Course Project Report

On

DIGITAL LOCK SYSTEM USING VERILOG

Submitted by

AHIRAJ K
AGNET MARIA SOY
ADWAITH RAVEENDRAN

In partial fulfilment for the award of Degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

For the course

ECT203 LOGIC CIRCUIT DESIGN

Course Coordinator

Prof. Annie George

Department of Electronics and Communication Engineering
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
KOTTAYAM - 686 501

November 2024

ABSTRACT

This project presents the design and simulation of a simple digital lock system using Verilog, developed and tested in Vivado. The digital lock system is designed to validate a 4-bit user-input code against a preset 4-bit password. It includes two primary outputs: an *unlock* signal, which activates when the correct code is entered, and an *alarm* signal, which triggers upon detection of an incorrect code entry, enhancing the security of the system by alerting to unauthorized attempts.

Additionally, a reset feature is incorporated, allowing the system to revert to a locked state and deactivate the alarm as necessary. The digital lock's functionality was rigorously tested using a Verilog testbench, covering various scenarios with both valid and invalid inputs to confirm correct operation. This simulation-based project provides a foundational approach to implementing access control systems in digital hardware design.

Contents

List of Figures

1.INTRODUCTION

In today's digital world, secure access control is a crucial aspect of protecting sensitive information and controlling entry to restricted areas. This project presents the design and implementation of a simple digital lock system using Verilog. The digital lock system aims to provide a secure mechanism by comparing a 4-bit user-input code with a preset 4-bit password, allowing access only upon successful validation of the code.

The system includes two primary output signals: an unlock signal that activates when the correct code is entered, and an *alarm* signal that triggers in response to an incorrect code. Additionally, a reset function is implemented to allow the lock to return to its initial locked state, disabling the alarm as necessary. This digital lock was developed and simulated in Vivado, and its functionality was verified through comprehensive testing using a Verilog testbench, covering both valid and invalid code entries.

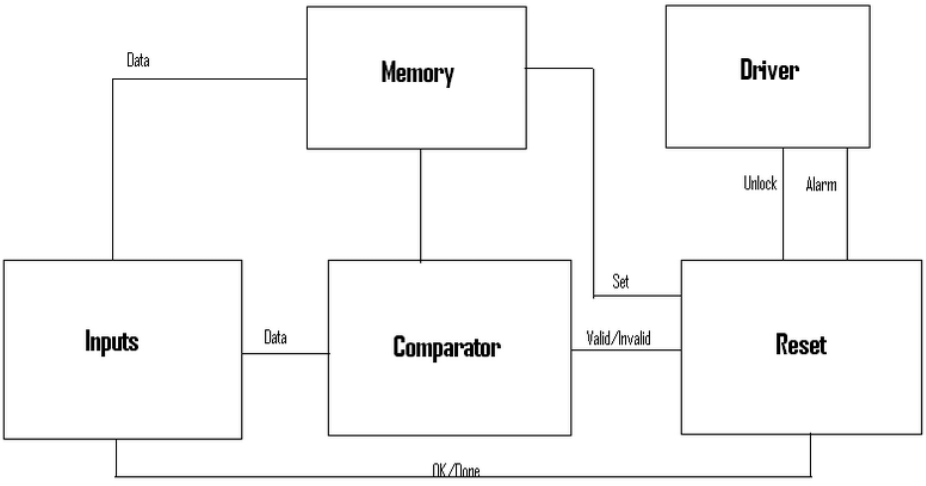


Figure 1.1 : Block diagram

2.SYSTEM DESCRIPTION

2.1 WORKING

The digital lock system operates by comparing a 4-bit input code with a predefined 4-bit password. When the system powers on, it remains in a locked state, awaiting user input. Each time a user enters a code, the system evaluates it against the preset password:

1. **Correct Code Entry**: When the user-input code matches the preset password, the *unlock* signal is activated, indicating that access is granted.
2. **Incorrect Code Entry**: If the entered code does not match the preset password, the *alarm* signal is triggered, signaling an unauthorized attempt.
3. **ALARM SYSTEM**: Alarm system in this digital lock activates on incorrect password entries, signaling potential unauthorized access by setting alarm = 1. It remains off with correct entries (alarm = 0) and resets to a safe state when reset is triggered. This provides feedback on failed attempts, enhancing security by alerting users to access issues.
4. **Reset Function**: A reset feature allows the system to be reset to its initial locked state, which also deactivates the alarm signal. This feature provides users with a convenient way to retry code entry after an incorrect attempt.

The system was tested through a Verilog testbench that verified its response to both correct and incorrect inputs, confirming that it transitions smoothly between locked and unlocked states under different scenarios. This testing confirmed the system’s functionality and robustness.

2.2 CODE

```
module Digital_Lock (
    input [3:0] password,          // 4-bit stored password
    input [3:0] input_code,        // 4-bit entered code
    input reset,                   // Reset signal
    output reg unlock,             // Unlock signal
    output reg alarm               // Alarm signal
);

    // Parameterized password (let's say the password is '1010')

    parameter [3:0] stored_password = 4'b1010;

    always @(*) begin
        if (reset) begin
            unlock = 0;
            alarm = 0;
        end else if (input_code == stored_password) begin
            unlock = 1;
            alarm = 0;  // No alarm if the code is correct
        end else begin
            unlock = 0;
            alarm = 1;  // Raise alarm if the code is incorrect
        end
    end
endmodule
```

2.3 TESTBENCH

```
`timescale 1ns / 1ps

module Digital_Lock_tb;

    // Inputs
    reg [3:0] input_code;
    reg reset;

    // Outputs
    wire unlock;
    wire alarm;

    // Instantiate the Digital_Lock module
    Digital_Lock uut (
        .password(4'b1010),   // Using the stored password '1010'
        .input_code(input_code),
        .reset(reset),
        .unlock(unlock),
        .alarm(alarm)
    );

    // Test procedure
    initial begin
        // Initialize inputs
        reset = 1;
        input_code = 4'b0000;

        // Wait for reset to take effect
        #10;
        reset = 0;

        // Test Case 1: Enter correct password
        input_code = 4'b1010;   // Correct password
        #10;
        $display("Test Case 1: Correct password");
        $display("Unlock: %b, Alarm: %b", unlock, alarm);
```

```
        // Test Case 2: Enter incorrect password
        input_code = 4'b1100;   // Incorrect password
        #10;
        $display("Test Case 2: Incorrect password");
        $display("Unlock: %b, Alarm: %b", unlock, alarm);

        // Test Case 3: Another incorrect password
        input_code = 4'b0110;   // Incorrect password
        #10;
        $display("Test Case 3: Incorrect password");
        $display("Unlock: %b, Alarm: %b", unlock, alarm);

        // Test Case 4: Reset the system
        reset = 1;
        #10;
        $display("Test Case 4: System Reset");
        $display("Unlock: %b, Alarm: %b", unlock, alarm);

        // End simulation
        $stop;
    end
endmodule
```

2.4 RTL SCHEMATIC DIAGRAM

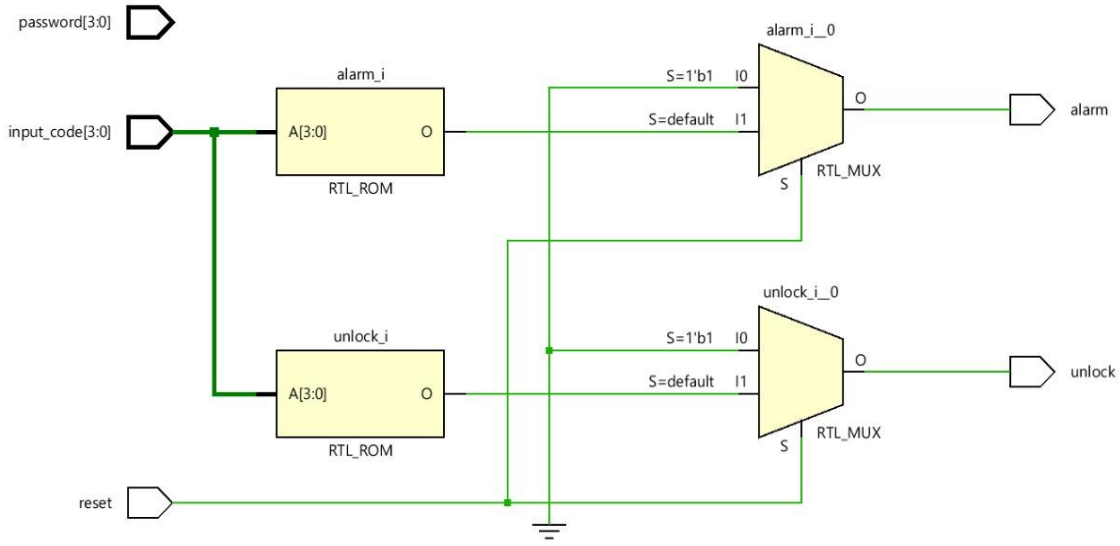


Figure 2.1 : RTL schematic diagram

2.5 TRUTH TABLE

Test case	Reset	Input_code	Unlock	Alarm	Description
1	0	1010	1	0	Correct password
2	0	1100	0	1	Incorrect password
3	0	0110	0	1	Incorrect password
4	1	XXXX	0	0	System reset

Figure 2.2 : Truth table

3.SOFTWARE USED

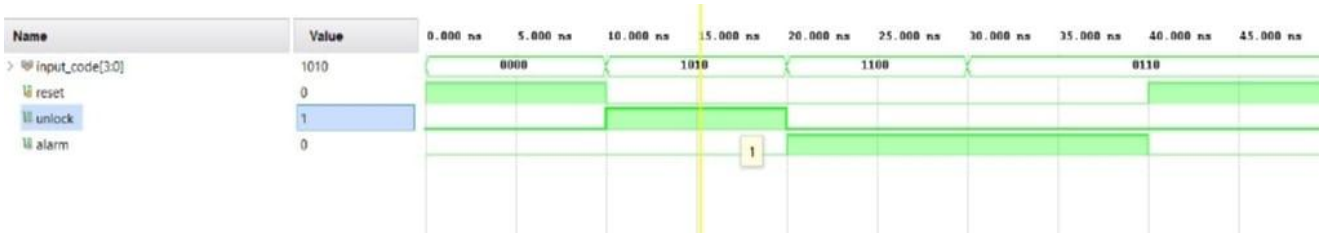
For this project, the **Xilinx Vivado Design Suite** was utilized as the primary environment for code development and simulation. Vivado is a robust tool from Xilinx tailored for hardware description languages like Verilog, providing a comprehensive platform for digital system design, simulation, and verification. Although this project focused solely on software simulation without hardware implementation, Vivado offered an efficient environment to fully test the digital lock system.

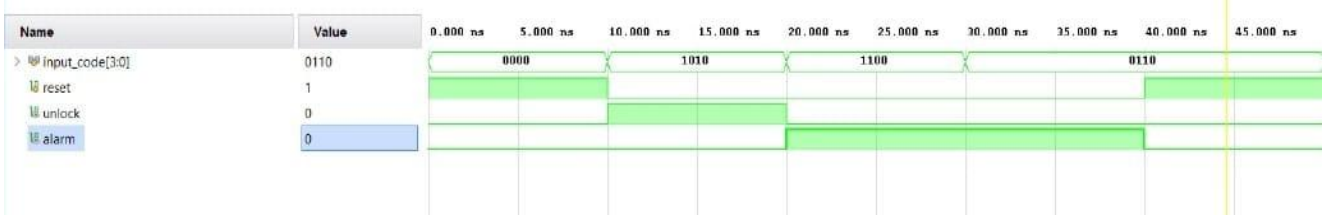
Key features of Vivado that supported the development of this project include:

1. **Code Development**: Vivado provides a powerful editor for Verilog, featuring syntax highlighting, error-checking, and code navigation. These capabilities streamlined the creation of the digital lock system's Verilog modules.
2. **Simulation and Debugging**: The Vivado simulator enabled thorough testing of the lock system by allowing the creation of a testbench that emulates user interactions. Through simulation, the system's response to both correct and incorrect codes was verified, confirming the proper behavior of the *unlock* and *alarm* signals.
3. **Waveform Viewer**: Vivado's waveform viewer provided a graphical representation of signal transitions over time, which was crucial for observing the state of output signals during testing. The waveform display helped confirm that the *unlock*, *alarm*, and *reset* signals responded as expected under different test scenarios.

By leveraging Vivado's simulation tools, the digital lock system was thoroughly verified for functionality and accuracy, ensuring reliable operation without the need for physical hardware. This simulation-based approach highlighted the strength of Vivado in digital system design and testing.

4.RESULS AND CONCLUTIONS





This project successfully demonstrates a basic lock mechanism using Verilog. By verifying an entered code against a stored password, the system can control access effectively. The design ensures that when a correct code is entered, the lock opens, indicated by the unlock signal. When an incorrect code is entered, an alarm is triggered, maintaining security.

The system’s straightforward architecture offers reliability and can serve as a foundation for more advanced digital security applications. Possible enhancements, such as adding a dynamic password update feature, attempt limitation, or a timer for automatic alarm reset, could increase its robustness and usability in real-world environments.

Overall, this project highlights the principles of digital security systems and demonstrates how hardware description languages like Verilog can be used to create functional and effective designs for controlled access systems.

References

1. IEEE Computer Society. IEEE Standard for Verilog Hardware Description Language. IEEE Std 1364-2005, IEEE, 2006.

2. Cavanagh, Joseph. Digital Design and Verilog HDL Fundamentals. CRC Press, 2008.

3. Brown, Stephen, and Zvonko Vranesic. Fundamentals of Digital Logic with Verilog Design. McGraw-Hill, 2007.

4. Verilog Tutorial on asic-world.com, [Online]. Available: https://www.asic-world.com/verilog/veritut.html