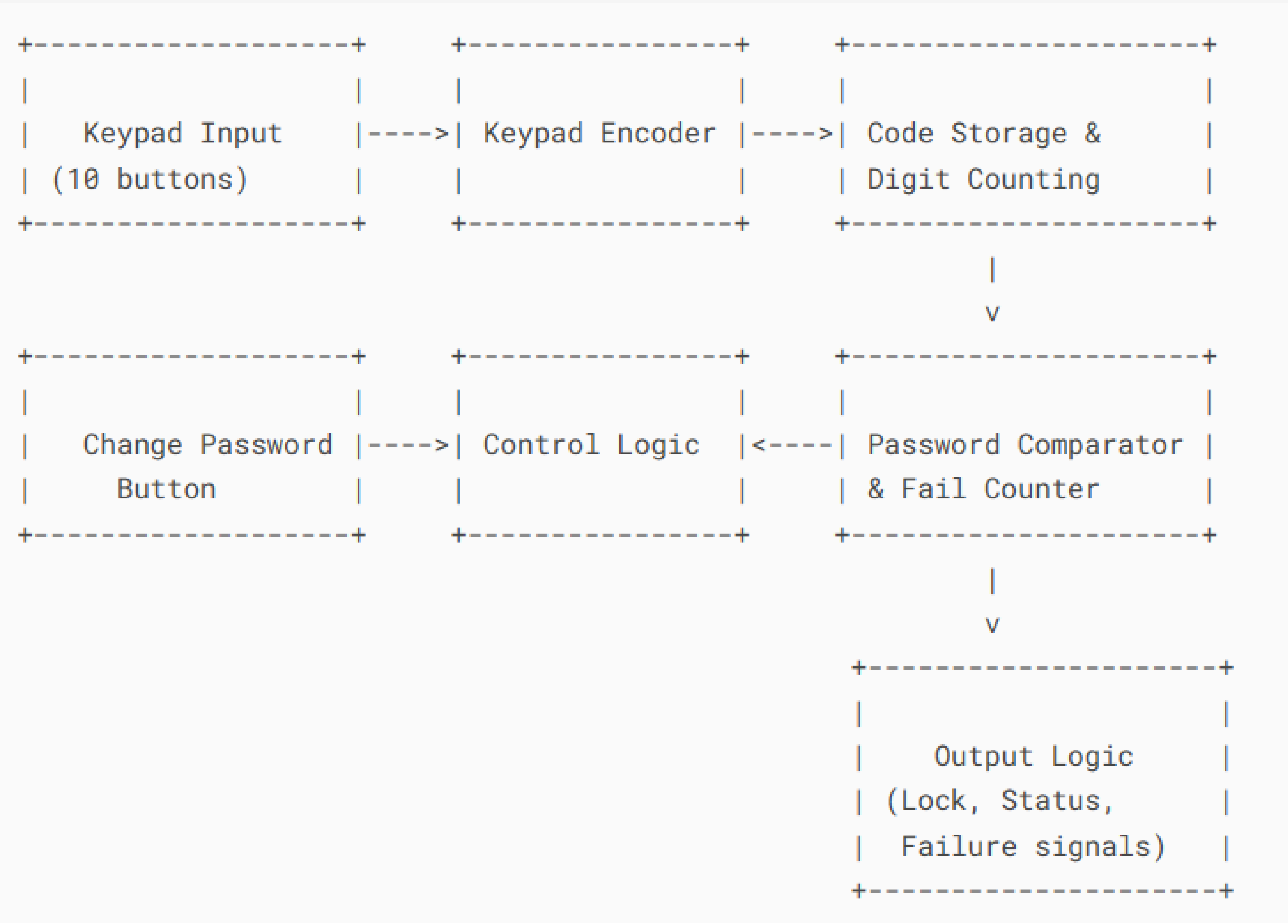


```
    stim_proc: process
    begin
-- Reset system
Rst <= '1'; wait for 2 * clk_period; Rst <= '0'; wait for 2 * clk_period;

Enter_Password(Keypad, "2345");
wait for 2 * clk_period;

Rst <= '1';wait for 2 * clk_period;Rst <= '0';wait for 2 * clk_period;

-- Start change password mode first
Change_Pass <= '1';
wait for clk_period;

-- Enter new password "9876"
Enter_Password(Keypad, "9876");

-- End change password mode
Change_Pass <= '0';
wait for 3 * clk_period;  -- WAIT here before entering the new password

Rst <= '1';wait for 2 * clk_period;Rst <= '0';wait for 2 * clk_period;

Enter_Password(Keypad, "9876");

wait for 5 * clk_period;

Rst <= '1';wait for 2 * clk_period;Rst <= '0';wait for 2 * clk_period;

Enter_Password(Keypad, "2345");
wait for 2 * clk_period;

Enter_Password(Keypad, "2345");
wait for 2 * clk_period;

Enter_Password(Keypad, "2345");
wait for 5 * clk_period;
        wait;
    end process;

end behavior;
```

1. **Entering the default correct password (2345)**
2. **Changing the password to a new one (9876)**
3. **Unlocking with the new password**
4. **Failing to unlock with the old password (2345) three times to trigger the failure LED**

Circuit and 3D Design -->

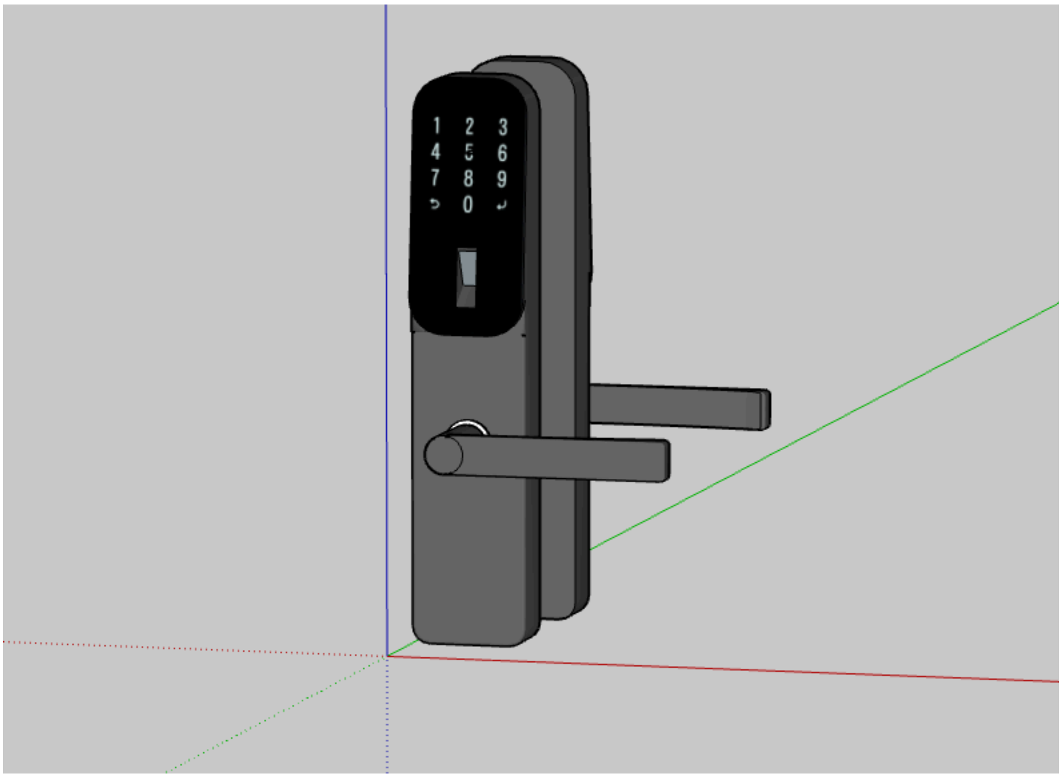
Circuit 📝

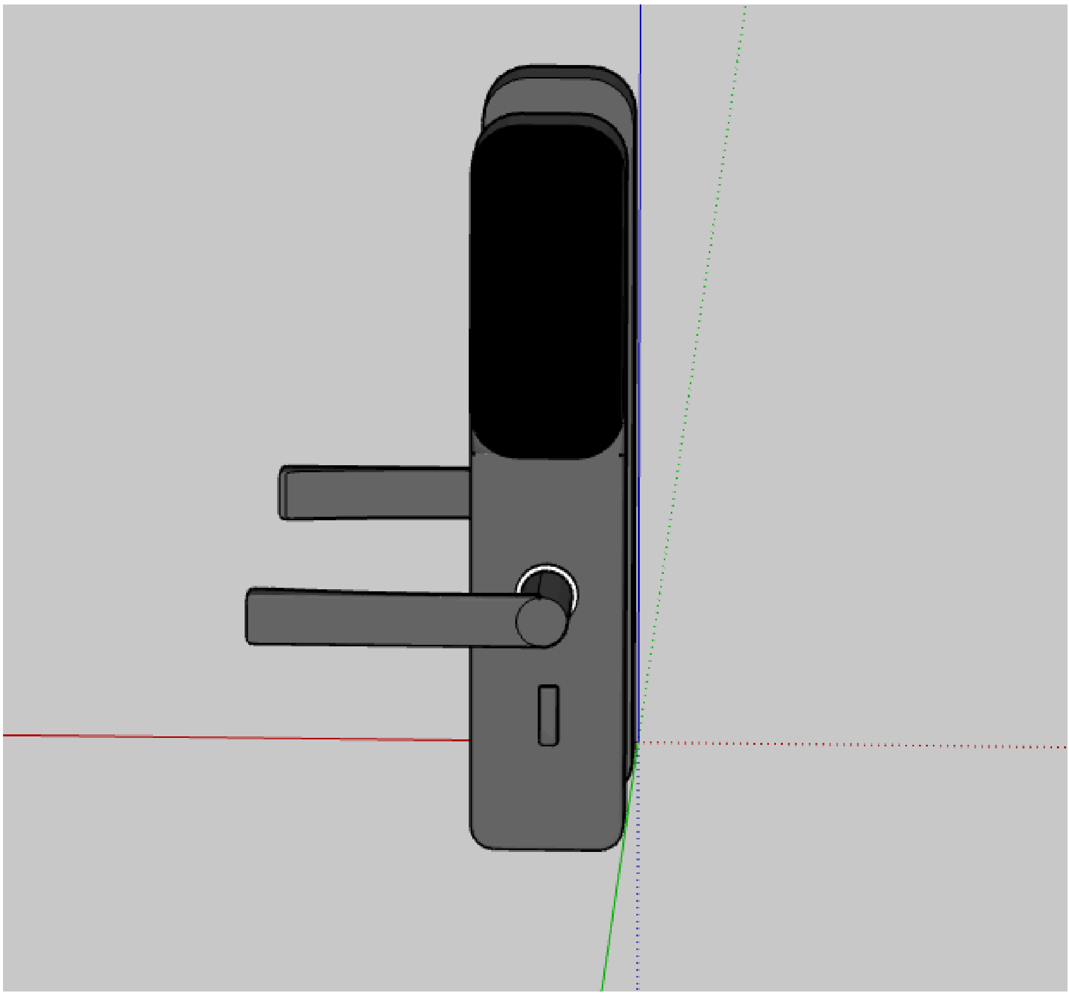


```
+-------------------+     +----------------+     +---------------------+
|                   |     |                |     |                     |
|   Keypad Input    |---->| Keypad Encoder |---->| Code Storage &      |
| (10 buttons)      |     |                |     | Digit Counting      |
+-------------------+     +----------------+     +---------------------+
                                                          |
                                                          v
+-------------------+     +----------------+     +---------------------+
|                   |     |                |     |                     |
|   Change Password |---->| Control Logic  |<----| Password Comparator |
|     Button        |     |                |     | & Fail Counter      |
+-------------------+     +----------------+     +---------------------+
                                                          |
                                                          v
                                                  +---------------------+
                                                  |                     |
                                                  |    Output Logic     |
                                                  | (Lock, Status,      |
                                                  |  Failure signals)   |
                                                  +---------------------+
```

Thank You

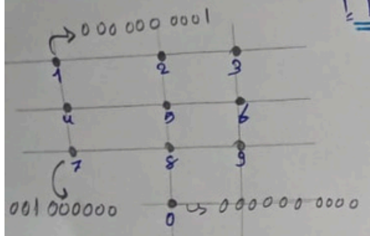
3D Printed Design 🕸️





↳ when rising edge (Clk) ↑

1- KeyPad

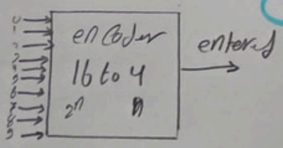


input KeyPad	output entered
0 0 0 0 0 0 0 0 0 1	X" 0" ~0000
0 0 0 0 0 0 0 0 1 0	X" 1 " ~0001
0 0 0 0 0 0 0 1 0 0	X "2" ~0010
0 0 0 0 0 0 1 0 0 0	X"3" ~0011
0 0 0 0 0 1 0 0 0 0	X"4" ~0100
0 0 0 0 1 0 0 0 0 0	X "5" ~0101
0 0 0 1 0 0 0 0 0 0	X "6" ~0110
0 0 1 0 0 0 0 0 0 0	X "7" ~0111
0 1 0 0 0 0 0 0 0 0	X "8" ~1000
1 0 0 0 0 0 0 0 0 0	X "9" ~1001

↳ Can Be encoded :-

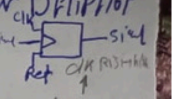
→ one Hot encoding & Prolovity encoder

- 16 to 4 that contains Butaeltan used 10 input 1 output



2- Storage & Digit Counting

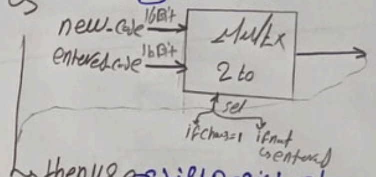
* this Part, have group of logic circuit :- [1] Register D flipflop



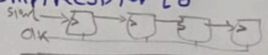
first (1) Check Reset if = 1 ⇝ Put all signals = zero

if not (2)
↳ check if system fail = '0' ⇝ to ignar to any entered

if not
↳ if (3) digital Count < 4
↳ check Valid input
↳ then use [2] multiplexer



↳ then use [3] shift Register to



output of multiplexer
→ if new-code

0000 0000 0000 0000
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

+ [1] will be

(4) Counter

0000 0000 0000 & 0010 entered if = X2"

+1 [2]

0000 0000 0010 & 0011 if = X"3"

+1 [3]

0000 0010 0011 & 0100 if = X"4"

[4]

0010 0011 0100 & 0101 if = X"5"

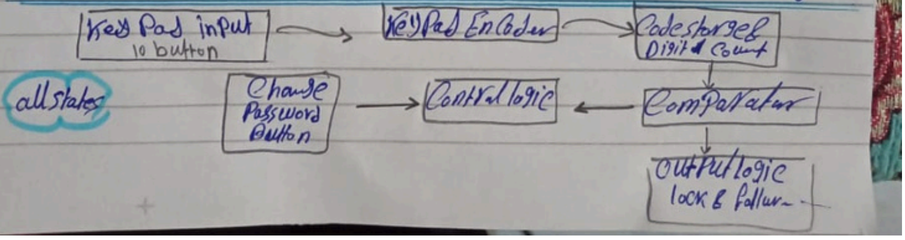
(5) Comprator
if digital_count = 4 → enable = '1'   go to comprator ↳ ⇒

3 in Comparator & fail Counter

```
1 Chear for RSt → it's all signals that used here = zero
  ifnot
  → when rising edge (Clk) ↑
      → Cheak for enable = 1 & system fail = '0'
                         → when entered are 4 digits
          → check mode
              change mode
                  storage = new code
              entered mode
                  → check if entered_code = storage_code
                      false
                          if fail_Cout ≥ 3
                              system_fail = '1'
                          f_c < 3
                              match = 0
                              f_c + 1
                      True
                          match = 1
                          fail Count = 0
                              → if there were less than 3 fail Cout before
```

4 in output logic

```
1 Check for Rst
  ifnot
  → when rising edge (clk) ↑
      if system_fail
          still lock
          failure = '1'   Red led
      ----→ elses
      if match = 1
          lock = 1        open
          Status = '1'    Blue led
          reset_attemP = '1'
              → for new entered
```



Key Pad input
10 button
KeyPad EnCoder
Code storge & Digit Count
all states
Change Password Button
Control logic
Comparator
output logic
lock & failure