HOME SECURITY SYSTEM

Samantha Fakhouri

Thomas Filarski

Nathan Kelley

Mathew Plaza

Overview:

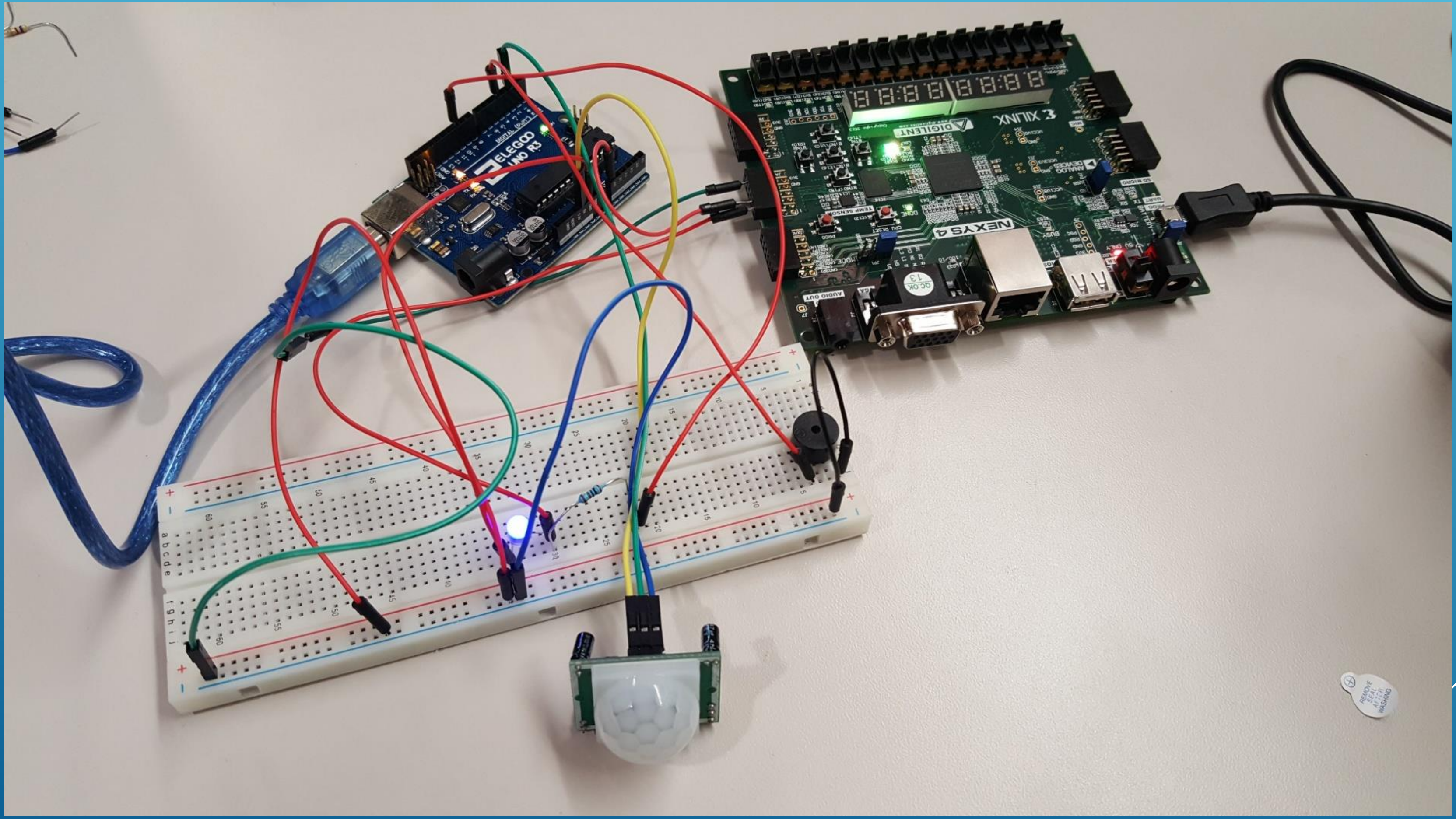
- The purpose of this project is to create a simple and affordable home security system to for use in home or business.

- This project was implemented using Nexys 4 board and an Arduino board to provide a 5V input, a motion sensor to detects the presence of a human, RGB LED to determine the state of the home, a series of switches as a keypad to enter the correct password and a timer, since you have 30 seconds to enter the correct password before the alarm turned on.

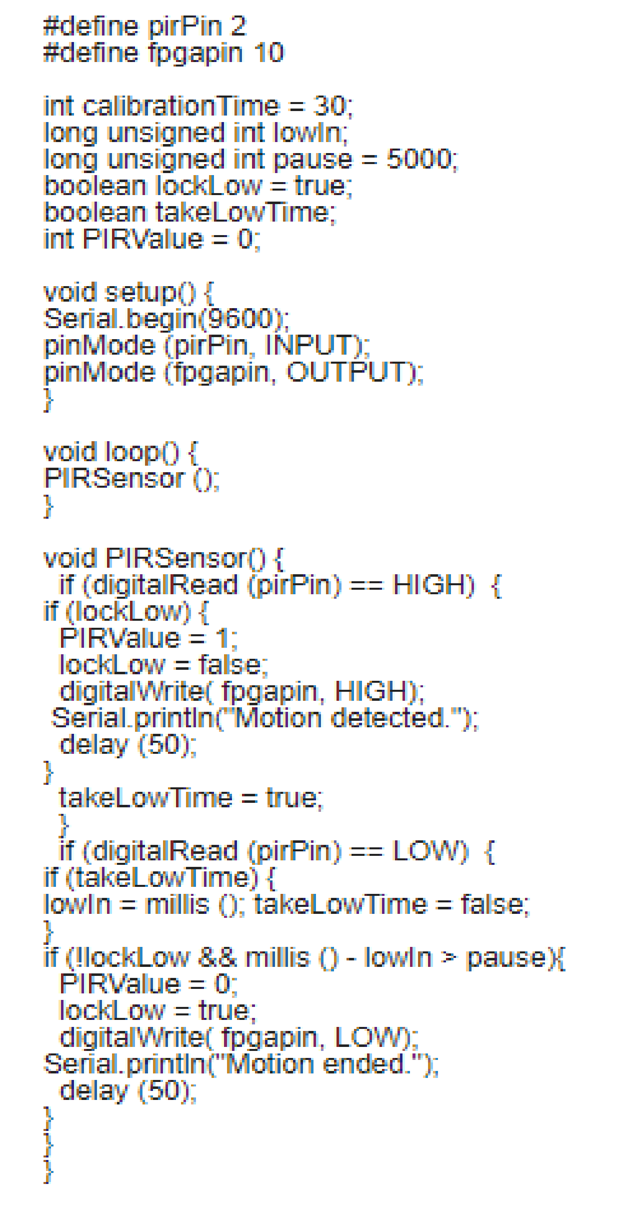
- The code was written using both, VHDL and Arduino C.

Home Security System Circuit

The circuit for this project was built using breadboard, motion sensor, Arduino board, Nexys 4 board, 1k Resistor, Buzzer (so the alarm makes a noise when it rises), LED (it turns on when a motion is detected) and wires.



Some tips about the motion sensor



```
#define pirPin 2
#define fpgapin 10

int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int PIRValue = 0;

void setup() {
Serial.begin(9600);
pinMode (pirPin, INPUT);
pinMode (fpgapin, OUTPUT);
}

void loop() {
PIRSensor ();
}

void PIRSensor() {
  if (digitalRead (pirPin) == HIGH)  {
if (lockLow) {
  PIRValue = 1;
  lockLow = false;
  digitalWrite( fpgapin, HIGH);
 Serial.println("Motion detected.");
  delay (50);
}
  takeLowTime = true;
  }
  if (digitalRead (pirPin) == LOW)  {
if (takeLowTime) {
lowIn = millis (); takeLowTime = false;
}
if (!lockLow && millis () - lowIn > pause){
  PIRValue = 0;
  lockLow = true;
  digitalWrite( fpgapin, LOW);
Serial.println("Motion ended.");
  delay (50);
}
}
}
```

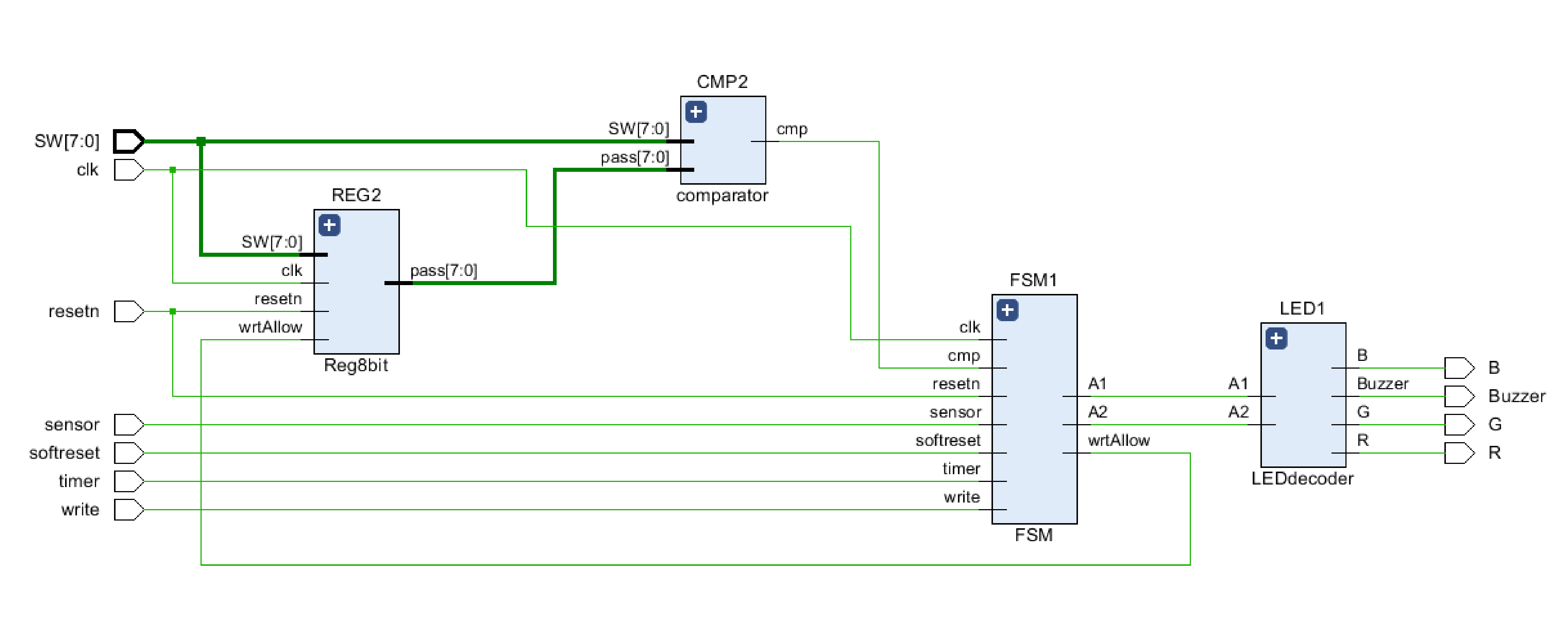
The motion Sensor detects within an angle of 120 degrees and 7 meters.

An LED was connected to the circuit to tell when a motion is detected (It turns on) or when a motion ended (turns off)

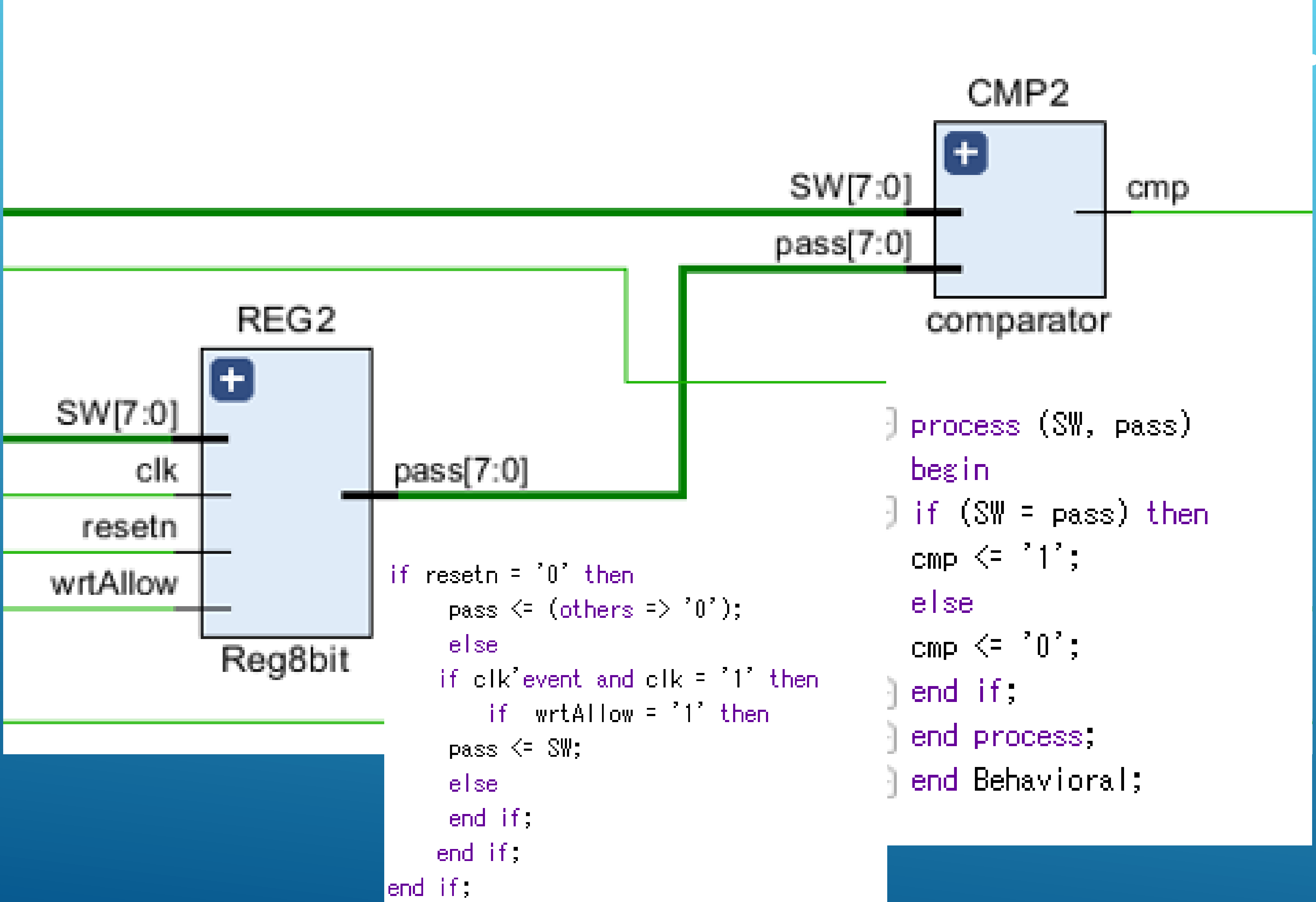
The motion Sensor was connected to an Arduino Board because it needs a 5V as an Input. Therefore, to make it easier, The sensor was codded using Arduino C language.

As noticed from the code, the sensor is an input (5V) and the output is 3.3V (on the Nexys board). So, when a motion is detected or ended, it gives a high output on the Nexys board, and therefore, the RGB LED on the board changes it color.

Block Diagram



Comparator/ Register:



```
if resetn = '0' then
    pass <= (others => '0');
    else
   if clk'event and clk = '1' then
      if  wrtAllow = '1' then
    pass <= SW;
    else
    end if;
   end if;
end if;
```

```
process (SW, pass)
begin
if (SW = pass) then
cmp <= '1';
else
cmp <= '0';
end if;
end process;
end Behavioral;
```

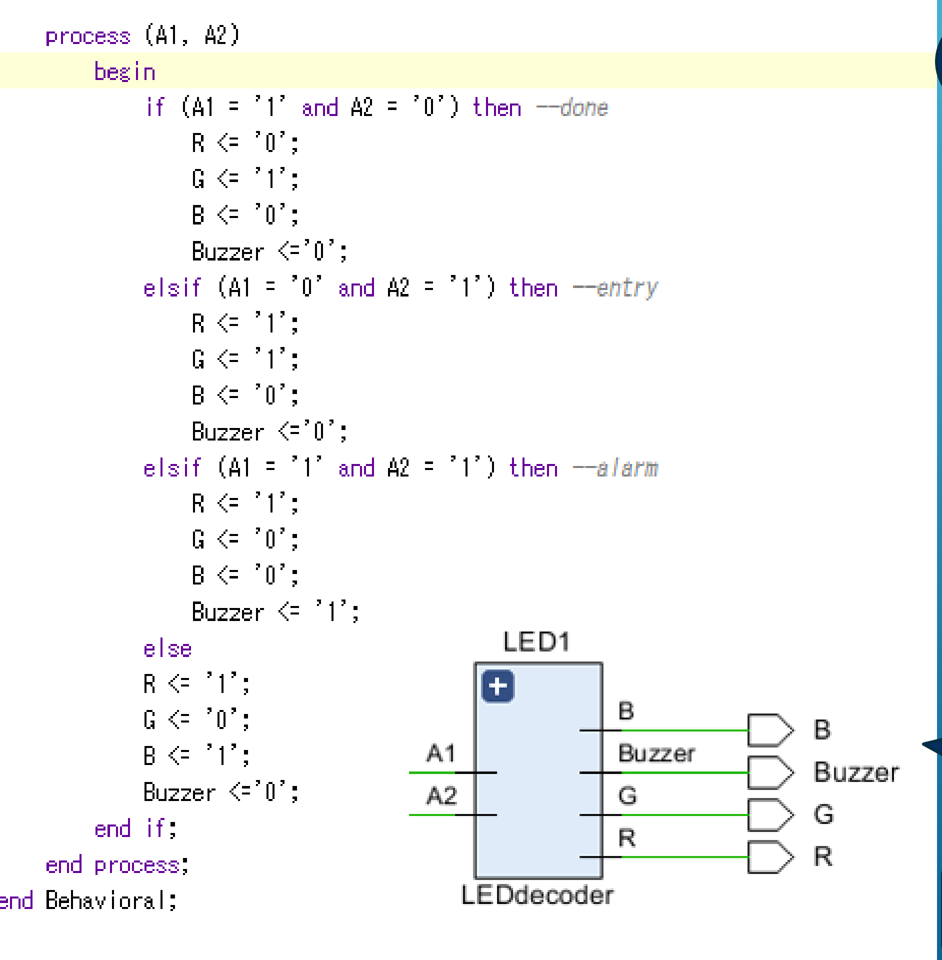
- Register Stores Password.
  - wrtAllow allows user to change password
  - wrtAllow controlled by FSM
  - Password is sent to comparator

Comparator:

- Compares password with current state of switches
- Outputs HIGH if correct passcode is entered

LED Encoder

The code for the LED encoder checks each state of the FSM and activates the correct LED based on each state. This allows the user to know that they have to enter a password, have entered the correct one or to RUN!!!



```
    process (A1, A2)
        begin
            if (A1 = '1' and A2 = '0') then --done
                R <= '0';
                G <= '1';
                B <= '0';
                Buzzer <='0';
            elsif (A1 = '0' and A2 = '1') then --entry
                R <= '1';
                G <= '1';
                B <= '0';
                Buzzer <='0';
            elsif (A1 = '1' and A2 = '1') then --alarm
                R <= '1';
                G <= '0';
                B <= '0';
                Buzzer <= '1';
            else
            R <= '1';
            G <= '0';
            B <= '1';
            Buzzer <='0';
        end if;
    end process;
end Behavioral;
```

CORRECT
10

RGB
010

BUZZER = 0

ENTRY
01

RGB
110

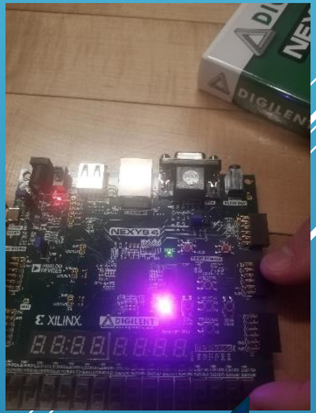
BUZZER = 0

ALARM
11

RGB
100

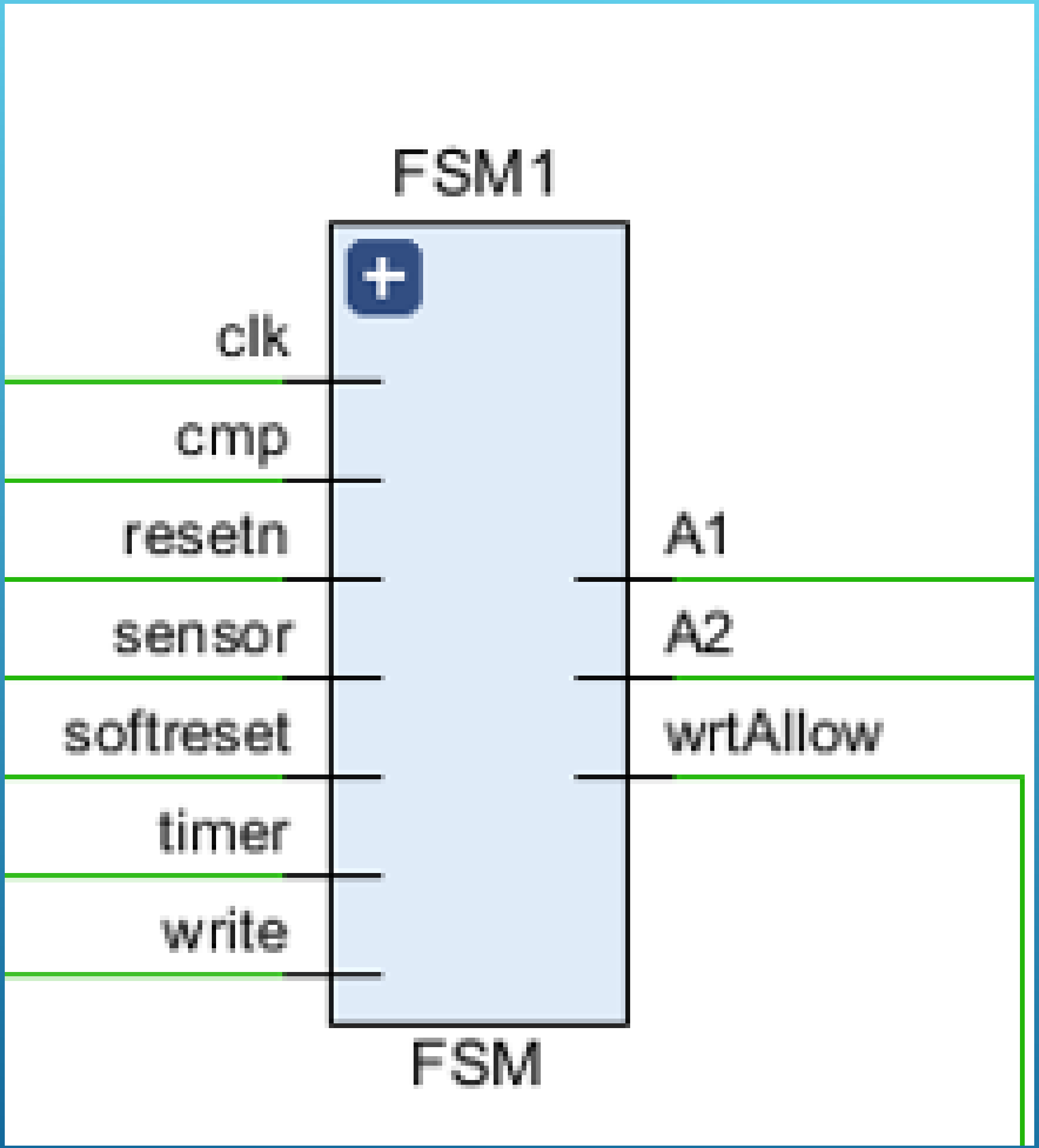
BUZZER = 1

SECURED
00

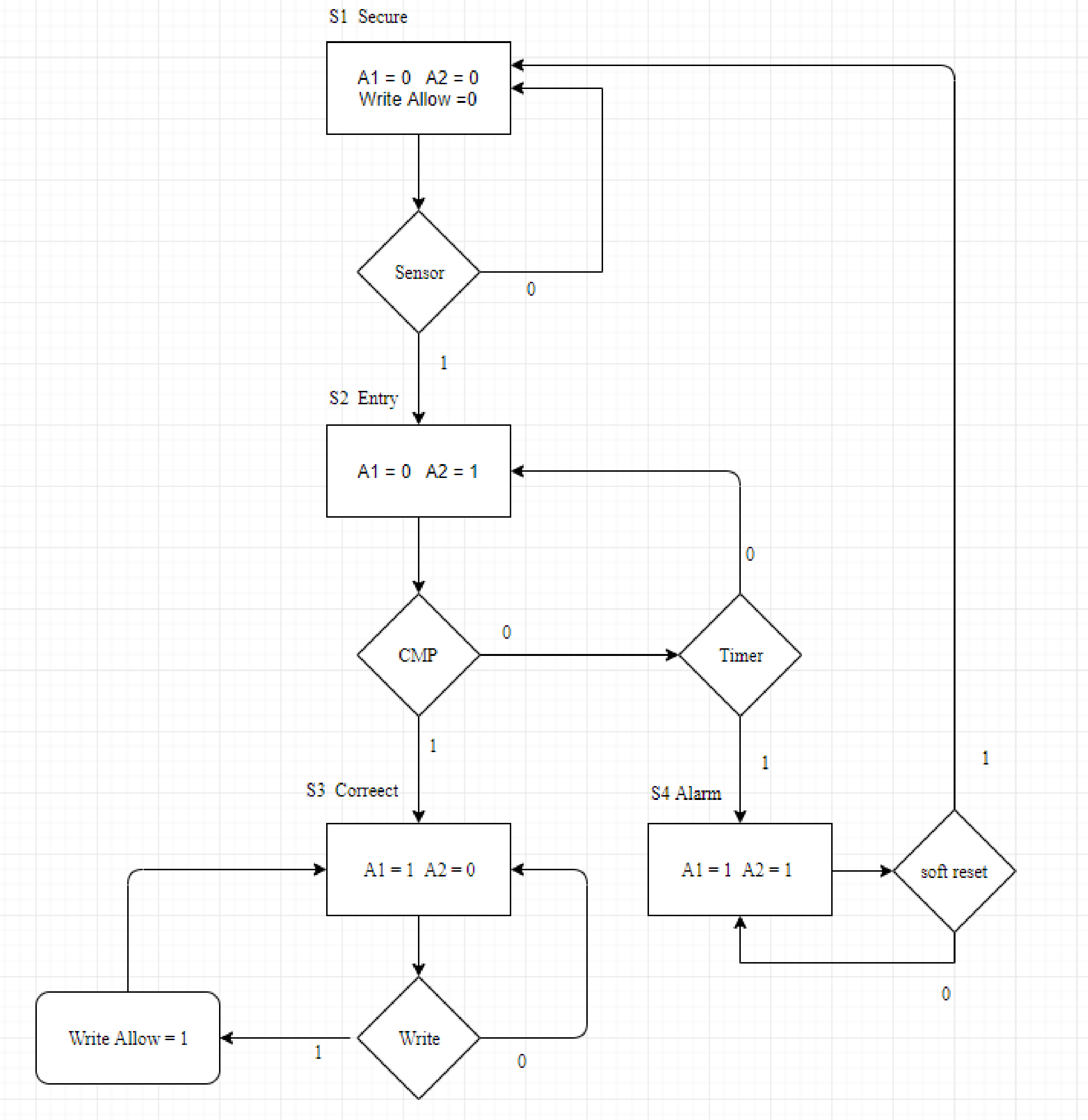


RGB
101

BUZZER = 0


FSM1
clk
cmp
resetn
sensor
softreset
timer
write
A1
A2
wrtAllow
FSM

ASM FSM.


S1 Secure
A1 = 0 A2 = 0
Write Allow =0
Sensor
0
1
S2 Entry
A1 = 0 A2 = 1
0
CMP
0
Timer
1
1
1
S3 Correect
S4 Alarm
A1 = 1 A2 = 0
A1 = 1 A2 = 1
soft reset
0
Write Allow = 1
1
Write
0

Things Planned:

7 Segment Display Output

- Current State of switches
- Eventually, Inputted #s on a keypad
- Current State of FSM

Keypad

- Currently just a single number key switch
- Eventually 4 Digit passcode. Entered Sequentially
- FSM will require all 4 digits to be entered

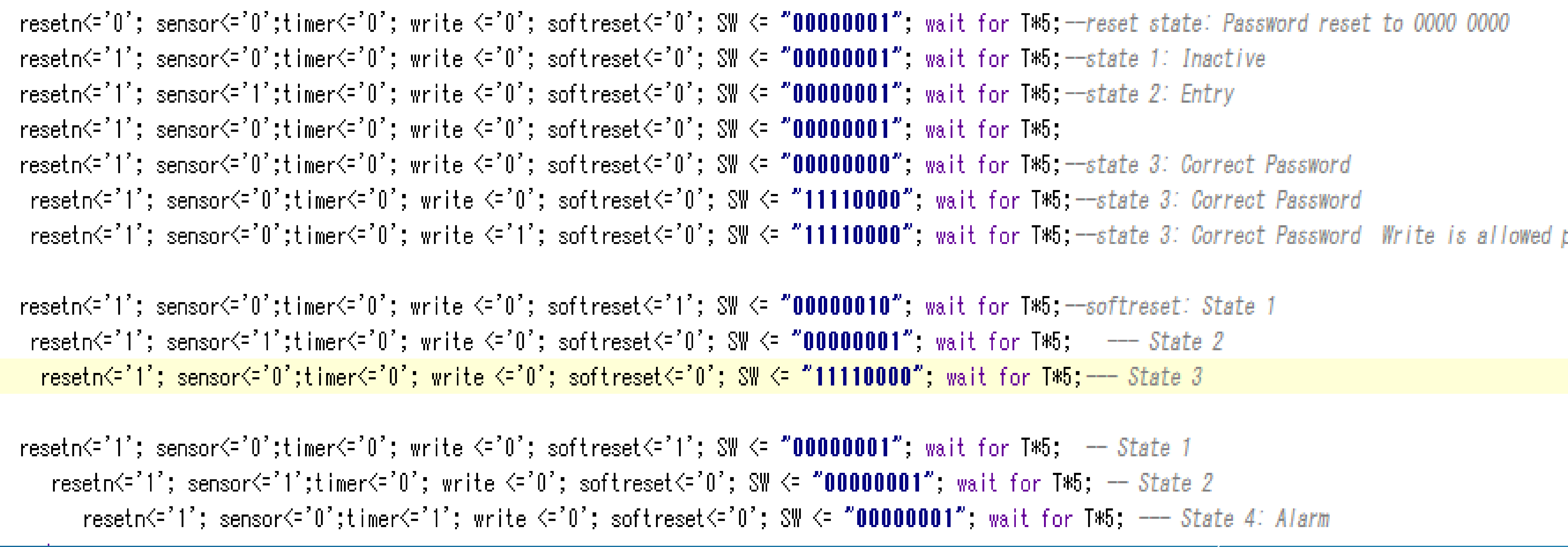
Timer

- Clock Divider
- 1023x1023x10 = 1s
- User will have 30s to enter a proper code sequence

Alarm Buzzer:

Active Piezo Buzzer makes annoying noise when alarm state is reached

Sample Code (From Test Bench)

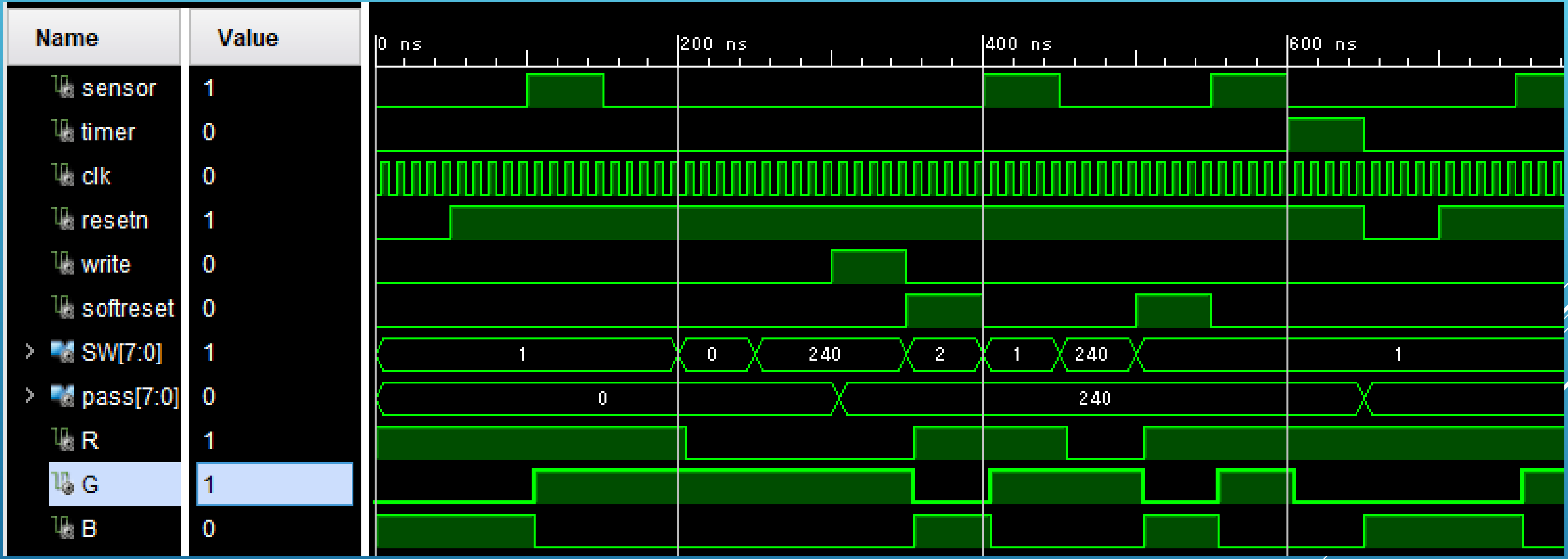


```
resetn<='0'; sensor<='0';timer<='0'; write <='0'; softreset<='0'; SW <= "00000001"; wait for T*5;--reset state: Password reset to 0000 0000
resetn<='1'; sensor<='0';timer<='0'; write <='0'; softreset<='0'; SW <= "00000001"; wait for T*5;--state 1: Inactive
resetn<='1'; sensor<='1';timer<='0'; write <='0'; softreset<='0'; SW <= "00000001"; wait for T*5;--state 2: Entry
resetn<='1'; sensor<='0';timer<='0'; write <='0'; softreset<='0'; SW <= "00000001"; wait for T*5;
resetn<='1'; sensor<='0';timer<='0'; write <='0'; softreset<='0'; SW <= "00000000"; wait for T*5;--state 3: Correct Password
 resetn<='1'; sensor<='0';timer<='0'; write <='0'; softreset<='0'; SW <= "11110000"; wait for T*5;--state 3: Correct Password
 resetn<='1'; sensor<='0';timer<='0'; write <='1'; softreset<='0'; SW <= "11110000"; wait for T*5;--state 3: Correct Password  Write is allowed p

resetn<='1'; sensor<='0';timer<='0'; write <='0'; softreset<='1'; SW <= "00000010"; wait for T*5;--softreset: State 1
 resetn<='1'; sensor<='1';timer<='0'; write <='0'; softreset<='0'; SW <= "00000001"; wait for T*5;    --- State 2
  resetn<='1'; sensor<='0';timer<='0'; write <='0'; softreset<='0'; SW <= "11110000"; wait for T*5;--- State 3

resetn<='1'; sensor<='0';timer<='0'; write <='0'; softreset<='1'; SW <= "00000001"; wait for T*5;   -- State 1
   resetn<='1'; sensor<='1';timer<='0'; write <='0'; softreset<='0'; SW <= "00000001"; wait for T*5; -- State 2
      resetn<='1'; sensor<='0';timer<='1'; write <='0'; softreset<='0'; SW <= "00000001"; wait for T*5; --- State 4: Alarm
```

Timing simulation:



Name	Value
sensor	1
timer	0
clk	0
resetn	1
write	0
softreset	0
SW[7:0]	1
pass[7:0]	0
R	1
G	1
B	0

Time To Test It!!