Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Design and Implementation of an FPGA-Based RFID Home Security System with Performance Analysis

Tejas Vinayak Kulkarni
School of E&TC Engineering
MIT Academy of Engineering
Pune, India

Kumar Hadole
School of E&TC Engineering
MIT Academy of Engineering
Pune, India

Nivrutti Bobade
School of E&TC Engineering
MIT Academy of Engineering
Pune, India

Navinya Mirase
School of E&TC Engineering
MIT Academy of Engineering
Pune, India

5th Rashmi Mahajan
School of E&TC Engineering
MIT Academy of Engineering
Pune, India

***Abstract* - This paper presents the design and implementation of an FPGA-based RFID home security system using the MFRC522 RFID reader and the Basys-3 FPGA platform. The objective of the system is to provide secure access control through RFID-based authentication using real-time hardware processing. The system is implemented using Verilog HDL and consists of a custom SPI communication module, a finite state machine (FSM) for RFID command handling, and a UID comparison logic.**

The design successfully achieved synthesis, implementation, and timing closure with positive slack using Xilinx Vivado. However, during hardware testing, the system failed to establish reliable communication with the RFID module. Issues related to SPI timing precision, constraint configuration, and protocol-level synchronization were identified as major causes of failure.

This paper presents a detailed analysis of the challenges encountered during hardware implementation and highlights the limitations of FPGA-based peripheral interfacing. The study provides valuable insights for future embedded system designs involving FPGA and communication modules.

I. INTRODUCTION

In recent years, the demand for secure and automated access control systems has increased significantly due to advancements in smart home technologies [11]. Traditional security systems based on mechanical keys suffer from limitations such as duplication, loss, and lack of real-time monitoring. RFIDbased systems provide a contactless and efficient solution for authentication and access control [1], [8].

Field Programmable Gate Arrays (FPGAs) are widely used in embedded system design due to their ability to perform parallel processing and provide deterministic timing [6], [10]. These features make them suitable for real-time applications such as security systems [12]. However, interfacing external devices such as RFID modules with FPGA introduces several practical challenges [13].

This paper presents the design and implementation of an RFID-based home security system using the MFRC522 RFID reader [2] and Basys-3 FPGA. While simulation and timing analysis showed correct functionality, hardware implementation revealed multiple challenges in communication and system integration.

The main objective of this work is not only to implement the system but also to analyze the reasons behind its failure and extract useful engineering insights.

II. SYSTEM ARCHITECTURE

A. Overview

The proposed system consists of an FPGA-based controller interfaced with an RFID reader module [1]. The system performs authentication by reading the UID of an RFID card and comparing it with a stored UID [8].

B. Hardware Components

The hardware components used in the system are listed in Table I.

TABLE I: Hardware Components

Component	Description
Basys-3 FPGA	Artix-7 FPGA used for processing
MFRC522 RFID Reader	RFID module using SPI communication
LED	Output indicator for authentication

C. Functional Modules

The system is divided into the following modules [10], [12]:

1) SPI Controller: Responsible for communication between FPGA and RFID module using SPI protocol [9].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

2) RFID Controller FSM: Controls the sequence of commands required for RFID communication [6], [7]. 3) UID Comparator: Compares received UID with stored UID [2].

4) Output Unit: Displays authentication result using LED.

D. System Operation

The operation of the system is as follows [1], [2]:

1) System is initialized and RFID module is configured.
2) REQA command is sent to detect RFID card.
3) Anti-collision command is used to retrieve UID.
4) UID is compared with stored value. 5) LED indicates access result.

E. RFID Controller Finite State Machine (FSM)

The operation of the RFID-based system is controlled using a Finite State Machine (FSM) [6], [7], which ensures proper sequencing of communication between the FPGA and the MFRC522 RFID module [2].

The FSM begins in the RESET state, where the RFID module is initialized. In this state, the system ensures that all internal registers and control signals are set to their default values before starting communication [2].

After initialization, the FSM transitions to the ANTENNA_ON state. In this state, the antenna of the RFID module is enabled to allow detection of nearby RFID tags [1]. This step is essential to activate the RF field required for communication.

Once the antenna is enabled, the FSM moves to the REQA state. In this state, a request command (REQA) is transmitted to detect the presence of an RFID card within the reader's range [1], [2].

If a card is detected, the FSM transitions to the WAIT_IRQ state. In this state, the system waits for an interrupt signal from the RFID module, indicating that the command execution has been completed and data is ready to be read [2].

After receiving the interrupt signal, the FSM proceeds to the READ state. In this state, the FPGA reads data from the RFID module, including the unique identifier (UID) of the RFID card [2], [8].

Finally, the FSM transitions to the DONE state, where the received UID is processed and compared with the stored UID for authentication [11]. Based on the comparison result, the system generates the appropriate output, such as enabling or disabling the LED indicator.

After completing the operation, the FSM returns to the initial state to allow detection of the next RFID card. This cyclic operation ensures continuous monitoring and real-time response of the system [12].

F. Hardware Setup

The hardware setup consists of the Basys-3 FPGA board interfaced with the MFRC522 RFID reader module using SPI communication [2], [13]. The connections include MOSI, MISO, SCLK, and chip select signals along with power and ground [9].

Fig. 1 shows the actual hardware implementation of the system.

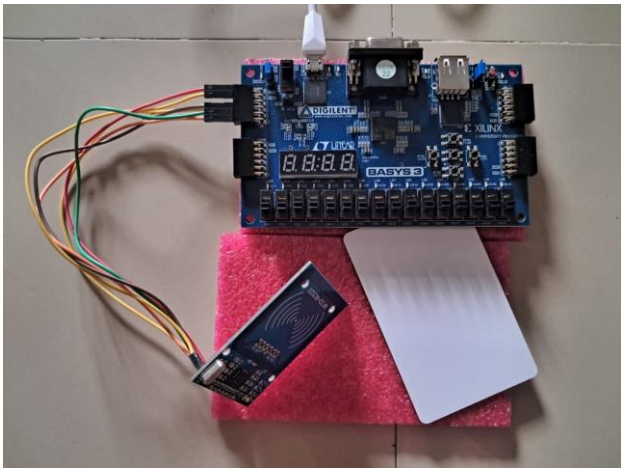


Fig. 1: Hardware Implementation of the FPGA-Based RFID Security System.

III. SPI COMMUNICATION DESIGN

The MFRC522 RFID module communicates with the FPGA using the Serial Peripheral Interface (SPI) protocol [2], [9]. SPI is a synchronous communication protocol that uses four signals: MOSI, MISO, SCLK, and CS [4], [9].

A. SPI Configuration

The SPI communication parameters used in this system are shown in Table II.

TABLE II: SPI Configuration

Parameter	Value
Mode	SPI Mode 0
Clock Frequency	~3 MHz
Data Width	8-bit
Clock Source	100 MHz FPGA clock

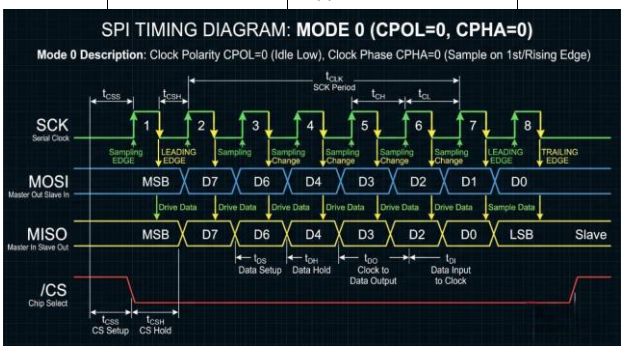


Fig. 2: SPI Timing Diagram for Mode 0 Operation (CPOL = 0, CPHA = 0).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Fig. 2 illustrates the SPI timing diagram for Mode 0 operation (CPOL = 0, CPHA = 0) used in the proposed system [9]. In this mode, the clock signal (SCLK) remains low when idle, and data transmission is synchronized with clock edges [4].

The Master Out Slave In (MOSI) line carries data from the FPGA to the RFID module, while the Master In Slave Out (MISO) line carries data from the RFID module back to the FPGA [2], [9].

Data on the MOSI line is updated on the falling edge of the clock and remains stable before the rising edge. The receiving device samples the data on the rising edge of the clock [9]. Similarly, data on the MISO line is sampled by the FPGA on the rising edge of SCLK.

Each SPI transaction consists of 8 clock cycles corresponding to one byte of data transfer [4]. The chip select (CS) signal is driven low to initiate communication and is held low throughout the transaction [2].

Accurate timing between SCLK, MOSI, and MISO is critical for reliable communication [13]. Any deviation in clock alignment or data stability can result in incorrect data transfer, which was observed as a major issue during hardware implementation.

B. SPI Simulation Results

To verify the correctness of the SPI communication logic, simulation was performed using the Vivado simulator [3]. The simulation waveform is shown in Fig. 3.

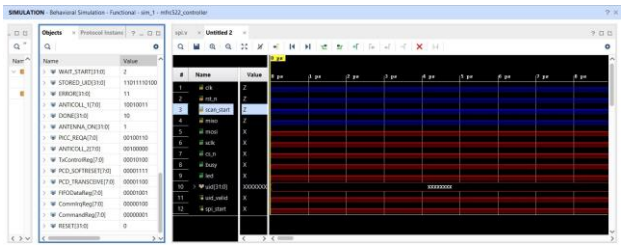


Fig. 3: SPI simulation waveform showing SCLK, MOSI, MISO, and spi_start signals during data transmission.

The waveform illustrates the behavior of key SPI signals including the serial clock (SCLK), Master Out Slave In (MOSI), Master In Slave Out (MISO), and the SPI start signal (spi_start) [9]. The SPI transaction is initiated when the spi_start signal is asserted, which triggers data transfer between the FPGA and the RFID module [2].

As observed in the waveform, the SCLK signal operates at a reduced frequency derived from the system clock [14]. The MOSI line carries transmitted data from the FPGA, while the MISO line represents received data from the RFID module [9]. Data is transmitted and received synchronously with the clock signal.

The waveform confirms that data on the MOSI line is stable before the rising edge of SCLK, and the data on the MISO line is sampled correctly on the rising edge [9]. This validates that the SPI communication follows Mode 0 timing requirements [2].

The simulation results demonstrate that the SPI module is functioning correctly at the logical level [3]. However, despite correct simulation behavior, hardware implementation revealed inconsistencies, indicating the presence of additional real-world constraints not captured during simulation [13].

C. SPI Implementation

The SPI controller was implemented using Verilog HDL [5], [6]. A clock divider was used to generate the SPI clock from the system clock [14]. Data transmission and reception were controlled using a shift register mechanism [13].

The SPI operation follows the sequence [2], [9]:

- Set chip select low
- Send register address
- Send or receive data
- Set chip select high

D. Challenges in SPI Implementation

Implementing SPI on FPGA requires precise control of timing signals [10], [13]. Unlike microcontrollers, FPGA does not provide built-in SPI modules, and the protocol must be implemented manually [6]. Minor timing errors can result in incorrect data transmission [4], [9].

IV. IMPLEMENTATION AND RESULTS

The design was implemented using Xilinx Vivado on the Basys-3 FPGA board [3].

A. Timing Analysis

The timing analysis results are shown in Table III.

TABLE III: Timing Results

Parameter	Value
Worst Negative Slack (WNS)	+6.326 ns
Total Negative Slack (TNS)	0 ns
Timing Violations	None

These results indicate that the design meets all timing requirements [3], [14].

B. Timing Analysis Report

To further validate the implementation results, the timing report generated by Xilinx Vivado is analyzed [3]. The timing summary is shown in Fig. 4.

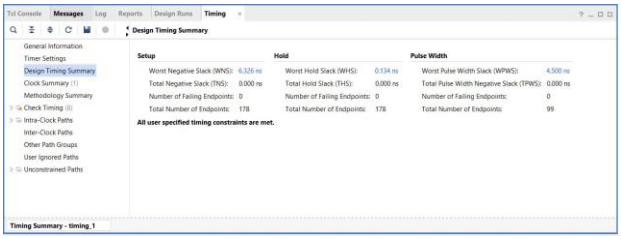


Fig. 4: Timing analysis report showing positive slack and successful timing closure.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The report indicates a Worst Negative Slack (WNS) of +6.326ns and Total Negative Slack (TNS) of 0ns, confirming that the design meets all timing constraints [3]. A positive slack value ensures that all signal paths satisfy the required setup and hold time conditions [14].

The absence of timing violations demonstrates that the clock distribution, logic design, and signal propagation are correctly implemented within the FPGA [10], [14]. This confirms that the system is operating reliably at the specified clock frequency.

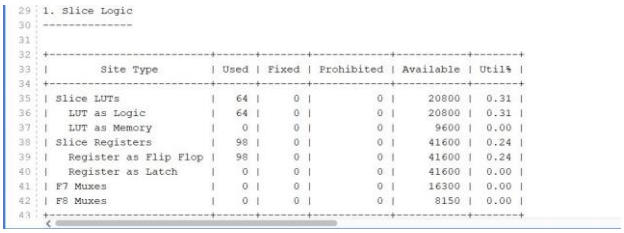
Despite achieving successful timing closure, the system failed to produce correct output during hardware testing. This highlights that meeting timing constraints alone does not guarantee functional correctness, especially in systems involving external communication modules such as RFID interfaces [12], [13].

C. Resource Utilization

TABLE IV: Resource Utilization

Resource	Usage
LUTs	~0.31%
Flip-Flops	~0.24%
Block RAM	~0%

The resource utilization of the proposed design was analyzed using the Vivado implementation report [3]. The summary of FPGA resource usage is shown in Fig. 5.



1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs	64	0	0	20800	0.31
LUT as Logic	64	0	0	20800	0.31
LUT as Memory	0	0	0	9600	0.00
Slice Registers	98	0	0	41600	0.24
Register as Flip Flop	98	0	0	41600	0.24
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

Fig. 5: FPGA resource utilization report showing LUT and register usage of the design.

The design utilizes a very small portion of the available FPGA resources. A total of 64 LUTs (0.31%) and 98 flipflops (0.24%) are used out of the available resources on the Artix-7 device [3]. No Block RAM (BRAM) or DSP slices are utilized in the design.

The low resource utilization indicates that the design is highly efficient and occupies minimal hardware resources [10]. This allows scope for further expansion of the system, such as adding additional features like GSM communication or multiple user authentication [12].

The results confirm that the proposed design is lightweight and suitable for implementation on low-cost FPGA platforms without resource constraints [3], [10].

D. Observations

The FPGA implementation was successful, and the system operated correctly in simulation [3]. However, hardware testing showed inconsistent behavior in RFID communication [2], [13].

TABLE V: Experimental Comparison

Function	Expected	Actual
SPI Communication	Stable	Unstable
UID Detection	Successful	Failed
Authentication	Working	Not Achieved

V. EXPERIMENTAL RESULTS

VI. FAILURE ANALYSIS

A. Constraint Issues

Initially, the design lacked proper I/O constraints, which prevented bitstream generation [3]. This issue was resolved by defining appropriate pin mappings and I/O standards [14].

B. Timing Issues

Clock and timing issues were observed during initial implementation [3], [14]. These were resolved by applying proper timing constraints [10].

C. SPI Communication Issues

Despite correct timing, SPI communication with the MFRC522 module failed due to strict protocol requirements [2], [9]. The MFRC522 requires adherence to specific SPI framing and register access conventions that go beyond standard SPI Mode 0 behavior [2].

D. FSM Design Issues

The FSM controlling RFID communication required precise synchronization with SPI signals [6], [7]. Initial design lacked proper synchronization, leading to unstable behavior [12], [13].

E. Root Cause

The main causes of failure are identified as [2], [10], [13]:

- Protocol-level incompatibility between FPGA SPI implementation and MFRC522 requirements [2], [9].
- Lack of precise timing alignment between SPI signals and RFID module expectations [13], [14].

VII. DISCUSSION

The results indicate that achieving timing closure alone does not guarantee correct system functionality [10], [12].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Realworld hardware introduces additional complexities such as signal integrity and protocol constraints [4], [13].

This highlights the importance of considering both hardware and protocol-level design aspects when developing FPGAbased systems that interface with external communication modules [2], [13], [15].

VIII. CONCLUSION

This paper presented the design and implementation of an FPGA-based RFID home security system [1], [2]. While the system achieved successful timing closure and implementation [3], it failed during hardware testing due to communication issues [2], [13].

The study provides valuable insights into FPGA-based system design [6], [10] and highlights challenges in interfacing external communication modules [12], [13]. Future work should focus on improving SPI protocol fidelity [9] and incorporating hardware-level debugging tools [3], [14].

REFERENCES

[1] K. Finkenzeller, *RFID Handbook*, 3rd ed., Wiley, 2010.
[2] NXP Semiconductors, *MFRC522 Standard Performance MIFARE Reader IC Datasheet*, 2016.
[3] Xilinx Inc., *Vivado Design Suite User Guide*, 2020.
[4] W. Stallings, *Data and Computer Communications*, 10th ed., Pearson, 2013.
[5] J. Bhasker, *A VHDL Primer*, 3rd ed., Prentice Hall, 1999.
[6] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, McGraw-Hill, 2009.
[7] J. F. Wakerly, *Digital Design: Principles and Practices*, 4th ed., Pearson, 2006.
[8] K. Domdouzis, B. Kumar, and C. Anumba, ”Radio-frequency identification (RFID) applications: A brief introduction,” *Advanced Engineering Informatics*, vol. 21, no. 4, pp. 350–355, 2007.
[9] A. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed., Pearson, 2011.
[10] C. Maxfield, *The Design Warrior’s Guide to FPGAs*, Elsevier, 2004.
[11] S. Garfinkel and B. Rosenberg, *RFID: Applications, Security, and Privacy*, Addison-Wesley, 2005.
[12] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*, Wiley, 2002.
[13] M. Barr and A. Massa, *Programming Embedded Systems*, O’Reilly, 2006.
[14] Xilinx Inc., *7 Series FPGA Clocking Resources User Guide*, 2018.
[15] H. Want, ”An introduction to RFID technology,” *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.