TEAM 39

DSD Project REPORT

Abdallah Ahmed Hassan **55-2021** T23
Ziad Abdelrahman Ali **55-1371** T20
Mennatullah Essam Omar **55-1132** T10
Sylvia Faris Wardkhan **55-24910** T10
Yousef Ahmed Elbrolosy **55-2615** T10
Nourhan Ibrahim **55-6314** T29

DECEMBER 2023

The aim of this project is to design a security system that employs both sound and motion detection to trigger an alarm and display a warning message on a 7-segment display. The system is activated by a user through a switch.

Inputs:

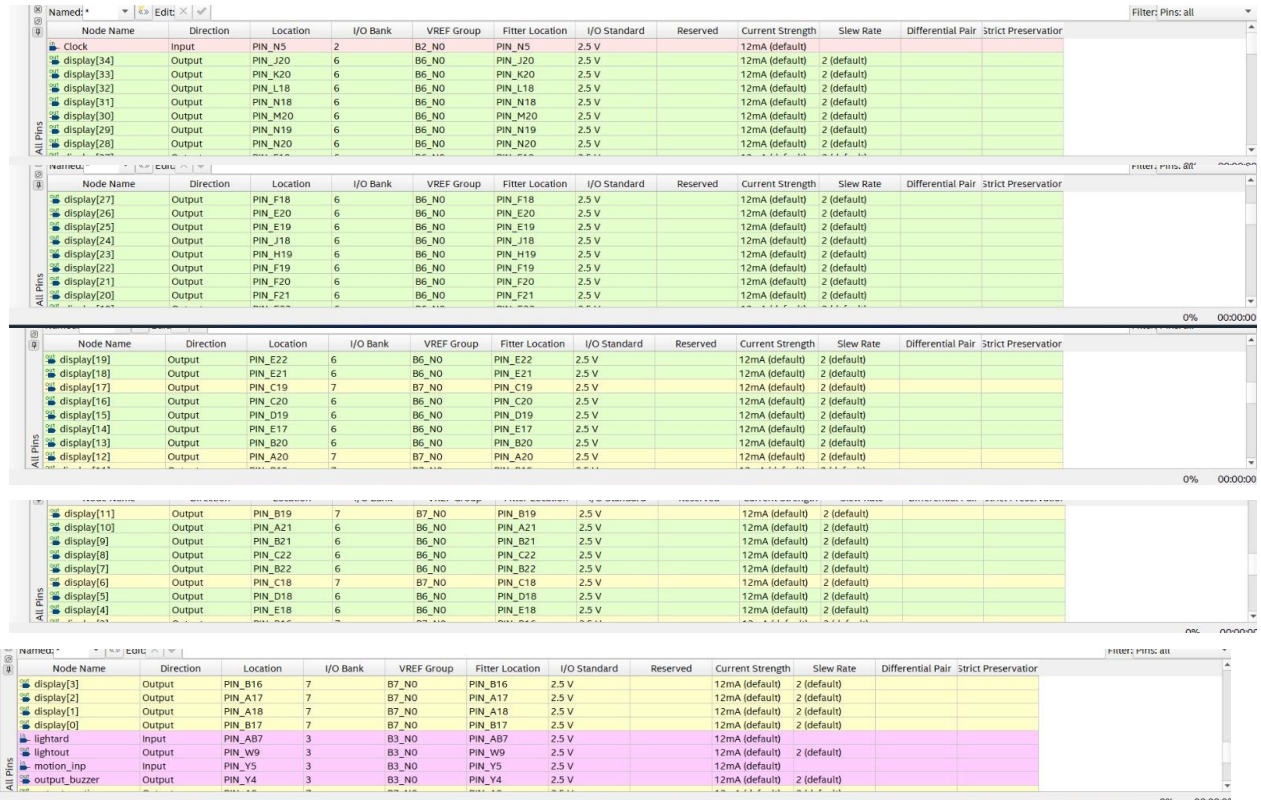
- `Clock`: System clock signal.
- `sound_inp`: Input signal from the sound sensor.
- `motion_inp`: Input signal from the motion sensor.
- `start`: Input signal to start the security system.

Outputs:

- `output_sound`: Output signal indicating sound detection.
- `output_motion`: Output signal indicating motion detection.
- `servo_move`: Output signal to activate the servo for door locking.
- `servo_default`: Output signal for resetting the servo.
- `output_buzzer`: Output signal for activating an alarm buzzer.
- `display`: Output signal for the 7-segment display.

Pin Assignments:

Note: Lightard and lightout input/output are for the bonus part



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
Clock	Input	PIN_N5	2	B2_N0	PIN_N5	2.5 V		12mA (default)			
display[34]	Output	PIN_J20	6	B6_N0	PIN_J20	2.5 V		12mA (default)	2 (default)		
display[33]	Output	PIN_K20	6	B6_N0	PIN_K20	2.5 V		12mA (default)	2 (default)		
display[32]	Output	PIN_L18	6	B6_N0	PIN_L18	2.5 V		12mA (default)	2 (default)		
display[31]	Output	PIN_N18	6	B6_N0	PIN_N18	2.5 V		12mA (default)	2 (default)		
display[30]	Output	PIN_M20	6	B6_N0	PIN_M20	2.5 V		12mA (default)	2 (default)		
display[29]	Output	PIN_N19	6	B6_N0	PIN_N19	2.5 V		12mA (default)	2 (default)		
display[28]	Output	PIN_N20	6	B6_N0	PIN_N20	2.5 V		12mA (default)	2 (default)		
display[27]	Output	PIN_F18	6	B6_N0	PIN_F18	2.5 V		12mA (default)	2 (default)		
display[26]	Output	PIN_E20	6	B6_N0	PIN_E20	2.5 V		12mA (default)	2 (default)		
display[25]	Output	PIN_E19	6	B6_N0	PIN_E19	2.5 V		12mA (default)	2 (default)		
display[24]	Output	PIN_J18	6	B6_N0	PIN_J18	2.5 V		12mA (default)	2 (default)		
display[23]	Output	PIN_H19	6	B6_N0	PIN_H19	2.5 V		12mA (default)	2 (default)		
display[22]	Output	PIN_F19	6	B6_N0	PIN_F19	2.5 V		12mA (default)	2 (default)		
display[21]	Output	PIN_F20	6	B6_N0	PIN_F20	2.5 V		12mA (default)	2 (default)		
display[20]	Output	PIN_F21	6	B6_N0	PIN_F21	2.5 V		12mA (default)	2 (default)		
display[19]	Output	PIN_E22	6	B6_N0	PIN_E22	2.5 V		12mA (default)	2 (default)		
display[18]	Output	PIN_E21	6	B6_N0	PIN_E21	2.5 V		12mA (default)	2 (default)		
display[17]	Output	PIN_C19	7	B7_N0	PIN_C19	2.5 V		12mA (default)	2 (default)		
display[16]	Output	PIN_C20	6	B6_N0	PIN_C20	2.5 V		12mA (default)	2 (default)		
display[15]	Output	PIN_D19	6	B6_N0	PIN_D19	2.5 V		12mA (default)	2 (default)		
display[14]	Output	PIN_E17	6	B6_N0	PIN_E17	2.5 V		12mA (default)	2 (default)		
display[13]	Output	PIN_B20	6	B6_N0	PIN_B20	2.5 V		12mA (default)	2 (default)		
display[12]	Output	PIN_A20	7	B7_N0	PIN_A20	2.5 V		12mA (default)	2 (default)		
display[11]	Output	PIN_B19	7	B7_N0	PIN_B19	2.5 V		12mA (default)	2 (default)		
display[10]	Output	PIN_A21	6	B6_N0	PIN_A21	2.5 V		12mA (default)	2 (default)		
display[9]	Output	PIN_B21	6	B6_N0	PIN_B21	2.5 V		12mA (default)	2 (default)		
display[8]	Output	PIN_C22	6	B6_N0	PIN_C22	2.5 V		12mA (default)	2 (default)		
display[7]	Output	PIN_B22	6	B6_N0	PIN_B22	2.5 V		12mA (default)	2 (default)		
display[6]	Output	PIN_C18	7	B7_N0	PIN_C18	2.5 V		12mA (default)	2 (default)		
display[5]	Output	PIN_D18	6	B6_N0	PIN_D18	2.5 V		12mA (default)	2 (default)		
display[4]	Output	PIN_E18	6	B6_N0	PIN_E18	2.5 V		12mA (default)	2 (default)		
display[3]	Output	PIN_B16	7	B7_N0	PIN_B16	2.5 V		12mA (default)	2 (default)		
display[2]	Output	PIN_A17	7	B7_N0	PIN_A17	2.5 V		12mA (default)	2 (default)		
display[1]	Output	PIN_A18	7	B7_N0	PIN_A18	2.5 V		12mA (default)	2 (default)		
display[0]	Output	PIN_B17	7	B7_N0	PIN_B17	2.5 V		12mA (default)	2 (default)		
lightard	Input	PIN_AB7	3	B3_N0	PIN_AB7	2.5 V		12mA (default)			
lightout	Output	PIN_W9	3	B3_N0	PIN_W9	2.5 V		12mA (default)	2 (default)		
motion_inp	Input	PIN_Y5	3	B3_N0	PIN_Y5	2.5 V		12mA (default)			
output_buzzer	Output	PIN_Y4	3	B3_N0	PIN_Y4	2.5 V		12mA (default)	2 (default)		

output_motion	Output	PIN_A2	8	B8_N0	PIN_A2	2.5 V		12mA (default)	2 (default)		
output_sound	Output	PIN_A8	7	B7_N0	PIN_A8	2.5 V		12mA (default)	2 (default)		
servo_default	Output	PIN_AA2	3	B3_N0	PIN_AA2	2.5 V		12mA (default)	2 (default)		
servo_move	Output	PIN_AB5	3	B3_N0	PIN_AB5	2.5 V		12mA (default)	2 (default)		
sound_inp	Input	PIN_W10	3	B3_N0	PIN_W10	2.5 V		12mA (default)			
start	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V		12mA (default)			
testout	Output	PIN_A10	7	B7_N0	PIN_A10	2.5 V		12mA (default)	2 (default)		

FPGA Code Explanation:

signal Counter : INTEGER := 0;
signal FLAG : STD_LOGIC := '0';

Counter: This signal is an integer used for counting the clock cycles. It is initially set to 0 and is incremented in the process

FLAG: This signal is a single-bit logic signal. It is used as a flag to prevent repetitive actions. When certain conditions are met in the process, FLAG is set to '1', and actions will not be repeated until FLAG is reset.

```
Door_Control: PROCESS(sound_inp, motion_inp)
  BEGIN
    IF rising_edge(Clock) THEN
      IF start = '1' THEN
        Counter <= Counter + 1;
        IF sound_inp = '1' AND FLAG = '0' THEN
          output_sound <= '1';
          output_buzzer <= '1';
          FLAG <= '1';  -- Set flag to prevent repetitive actions
          servo_move<='1';
          servo_default<='0';
          display <= "00100100010010011100001001001001000";
         lightout<= '1';
        END IF;
        IF motion_inp = '1' AND FLAG = '0' THEN
          output_motion <= '1';
          output_buzzer <= '1';
          FLAG <= '1';  -- Set flag to prevent repetitive actions
          servo_move<='1';
           servo_default<='0';
           display <= "00100100010010011100001001001001000";
           lightout<= '1';
        END IF;
         IF Counter = 80000000 THEN  -- Assuming a clock frequency of 10 MHz for 8 seconds delay
          output_sound <= '0';
          output_motion <= '0';
          FLAG <= '1';  -- Set flag to prevent repetitive actions
          output_buzzer <= '0';
```

END IF;

Door_Control: This is the name of the process and is triggered when there is a change in sound_inp and motion_inp signal and then it checks for a rising edge of the clock signal and then checks whether the start signal or motion signal are activated or not: If the start signal is 1, then counter is incremented and then checks if sound_inp or motion_inp are detected:

If the sound is activated and flag is 0 then the `output_buzzer` signal is set to '1', indicating that the buzzer should be activated. Also, The FLAG is set to '1' to prevent repetitive actions until it is reset. This sets the `servo_move` signal to '1', activating the servo for door locking. Moreover, the `servo_default` signal is set to '0', potentially indicating a state change in the servo AND the 7-segment display will display "22FSH". Also, lightout signal is set to 1.

If the motion signal is detected and flag =0 then, the `output_motion` signal is set to '1', indicating that motion has been detected.Also, `output_buzzer` signal is set to '1', indicating that the buzzer should be activated. The FLAG is also set to '1' to prevent repetitive actions until it is reset. In addition, This sets the `servo_move` signal to '1', activating the servo for door locking.and setting the `servo_default` signal to '0', potentially indicating a state change in the servo and the 7-segment display will display "22FSH". Also, lightout signal is set to 1.

If the `Counter = 80000000`, which assumes a clock frequency of 10 MHz for an 8-second delay. If this condition is true, the following actions are taken: `output_sound` signal is set to '0', indicating that the sound detection output is turned off. Also, `output_motion` signal is set to '0', indicating that the motion detection output is turned off**.**In addition, ,The FLAG is set to '1' to prevent repetitive actions until it is reset.And the `output_buzzer` signal is set to '0', turning off the buzzer.

ELSE

Counter <= 0; -- Reset counter if start is not '1'

FLAG <= '0'; -- Reset flag if start is not '1'

output_sound <= '0';

output_motion <= '0';

output_buzzer <= '0';

servo_move<='0';

servo_default<='1';

display <= "1111111111111111111111111111111111";

ightout<= '0'

If the start signal is not 1, `Counter` signal is reset to 0, ensuring that the delay counter starts from the beginning if the system is not in the "start" state. The `FLAG` signal is reset to '0', allowing the system to perform actions when sound or motion is detected. In addition, The `output_sound` signal is set to '0', indicating that sound detection output is turned off. The `output_motion` signal is set to '0', indicating that motion detection output is turned off. Moreover, The `output_buzzer` signal is set to '0', turning off the buzzer. `Setting` The `servo_move` signal to '0', deactivating the servo for door locking. And finally, The `servo_default` signal is set to '1', potentially indicating a default state for the servo and the 7-segment will display nothing. Also, lightout is set to 0.

Arduino Code Explanation:

```
#include <Servo.h>
Servo myservo;
```

Servo myservo: creates an instance of the Servo class called 'myservo'. This object will be used to control the servo motor connected to the Arduino.

```
void setup() {
  myservo.attach(9);

}
```

`setup()`: A standard Arduino function that runs once when the board is powered up or reset. Inside this function:

- `myservo.attach(9);`: This line attaches the servo motor to pin 9 on the Arduino board. It means that the signal wire of the servo is connected to pin 9, allowing the Arduino to control the servo.

```
void loop() {
  int out;
  int out2;
  out = digitalRead(8);
  out2 = digitalRead(10);
```

The `loop()` function is another standard Arduino function that runs repeatedly after the `setup()` function.

*`Inside this function we declared` **`int out;`** **and** **`int out2;`**: to store the state of digital inputs connected to pins 8 and 10, respectively.*

- **`out = digitalRead(8);`** **and** **`out2 = digitalRead(10);`**: *These lines read the digital state (HIGH or LOW) of pins 8 and 10, respectively, and store the values in the variables* **`out`** **and** **`out2`**

```
if (out == HIGH){
   myservo.write(0);
   }
```
This conditional statement checks if the input connected to pin 8 is in a HIGH state If true: the servo motor is instructed to move to a position corresponding to 0 degrees.

```
  if (out2 == HIGH){
   myservo.write(90);
   }
}
```
This conditional statement checks if the input connected to pin 10 is in a HIGH state. If true:the servo motor is instructed to move to a position corresponding to 90 degrees.