A CSCI221- Logic Design Project

Digital Lock for a Safe

Team Members:
Ameer Akram 202000249
Dina Magdy 202001451
Hadeer Said 202001540
Mariam Barakat 202000210
Nada Aboulfotouh 202000046

Project Description:

Due to the ever-growing scientific technological advancements, crime rates are subsequently increasing throughout the world each day. Therefore, our team designed a simple, cheap yet secure, and reliable system of a digital lock for the safe use of logic gates, comparators, registers, and flip-flops. This system will help to prevent intruders and thieves from accessing unless they can provide a binary password that correctly matches the preset password of the safe; if the user entered an incorrect password three consecutive times, the safe will no longer accept any entries unless the user presses the reset button. The system works by comparing each bit in the passcode by its adjacent bit in the preset code; this is done using 4 XNOR gates so that if they are all the same, the output is one. Subsequently, the output of the four gates will be passed by an AND gate, so its output is 1 if all the bits match and is zero otherwise. To stop the safe from accepting any input after three unsuccessful trials, we used a NOT gate to invert the zero output of the AND gate, to count the incorrect trials, so we will be incrementing using a 2 bit synchronous counter using 2 T-flip-flops. The output for the counter is passed through a decoder to output the count of the current entry attempt, if it reaches 3 then the user should click the reset button to reset the digital lock, and be able to continue trying to open the safe. If the binary passcode was correct at any trial the lock will open safely. Asynchronous Edge detector was implemented using NOT gates and XOR gates to detect a change in input and acts as a clock for the two T flip-flops.

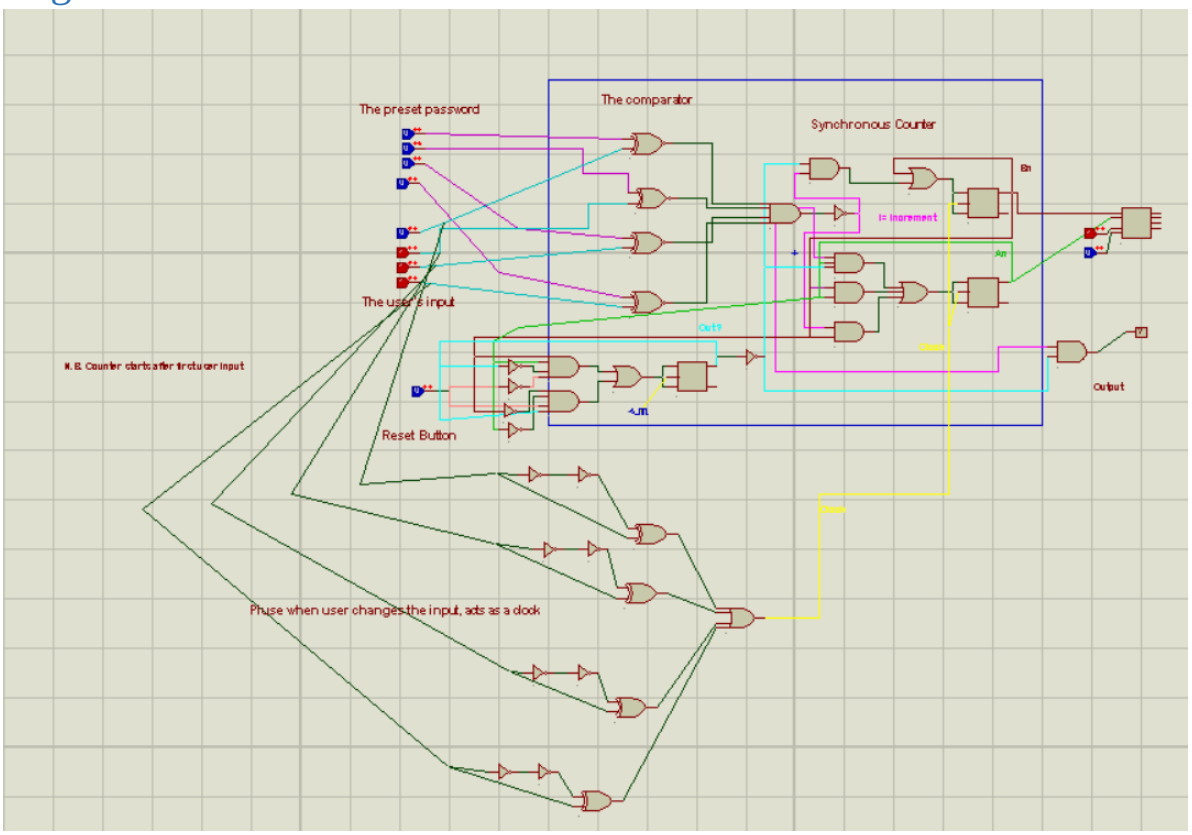
Components Used:

AND Gate (7408)	**5 EGP**
AND_3 Gate (7411)	**10 EGP**
AND_4 Gate	
OR (7432)	**5 EGP**
OR_3	
OR_4	
XOR (7486)	**10 EGP**
Quad 2-input exclusive NOR gate (4077)	**10 EGP**

NOT (7404)	**5 EGP**
DECODER_2_4 (7448)	**15 EGP**
JKFF (7476)	**20 EGP**

Proteus Simulation:

Logic Circuit



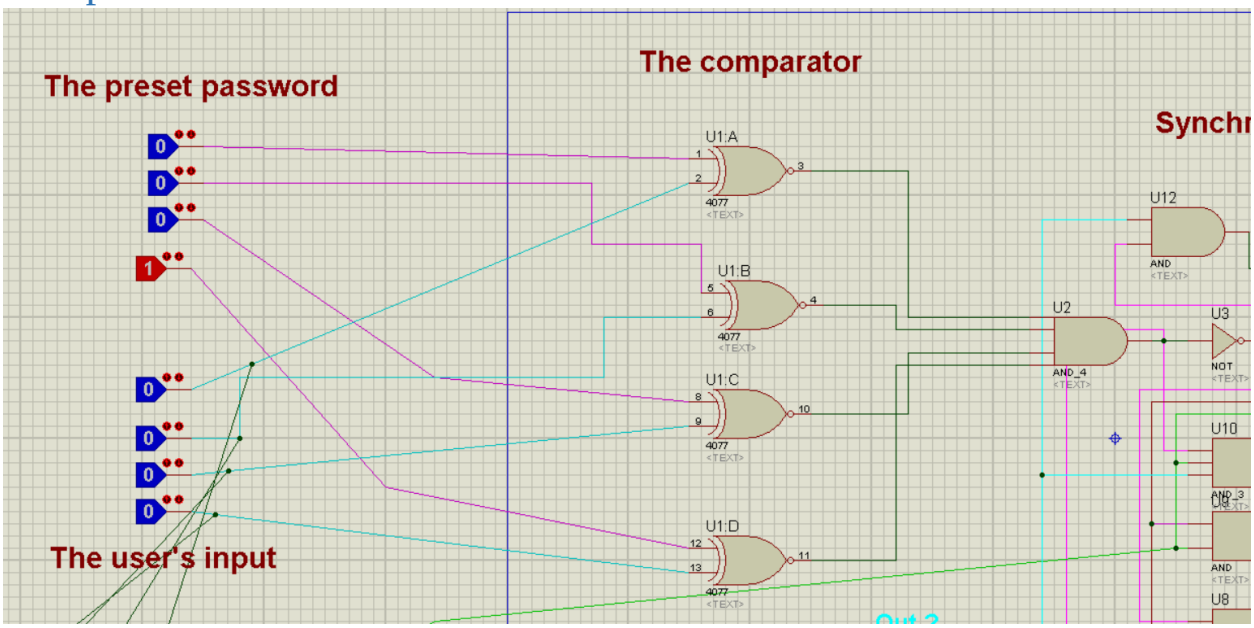
Parts:

1. Comparator.
2. Synchronous Counter.
3. Asynchronous Edge detector acting as our clock.
4. A partition that activates the counter when out of range, by user clicking the switch, then inputs his data, and then closes it again.
5. Output.

Note:

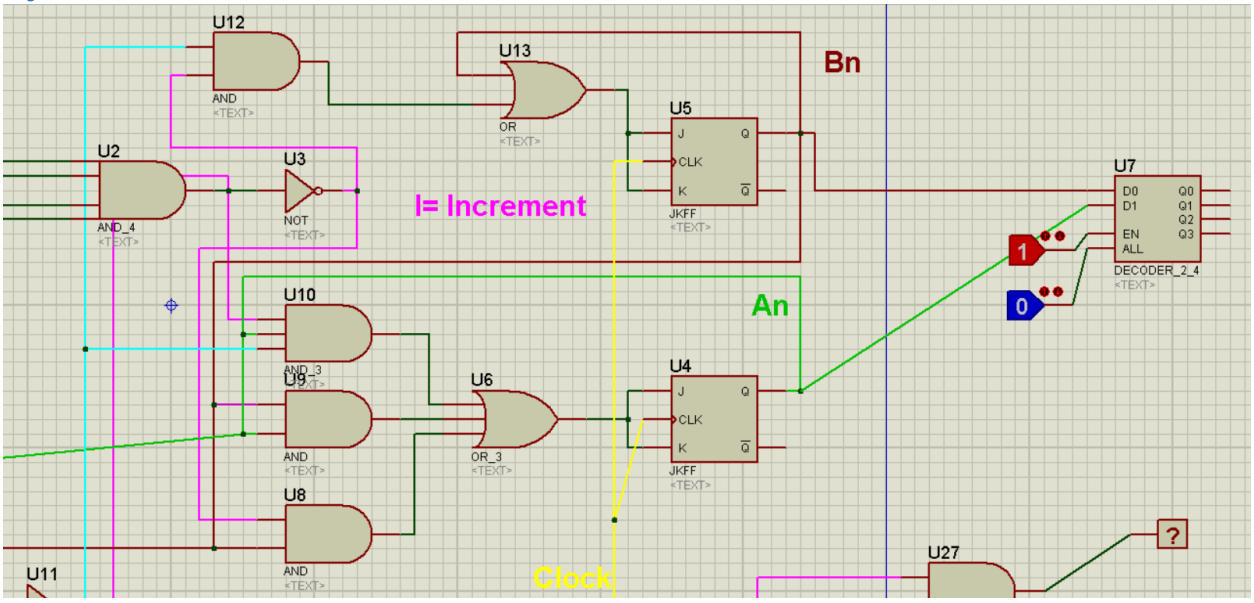
- All wires are color-coded, search for the variable with the same color.
- Counting starts as the user inputs new data.

Comparator



- The output of the comparator is 1 if the passwords are not the same.
- The output of the comparator is 0 if the passwords are the same.
- It is opposite to intuition, but we needed to count the incorrect trials, so the one is used for incrementing.

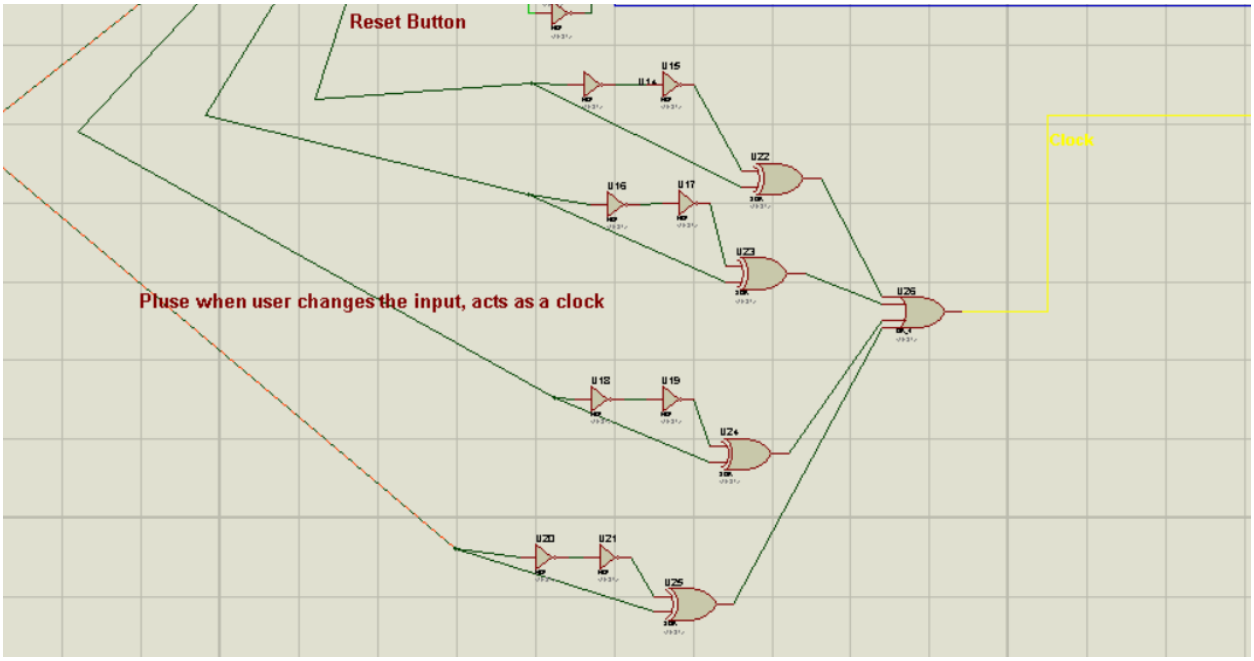
Synchronous Counter



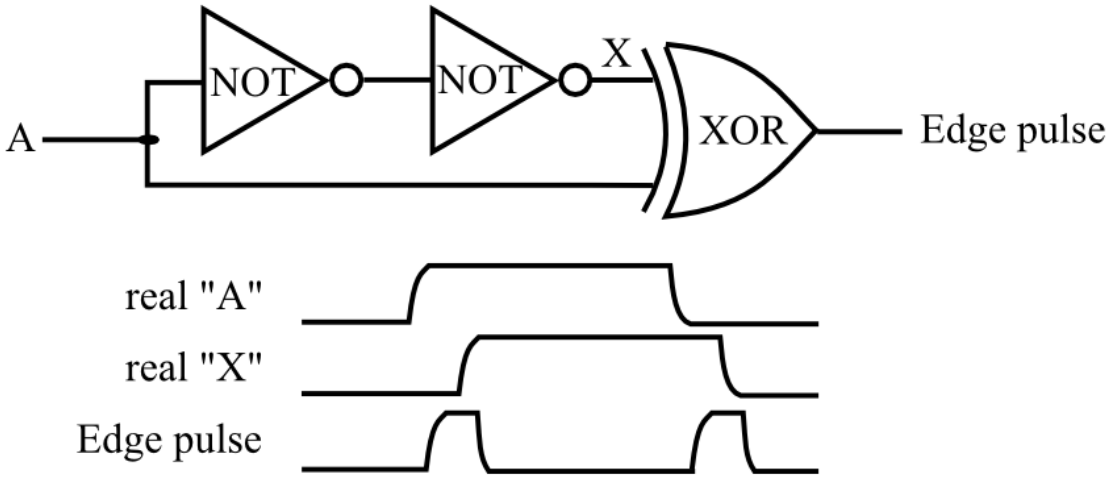
- This uses 2 JK flip-flops acting as a T flip flop.
- The decoder is used to output the count we are in now.
- It only increments if we have incorrect input, and we are out of range. Other than that the counter will be 00.

- The turquoise wire is the **Out** variable that is used to calculate whether we are out of range and if out of range, whether the reactivation switch is on or not.

Asynchronous Edge detector - Clock

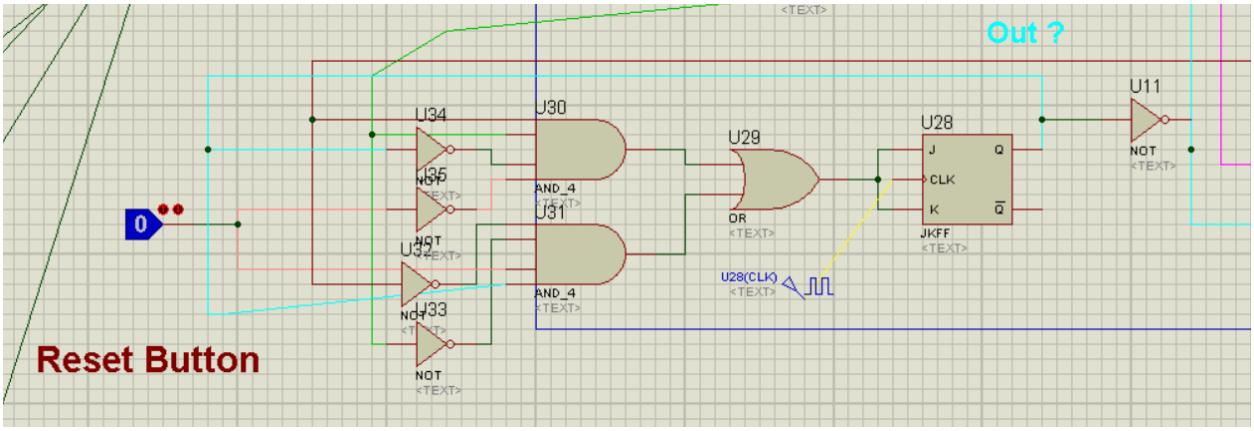


- The wires are coming from the user input.
- This double NOT is used for delay.
- The XOR is used to detect a change in input.



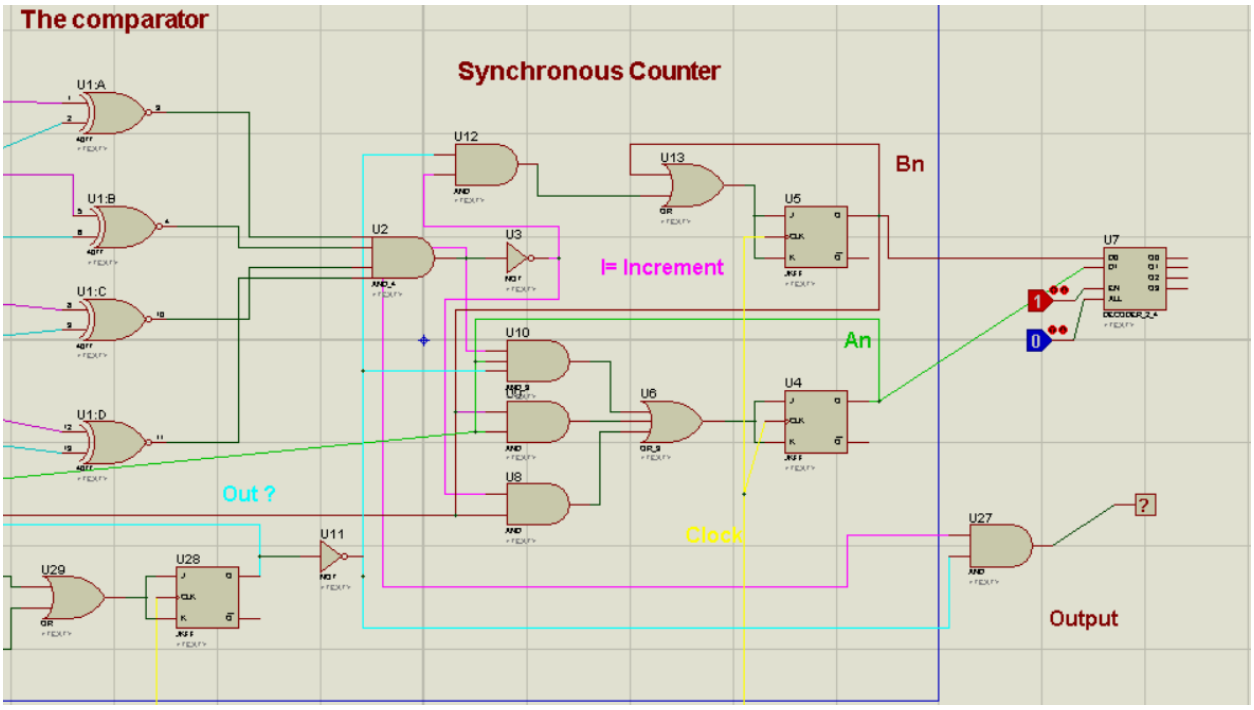
https://stackoverflow.com/questions/55327429/how-to-create-an-asynchronous-edge-detector-in-vhdl

Switch



- The output of the variable Out is 1 if we are out of range and the user didn't use the reset button, the output is 0 if still, we are in range.
- Used a clock with a frequency of 10, to update fast, as the output of out is required in the synchronous counter.

Output

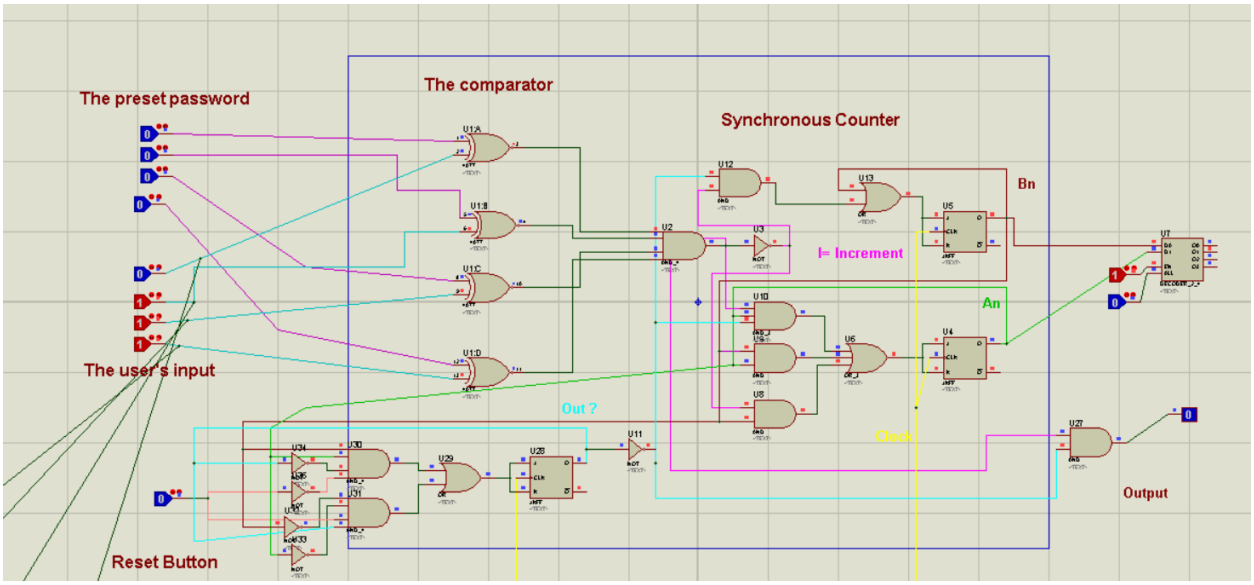


- This outputs 1 if the user inputs the correct password and we are still within the range, other than that, it outputs 0.

Demonstration

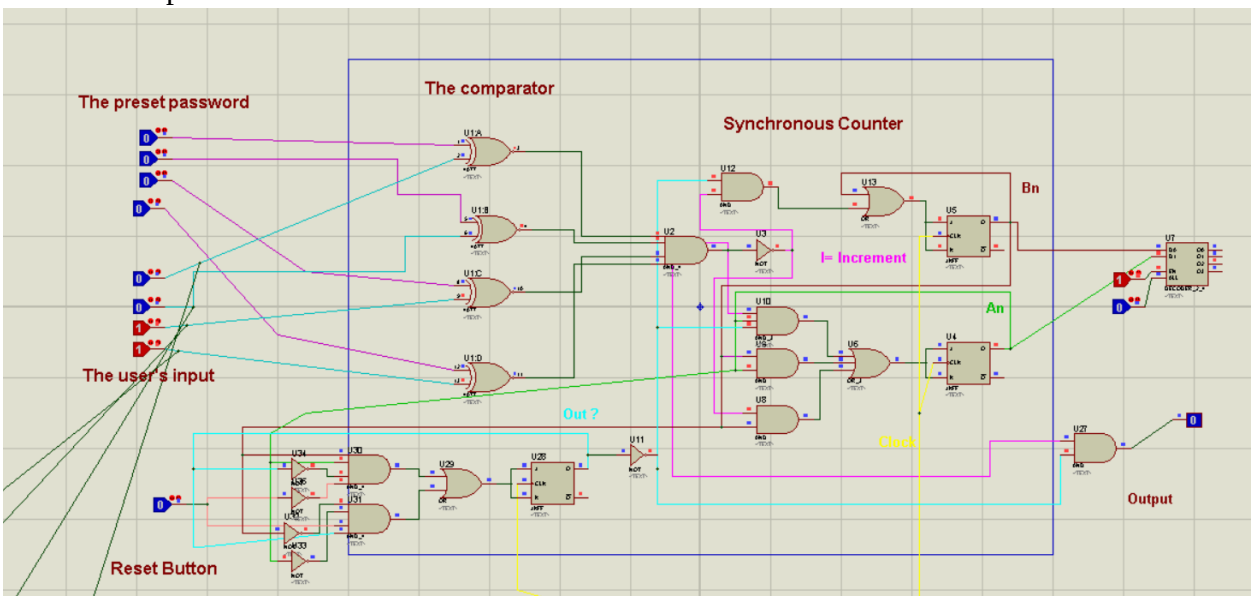
Passwords is 0000

1. User inputs 0111



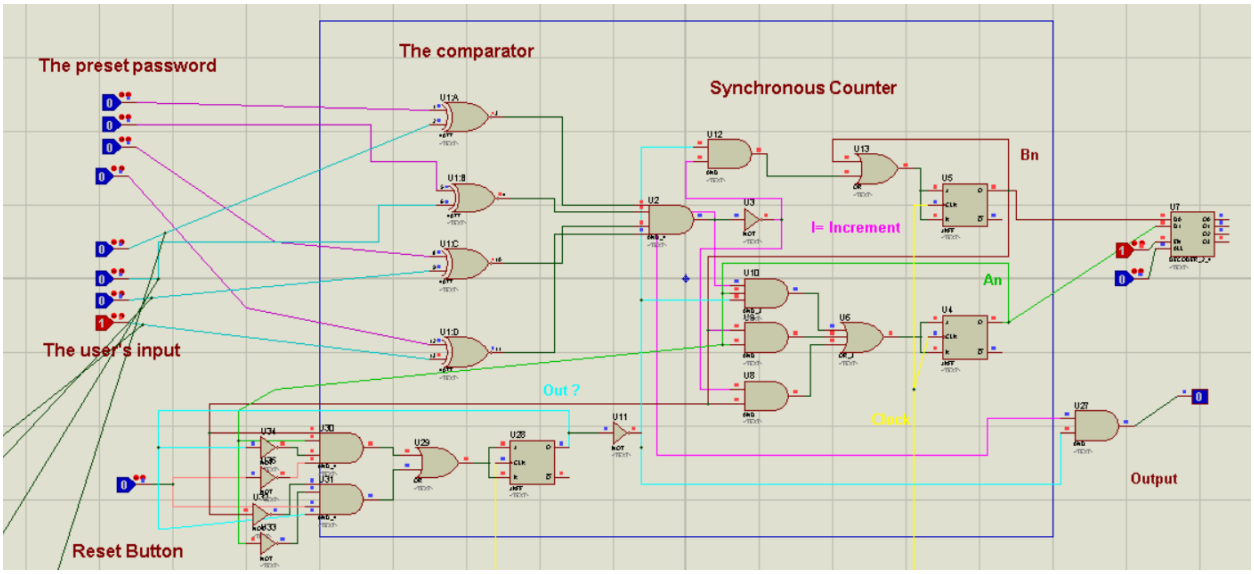
- The counter is 1
- The output is 0

2. User inputs 0011



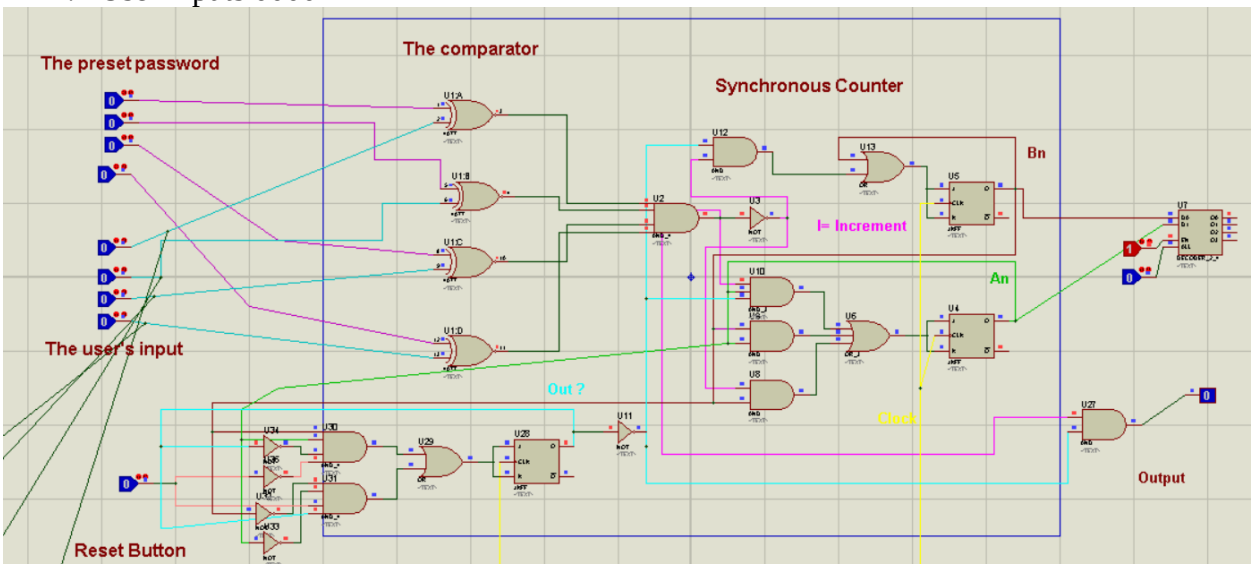
- The counter is 2
- The output is 0

3. User inputs 0001



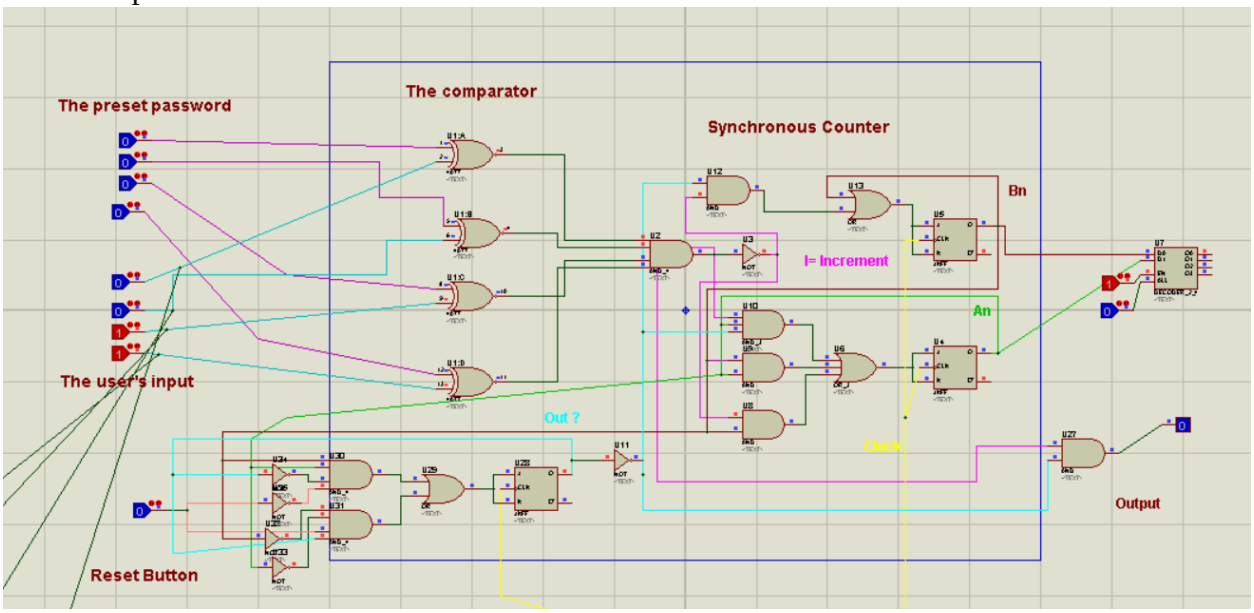
- The counter is 3
- The output is 0

4. User inputs 0000



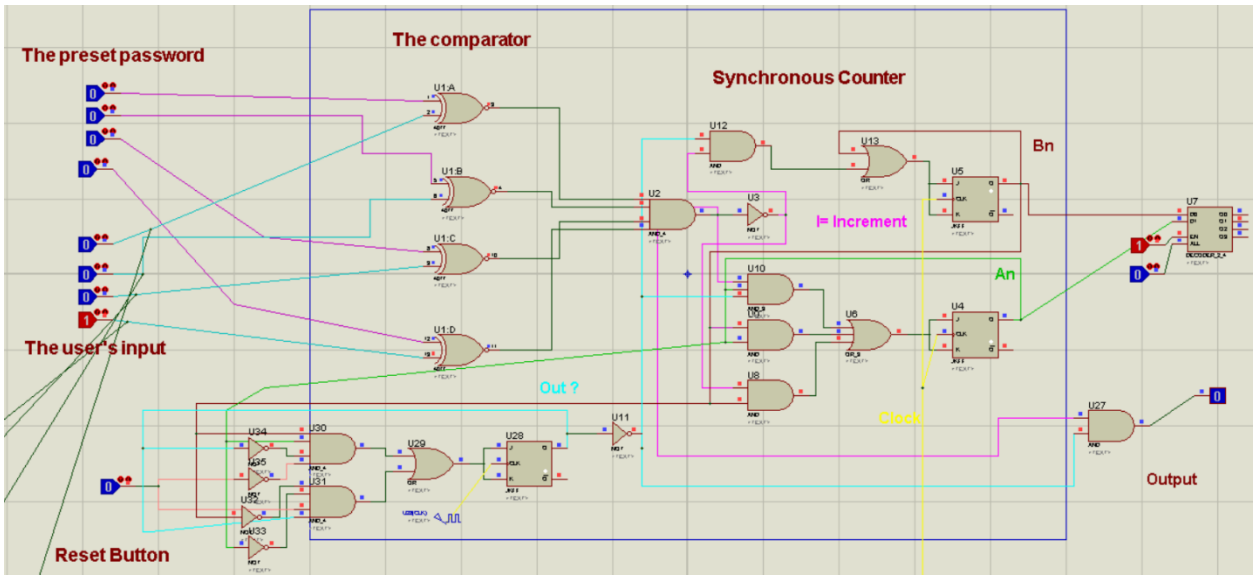
- The counter was reset to 1 and will be frozen like this until reset
- The output is 0, even though the input was correct. But was out of range

5. inputs 0011



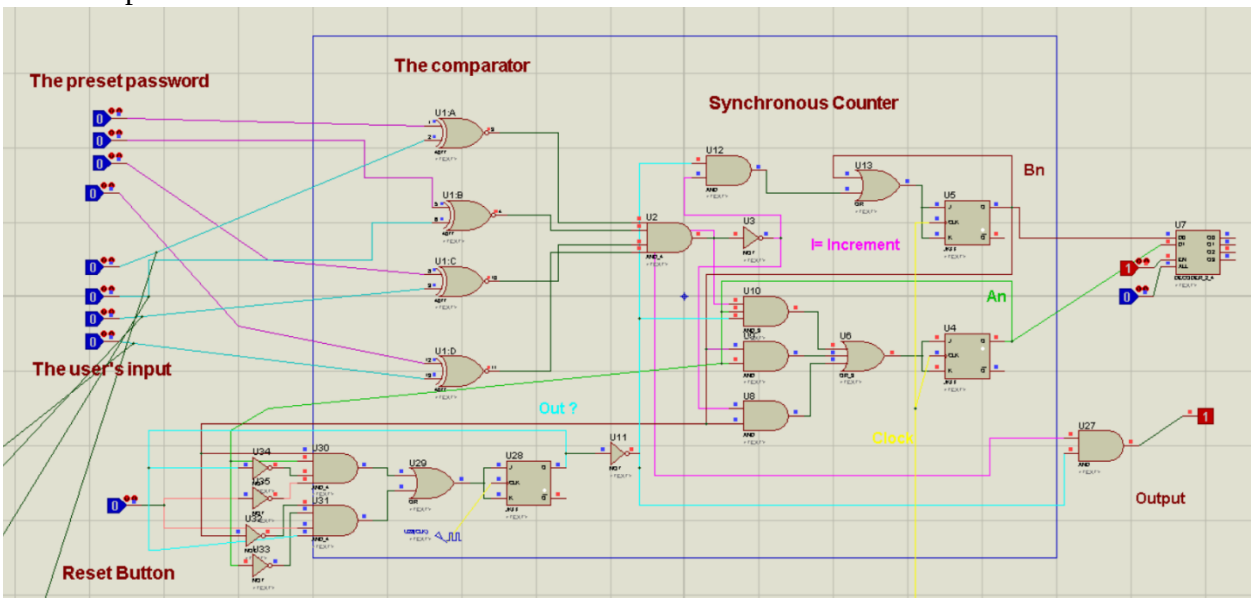
- The counter froze, waiting for the reset button

6. reset, input 0001, close the switch



- Started re-counting
- Still not correct password

7. Input 0000



- finally opened the safe

TinkerCad Simulation:

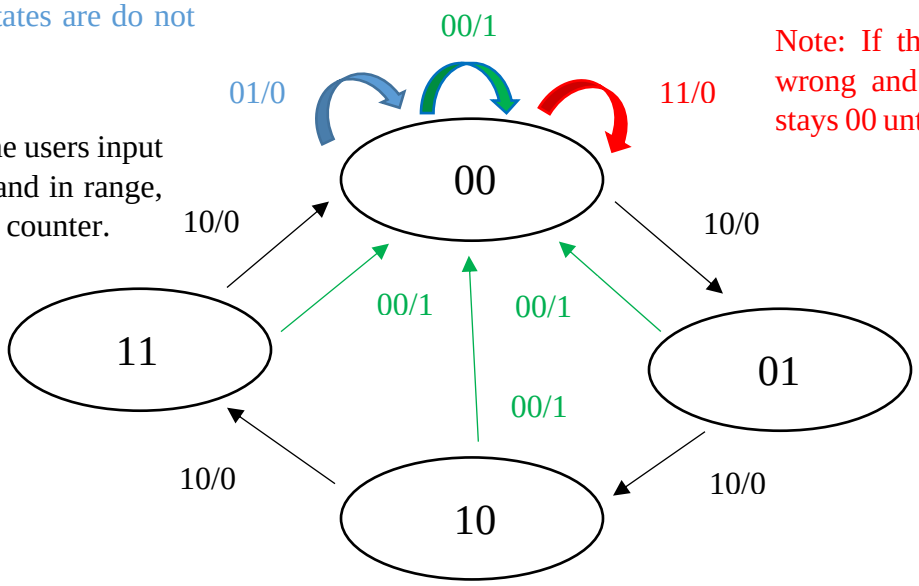
State Diagram:

Arrow labels: Increment, Out/Output

- Increment: 0 if passwords are identical, 1 otherwise.
- Out (out of range?): 1 if counter is more than three, 0 otherwise.
- Output (will the safe open?): 1 if safe successfully opened, 0 otherwise.

Note: If the users input is correct and out of range, state stays 00 and output 0, other states are do not care.

Note: If the users input is wrong and in range, increment counter.



Note: If the users input is wrong and out of range it stays 00 until the user resets.

Note: If the users input is correct and in range, reset and output 1.

Truth Tables, K Maps and Equations:

Truth Table for the entire synchronous counter

- Y is the final Output.
- X is do not care as already stops at zero, PS: count will still occur.

I	An	Bn	Out	A n+1	B n+1	TA	TB	Y
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	1	1
0	0	1	1	X	X	X	X	0
0	1	0	0	0	0	1	0	1
0	1	0	1	X	X	X	X	0
0	1	1	0	0	0	1	1	1
0	1	1	1	X	X	X	X	0
1	0	0	0	0	1	0	1	0
1	0	0	1	0	0	0	0	0
1	0	1	0	1	0	1	1	0
1	0	1	1	X	X	X	X	0
1	1	0	0	1	1	0	1	0
1	1	0	1	X	X	X	X	0
1	1	1	0	0	0	1	1	0
1	1	1	1	X	X	X	X	0

K Map for TA

I, An \ Bn, Out	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Equation for TA

TA= I’ An Out’ + An Bn + I Bn

K Map for TB

I, An \ Bn, Out	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Equation for TB

TB= Bn + I Out’

K Map for Y the final Output

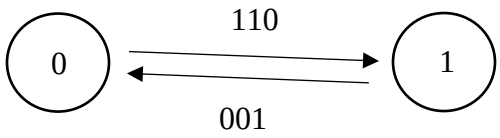
I, An \ Bn, Out	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Equation for Y the final Output

Y= I’ Out’

Truth Table for the Out variable in turquoise

Out is 1 when the counter reaches 3 trials (110), other than that out will stay 0.



An	Bn	Switch	Out	Out n+1	T out
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	1	1	0

K Map for Tout

An, Bn \ S, Out	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Equation for Tout

Tout = An’ Bn’ S Out + An Bn S’ Out’