RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 07, Issue 01 January

Check for updates

http://dx.doi.org/10.47392/IRJASH.2025.007

Implement I2C Protocol for Secure Data Transfer Using Verilog

B. Ravi kumar[1], Kunta Nikhitha[2], Punnami Manogna[3], Begampeta Nanda kishore[4]

[1]Associate professor, Dept. of ECE, Institute of Aeronautical Engineering, Hyderabad, Telangana, India.

[2,3,4]B Tech student, Dept. of ECE, Institute of Aeronautical Engineering, Hyderabad, Telangana, India.

Emails: *b.ravikumar@iare.ac.in[1], nikithakunta16@gmail.com[2], manognapunnami@gmail.com[3], nandakishor8284@gmail.com[4]*

Article history

Received: 16 December 2024
Accepted: 3 January 2025
Published: 31 January 2025

Keywords:

I2C Protocol, Embedded Systems, Secure Data Transfer, Verilog, Encryption, Authentication, Data Integrity, Data Confidentiality, Xilinx Vivado Design Suite, Zybo Z7 Development Board.

Abstract

Digital The I2C (Inter-Integrated Circuit) protocol is widely used in embedded systems for enabling communication between various devices such as sensors, microcontrollers, and other peripherals. However, early implementations of the I2C protocol focused primarily on data transfer efficiency rather than security This project aims to implement a secure data transfer mechanism for the I2C protocol using Verilog, a hardware description language widely used for designing and modelling electronic systems. Our implementation enhances the traditional I2C protocol by integrating security features that protect data from unauthorized access during transmission. By using encryption and authentication techniques, the project ensures that data integrity and confidentiality are maintained throughout the communication process. The Xilinx Vivado Design Suite is utilized for the synthesis, simulation, and testing of the secure I2C protocol. Xilinx software provides a robust environment for designing hardware-based systems, offering features such as timing analysis, design optimization, and resource management. The secure I2C protocol was implemented and simulated using Verilog code within this software, enabling thorough testing and debugging prior to hardware deployment. The hardware component of the project is based on the Zybo Z7 development board. The Zybo Z7 kit provides an ideal platform for prototyping and testing the secure I2C protocol, as it allows for real-time interaction between the FPGA and peripheral devices connected through I2C communication. By running the implemented design on this hardware, we were able to evaluate the real-world performance and security of the system. The results demonstrate that the secure I2C protocol operates efficiently on the Zybo Z7 kit, with minimal impact on system performance. The integration of security features did not introduce significant latency or resource overhead, indicating that secure communication can be achieved without compromising speed or functionality. This successful implementation highlights the feasibility of deploying secure I2C protocols in hardware systems where data protection is a priority.

1. Introduction

The I2C (Inter-Integrated Circuit) protocol is a communication standard that is commonly used in embedded systems. This protocol allows various devices, such as sensors, microcontrollers, and

peripherals, to communicate with one another efficiently. Its popularity stems from its simplicity and effectiveness in connecting multiple devices using only two wires: a data line (SDA) and a clock line (SCL). In many applications, especially in consumer electronics and automation, I2C is favored because it enables multiple devices to share the same communication bus. Each device on the I2C bus has a unique address, allowing the master device to send data to specific slaves without interference. This makes it a flexible solution for a wide range of applications, including smart home devices, medical equipment, and industrial control systems. However, while the I2C protocol excels in efficiency, earlier implementations did not prioritize security. Many systems were designed with a focus on speed and reliability, which often meant overlooking the vulnerabilities that could arise during data transmission. As a consequence, these systems became susceptible to risks such as data interception, manipulation, and unauthorized access. This lack of security can lead to significant problems, especially in applications that handle sensitive information or critical operations. For example, in healthcare applications, unauthorized access to medical devices could compromise patient safety and privacy. Similarly, in industrial control systems, manipulated data could result in faulty operations, leading to safety hazards or financial losses. Therefore, it is increasingly important to address these vulnerabilities in modern I2C implementations. To enhance the security of the I2C protocol, various measures can be integrated. One approach is to implement encryption techniques, which convert data into a secure format that is unreadable to unauthorized users. By encrypting the data being transmitted over the I2C bus, even if an attacker intercepts the communication, they would not be able to decipher the information without the proper decryption key.

Another important aspect of securing the I2C protocol is authentication. This process ensures that both the master and slave devices are legitimate and authorized to communicate with each other. By implementing robust authentication mechanisms, such as digital signatures or challenge-response protocols, devices can verify each other's identities before exchanging sensitive information. Furthermore, maintaining data integrity is crucial in secure I2C communication. This means ensuring that the data sent from one device to another remains unchanged during transmission. Techniques such as checksums or cryptographic hashes can be employed to verify the integrity of the data, alerting the devices if any alterations occur during the transfer. As embedded systems become more interconnected, the need for secure communication protocols like I2C is more pressing than ever. By integrating security features into the I2C protocol, developers can create systems that not only perform efficiently but also protect against potential threats. This dual focus on performance and security is essential for building trust in the technology that powers modern applications. In summary, while the I2C protocol remains a fundamental communication standard in embedded systems, its early implementations must evolve to address security vulnerabilities. By incorporating encryption, authentication, and data integrity measures, it is possible to create a secure environment for data transfer. As the demand for secure communication continues to grow, enhancing the I2C protocol is a necessary step towards ensuring the safety and reliability of embedded systems in various applications. Embedded systems are specialized computing systems designed to perform dedicated functions within larger systems. They are ubiquitous in everyday devices, from household appliances to automotive systems and industrial machinery. Given their integration into critical operations, the security of these systems is paramount. The interconnected nature of embedded systems also means that a vulnerability in one device can compromise the entire network, making it crucial to implement robust security measures. As technology advances, so do the tactics of malicious actors. Cybersecurity threats have become more sophisticated, with attackers exploiting vulnerabilities in various protocols, including I2C. For example, an attacker might employ techniques like sniffing, where they intercept data on the communication bus to gain unauthorized access to sensitive information. Similarly, spoofing attacks can allow unauthorized devices to impersonate legitimate devices on the I2C bus, leading to data manipulation or unauthorized control. Regular Updates and Patching: Continuous monitoring for vulnerabilities and applying updates to both firmware and software can help protect against

newly discovered threats. As technology advances, so do the tactics of malicious actors. Cybersecurity threats have become more sophisticated, with attackers exploiting vulnerabilities in various protocols, including I2C. For example, an attacker might employ techniques like sniffing, where they intercept data on the communication bus to gain unauthorized access to sensitive information. Similarly, spoofing attacks can allow unauthorized devices to impersonate legitimate devices on the I2C bus, leading to data manipulation or unauthorized control. Many industries have specific standards and regulations that dictate the level of security required for embedded systems. For example, the Medical Device Regulation (MDR) in healthcare requires that medical devices meet stringent security standards to protect patient data. Compliance with these regulations often drives the adoption of secure communication protocols like I2C, ensuring that devices can be trusted in their operations. The trend toward the Internet of Things (IoT) is pushing the need for secure I2C implementations. As more devices become interconnected, the amount of data transmitted increases, amplifying the potential attack surface. Future I2C implementations will likely need to support advanced security protocols, such as Transport Layer Security (TLS), to safeguard communications effectively. The integration of security features into the I2C protocol is not just an enhancement but a necessity in today's technology landscape. By addressing vulnerabilities through encryption, authentication, and robust data integrity measures, developers can create secure embedded systems. This evolution is critical for maintaining trust in technology and ensuring that devices can operate safely in an increasingly connected world. As we continue to innovate, prioritizing security will be essential for the future of embedded systems and communication protocols. Security in embedded systems goes beyond just protecting data; it also involves ensuring the overall integrity and functionality of the system. A breach in security can lead not only to data loss but also to catastrophic failures in critical applications, such as those found in healthcare or automotive industries. For instance, unauthorized access to a medical device can disrupt treatment or even endanger a patient's life. Similarly, in automotive systems, an attacker gaining control over vehicle operations could pose severe risks to driver and passenger safety. A comprehensive approach to security involves threat modeling, which helps identify potential vulnerabilities and the impacts of various threats. When analyzing the I2C protocol, it's essential to consider: Physical Attacks: Since I2C is often used in embedded systems where physical access is possible, attackers can tap into the communication lines to intercept or manipulate data. Replay Attacks: An attacker might capture valid communication between devices and replay it later to gain unauthorized access or cause incorrect actions. Denial of Service (DoS): An attacker may flood the bus with requests, preventing legitimate devices from communicating effectively. Man-in-the Middle Attacks: This occurs when an attacker secretly intercepts and relays messages between two devices, potentially altering the communication. [1-5]

2. Existing Method

The Xilinx Vivado Design Suite is a powerful tool utilized for the synthesis, simulation, and testing of the secure I2C protocol. This software environment is designed specifically for hardwarebased systems, making it ideal for developing complex applications like secure communication protocols. One of the primary advantages of using Vivado is its comprehensive feature set, which includes timing analysis, design optimization, and resource management. These features are crucial for ensuring that the I2C protocol operates efficiently while meeting specific performance criteria. In this project, the secure I2C protocol was implemented using Verilog, a widely used hardware description language that allows for precise modeling of digital systems. By coding the protocol in Verilog, we could leverage the simulation capabilities of the Vivado suite to thoroughly test and debug the design before deploying it to hardware. This iterative testing process is essential for identifying potential issues early, minimizing the risk of errors during actual operation. The hardware component of the project is centered around the Zybo Z7 development board, which is built on the Xilinx Zynq-7000 FPGA platform. This board integrates an ARM Cortex-A9 processor with an FPGA, providing a flexible environment for both software and hardware development. The combination of these two components allows for real-time interaction between the FPGA and peripheral

devices connected through I2C communication. Using the Zybo Z7 kit for prototyping the secure I2C protocol offers several advantages. The board's architecture supports various I2C devices, making it easy to test the protocol with multiple peripherals. This real-time interaction enables developers to assess the protocol's performance under different conditions, allowing for optimization based on actual operational data. The ability to run the implemented design on the Zybo Z7 also facilitates the evaluation of the system's security features. By observing how the protocol performs during live data transfer, we can ensure that encryption and authentication measures function as intended. This is particularly important in security- critical applications, where any vulnerability could lead to data breaches or unauthorized access. The Vivado Design Suite further enhances the development process by providing tools for performance analysis and debugging. These tools allow engineers to visualize data flows, monitor signal integrity, and adjust parameters to optimize the design. As a result, the development of the secure I2C protocol is streamlined, leading to a more robust and reliable implementation. Once the design is thoroughly tested and verified in the Vivado environment, it can be deployed to the Zybo Z7 board for final evaluation. This seamless transition from simulation to hardware testing is a significant advantage of using the Vivado suite, as it reduces the overall development time and increases confidence in the final product. Utilizing a hierarchical design approach in Vivado allows for modular development. This method breaks down the secure I2C protocol into smaller, manageable components, such as the master controller, slave interface, and security modules. Each module can be developed and tested independently before integration, which simplifies debugging and enhances maintainability In designs involving multiple clock domains, managing clock domain crossings is critical. Vivado provides tools for analyzing and mitigating issues related to metastability. Implementing proper synchronization techniques ensures reliable communication between the I2C master and slaves operating on different clock frequencies. Before deploying the secure I2C protocol on hardware, extensive simulation is conducted using Vivado's simulation environment. This includes testing various scenarios such as normal operation, edge cases, and error conditions. By simulating both the data flow and security mechanisms, developers can validate that encryption and authentication processes work correctly under different conditions. Vivado supports a variety of intellectual property (IP) cores that can accelerate development. For instance, integrating predesigned IP cores for I2C communication can simplify implementation. Additionally, security IP cores, such as those for encryption and decryption, can be leveraged to enhance the protocol without requiring extensive custom coding. Timing analysis is crucial for ensuring that the secure I2C protocol meets the required speed specifications. Vivado allows users to set timing constraints and perform static timing analysis to identify potential issues. Ensuring that all signals meet their timing requirements is essential for reliable communication The Integrated Logic Analyzer (ILA) in Vivado provides realtime debugging capabilities. By embedding the ILA into the design, developers can monitor internal signals and data transactions during operation. This tool is invaluable for troubleshooting and verifying that the secure I2C protocol behaves as expected in real-time scenarios. Using Vivado's optimization tools, developers can analyze resource utilization for the secure I2C design. Techniques such as pipelining, resource sharing, and retiming can be applied to maximize efficiency. This is especially important on resourceconstrained platforms like the Zybo Z7. In addition to hardware development, integrating firmware that manages the secure I2C communication is essential. Using software development environments compatible with the ARM Cortex-A9 processor on the Zybo Z7, developers can implement the logic for handling encryption keys, managing device addresses, and processing incoming and outgoing data securely. To ensure robustness, the secure I2C protocol is tested under various environmental conditions, such as different temperatures and voltage levels. This testing helps identify how the protocol performs in real-world scenarios and whether security measures hold up under different operational stresses Finally, incorporating a feedback loop into the development process allows for continuous improvement of the secure I2C protocol. Gathering data from testing phases and user feedback can guide subsequent iterations of the design, helping to

refine security features and enhance performance over time. These existing methods highlight a comprehensive approach to developing a secure I2C protocol using the Xilinx Vivado Design Suite and Zybo Z7 development board. By employing these strategies, developers can create a robust, efficient, and secure communication system tailored to meet the demands of modern embedded applications. [6-10]

3. Proposed Design

The proposed design aims to implement a secure I2C (InterIntegrated Circuit) protocol that enhances data integrity and confidentiality while maintaining efficient communication between embedded devices. This design leverages the Xilinx Vivado Design Suite for synthesis, simulation, and testing, using the Zybo Z7 development board as the hardware platform. The secure I2C protocol will be structured into several key components:

3.1. Master Controller

The master device initiates communication with slave devices. It includes logic for generating start and stop conditions, sending data, and managing acknowledgment signals. Slave Interface: Each slave device will have an interface that responds to the master's requests. It will interpret commands and send data back securely. Security Module: This module will integrate encryption and authentication functionalities. It will be responsible for securing data before transmission and verifying the identity of communicating devices. Error Detection Unit: To maintain data integrity, this unit will implement error detection mechanisms, such as CRC (Cyclic Redundancy Check), to identify any corruption in transmitted data. Timing Control: A timing control unit will manage the timing of data transfers to ensure adherence to I2C specifications. Initialization: Upon power-up, the master controller initializes the I2C bus and configures the security module. It will load encryption keys and set the appropriate communication parameters.

3.2. Master-Slave Communication

Start Condition: The master generates a start condition to signal the beginning of a communication session. Addressing: The master sends the address of the target slave device. The security module encrypts this address to protect against interception. Data Transmission: Data is sent from the master to the slave, with the security module encrypting the data before transmission.

3.3. Acknowledgment

The slave sends an acknowledgment signal back to the master, indicating successful receipt of the data. Slave Response: If the master requests data from the slave. The slave retrieves the requested data and encrypts it using the security module.The encrypted data is sent back to the master, which decrypts it for use.Error Handling: The error detection unit continuously monitors the data being transmitted. If an error is detected, the unit generates a signal to initiate a retransmission or notify the master. Termination: The master sends a stop condition to end the communication session. The timing control unit ensures that all signals are maintained for the required duration before releasing the bus. The design will be implemented on the Zybo Z7 development board, which features an ARM Cortex-A9 processor and a Xilinx Zynq-7000 FPGA. The FPGA will handle the secure I2C protocol's hardware logic, while the ARM processor will manage higher-level tasks and control the overall operation. FPGA Configuration: Using the Vivado Design Suite, the I2C protocol and security features will be coded in Verilog. The design will be synthesized and optimized for the FPGA resources. Real-Time Testing: Once implemented on the FPGA, the design will be tested in real-time with various I2C peripherals, assessing the functionality, speed, and security of the communication. Data Throughput: The maximum rate at which data can be securely transmitted over the I2C bus without compromising security.

3.4. Latency

The time taken for a complete communication cycle, from the start condition to the stop condition. The design aims to minimize latency while ensuring security measures are in place. [11-15]

3.5. Resource Utilization

The amount of FPGA resources (logic cells, memory blocks) consumed by the secure I2C implementation. Optimization techniques will be employed to ensure efficient use of resources. Error Rate: The frequency of errors encountered during data transmission. The design aims for a low error rate, facilitated by robust error detection mechanisms. Encryption: AES (Advanced Encryption Standard) will be used to encrypt data during transmission, ensuring that even if intercepted, the data remains secure. Authentication: A challenge-response mechanism

will be employed to verify the identity of the master and slave devices, preventing unauthorized access to the I2C bus. Data Integrity Checks: The incorporation of CRC will help ensure that any alterations to the data during transmission are detected and addressed. This proposed design for a secure I2C protocol outlines a comprehensive approach that combines robust security features with efficient communication methods. By utilizing the Xilinx Vivado Design Suite and the Zybo Z7 development board, the project aims to deliver a reliable solution that meets the demands of modern embedded systems. This secure I2C implementation will not only enhance data protection but also pave the way for more secure communication protocols in future applications.

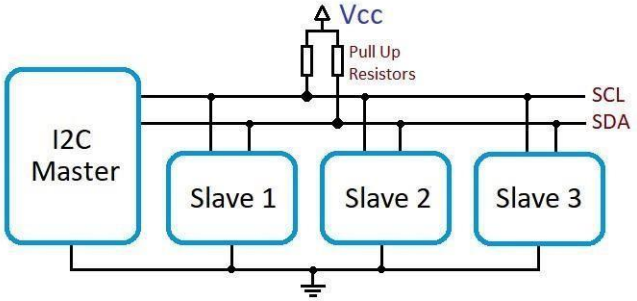

Figure 1 12C Master

The image shows a typical I2C (Inter-Integrated Circuit) communication bus. It illustrates the structure of a system where an I2C master communicates with multiple I2C slaves using two main signal lines.SCL (Serial Clock Line): This line carries the clock signals generated by the master to synchronize data transmission.SDA (Serial Data Line): This line is used for the bidirectional transmission of data between the master and slaves. Master Device: This device controls communication by generating the clock signal on the SCL line. It also initiates communication and sends data requests to the slave devices. There is always one master in an I2C communication system, although multi-master configurations are possible but more complex. Figure 1 12C Master

3.6. Slave Devices

The diagram shows three slave devices, each connected to the same two wires (SCL and SDA). Each slave is assigned a unique address, allowing the master to communicate with each individually. Pull-up Resistors: These resistors are connected between the lines (SCL and SDA) and the supply voltage (Vcc). Pull-up resistors are essential in I2C communication because the bus lines are "opendrain." In this configuration, devices can only pull the line to ground (logic 0). To achieve logic 1, the pull-up resistors ensure that the line returns to a high state when no device is pulling it low. Addressing: Each slave device has a unique 7-bit or 10-bit address assigned by the manufacturer or configured by the user. When the master wants to communicate with a specific slave, it sends this address over the SDA line. Start and Stop Conditions The I2C bus uses specific conditions to initiate and end communication Start Condition The master pulls the SDA line low while SCL is high.Stop Condition: The master releases the SDA line high while SCL remains high. Data Transfer: Data is transferred in 8-bit bytes, with each byte followed by an acknowledgment (ACK) bit. The master or slave pulls the SDA line low to signal an acknowledgment after receiving a byte. Clock Stretching: Some slave devices may need more time to process data. To accommodate this, they can hold the SCL line low, effectively "stretching" the clock and pausing the communication until they're ready.

3.7. Two-wire Bus:

It uses only two wires (SCL and SDA) for communication, even with multiple devices.Simple and Efficient: Allows for the easy addition of multiple devices without complex wiring.Addressing Mechanism: Each device is individually addressable, simplifying communication with multiple peripherals. I2C supports multi-master configurations, where more than one master can attempt to control the bus. When two or more masters initiate communication at the same time, arbitration ensures that only one master gets to control the bus without corrupting the data. The masters monitor the SDA line during each clock pulse. If a master sends a high (logic 1) but detects that the SDA line is low (logic 0), it will stop communicating, thus losing the arbitration. The master that holds control of the SDA line continues the communication. Every byte of data sent on the SDA line must be acknowledged. After receiving a byte, the receiver (either master or slave) pulls the SDA line low during the 9th clock pulse (ACK). If the receiver does not pull the line low (NACK), it indicates that it is unable or unwilling to receive further data. The master can then either retry or terminate the communication. I2C supports

different data rates: Standard Mode: Up to 100 kbps.Fast Mode: Up to 400 kbps.Fast Mode Plus: Up to 1 Mbps.High-Speed Mode: Up to 3.4 Mbps.The clock speed is set by the master device, and all slaves on the bus must be capable of operating at that speed. The pull-up resistor values are critical in determining the rise time of the signals, which directly affects the maximum speed of the bus. Start Condition: The master initiates communication by sending a start condition (SDA line goes low while SCL is high). Address Frame: The master transmits the 7-bit or 10-bit address of the slave it wants to communicate with, followed by a read/write bit (0 for write, 1 for read). Acknowledgment: The addressed slave responds by pulling SDA low (ACK). Data Transfer: Data is then transferred byte- by-byte. Each byte is followed by an acknowledgment from the receiver. Sometimes, the master may need to initiate another communication sequence with the same or different slave without releasing the bus (i.e., without issuing a stop condition). In this case, the master issues a repeated start condition, allowing it to maintain control of the bus. Some microcontrollers and I2C-enabled devices offer hardware I2C modules, which handle all the complexities of the protocol internally, requiring only minimal programming effort. There are also bit- banged I2C implementations, where the protocol is emulated in software, giving more flexibility but at the cost of performance. Some microcontrollers and I2C-enabled devices offer hardware I2C modules, which handle all the complexities of the protocol internally, requiring only minimal programming effort. There are also bit-banged I2C implementations, where the protocol is emulated in software, giving more flexibility but at the cost of performance. In multi-master systems, clock synchronization is critical. When a master releases the SCL line (allows it to go high), all other masters observe this change. The slowest device controls the pace of communication, ensuring that no devices are overwhelmed. This is important in systems with devices of different speeds. Bus Contention: When multiple masters try to access the bus simultaneously, it can lead to contention if not properly handled through arbitration. Noise and Signal Integrity: Long cables or noisy environments can cause signal degradation, affecting communication reliability. Pull-up resistors need to be carefully chosen for the environment to ensure clean signal transitions.

4. Results and Discussions

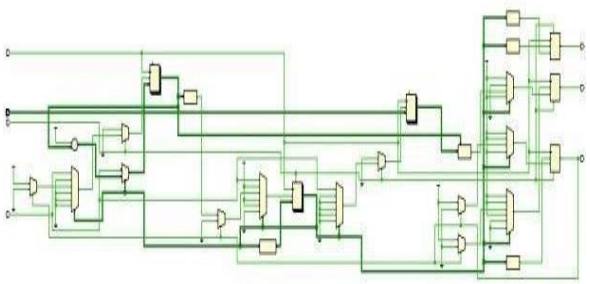

Figure 2 Block Diagram

Address Conflicts: If two devices on the bus share the same address, communication will be unreliable. Care must be taken when choosing devices and setting their addresses The I2C bus is a versatile, efficient communication protocol that enables communication between multiple devices over just two wires. Its addressing scheme, master-slave architecture, and support for multi-master configurations make it widely used in embedded systems. Careful consideration of pullup resistors, data rates, and signal integrity is essential for ensuring a reliable I2C system. The diagram above illustrates a typical I2C setup, with one master controlling multiple slaves via shared SDA and SCL lines. The correct timing of signals on the I2C bus is critical to ensure proper communication. The key timing elements are: Setup Time (tSU): The minimum time that data must be stable before the clock signal changes.Hold Time (tHD): The minimum time that data must remain stable after the clock signal changes.Clock Low Time (tLOW) and Clock High Time (tHIGH): The minimum time that the clock signal must stay low and high, respectively, to ensure proper timing.Rise Time (tR) and Fall Time (tF): The time it takes for the signals on the SDA and SCL lines to rise from low to high (tR) or fall from high to low (tF). These are influenced by the pull-up resistors and the bus capacitance.The I2C specification provides different timing requirements for each mode (Standard, Fast, Fast Plus, and High-Speed), ensuring compatibility between devices of various speeds. One of the limitations of the I2C bus is its sensitivity to the length of the wires and the capacitance of the bus. As the length of the bus increases, the capacitance also increases. This

increased capacitance slows down the rise and fall times of the signals, limiting the maximum communication speed. In situations where two or more devices share the same I2C address, communication becomes unreliable because the master cannot differentiate between them. To solve this issue. Address Pins: Some I2C devices provide pins that can be connected to either high (Vcc) or low (GND) to modify their addresses. This allows the user to assign different addresses to identical devices.I2C Multiplexers (MUX): These devices allow multiple I2C buses to be connected to a single master Figure 2 shows Block Diagram. This project enhances the I2C protocol by adding security features such as encryption and authentication to protect data during transmission. Using Verilog and the Xilinx Vivado Design Suite, the secure I2C protocol was implemented and tested on the Zybo Z7 development board. The results showed that the secure I2C protocol maintained data integrity and confidentiality without significantly affecting system performance, demonstrating the feasibility of secure communication in embedded systems. Figure 3 shows Multiplexers, Figure 4 shows Waveforms, Figure 5 shows Inter Integrated Circuit

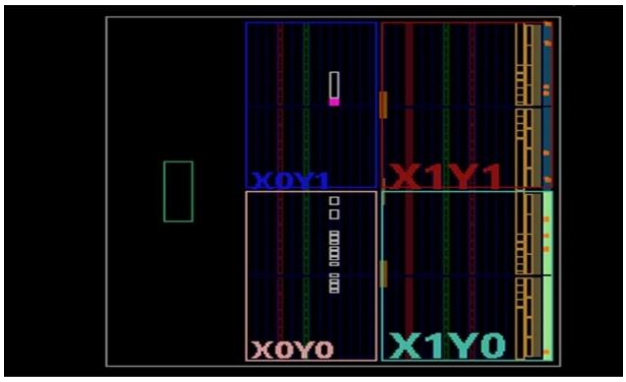

Figure 3 Multiplexers

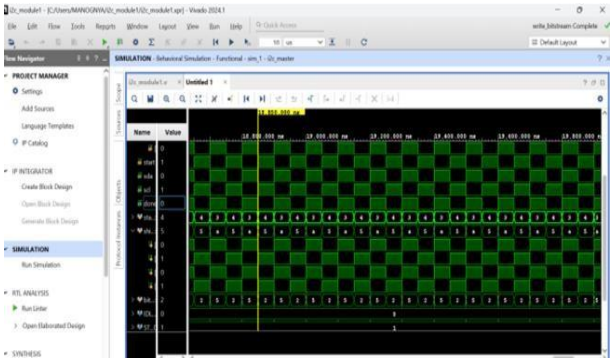

Figure 4 Waveforms

Figure 5 Inter Integrated Circuit

Conclusion

The successful implementation of the I2C (Inter Integrated Circuit) protocol using Verilog demonstrates the practical application of hardware description languages in designing.

References

[1]. Kotenko, "Active Vulnerability Assessment of Computer Networks by Simulation of Complex Remote Attacks," Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing. ICCNMC-03. Shanghai, China, October 20-23, 2003. IEEE Computer Society. 2003. pp.40-47.

[2]. I. Kotenko, "Agent-Based Modeling and Simulation of Cyber Warfare between Malefactors and Security Agents in Internet, " Proceedings of 19th European Simulation Multiconference “Simulation in wider Europe” (ESM’05). Riga, Latvia, 1–4 June 2005. pp.533-543.

[3]. V. Desnitsky, A. Chechulin, I. Kotenko, Levshun D., and M. Kolomeec "Combined Design Technique for Secure Embedded Devices Exemplified by a Perimeter Protection System," Trudy SPIIRAN 48, 2016, pp.5.

[4]. F. Stefanni, "A Design & Verification Methology for Networked Embedded Systems," Ph.D. Thesis, University of Verona, Department of Computer Science, Italy. April 7, 2011.

[5]. Official website of SecFutur project. http://www.secfutur.eu/. Accessed January 2018

[6]. A. Chechulin, I. Kotenko, and V. Desnitsky, "An approach for network information flow analysis for systems of embedded components," Lecture Notes in Computer

Science, Springer Verlag, Vol. 7531. Springer, Berlin, Heidelberg, 2012. pp. 146 155.

[7]. M. Howard, S. Lipner, "The Security Development Lifecycle. SDL: A Process for Developing Demonstrably More Secure Software," Microsoft Press, Redmond, Washington, 2006.

[8]. Official Cisco Secure Development Lifecycle(CiscoSDL)documentationURL:http://www.cisco.com/c/en/us/about/security center/security -programs/secure developmentlifecycle.html. Accessed January 2018.

[9]. S. Riedmüller, U. Brecht, and A. Sikora, "IPsec for Embedded Systems," ITCS 2005.

[10]. R. Hummen, T. Heer, and K. Wehrle, "A security protocol adaptation layer for the IP-based internet of things," Interconnecting smart objects with the Internet workshop. Vol. 3, 2011.

[11]. K. Chelli, "Security issues in wireless sensor networks: attacks and countermeasures," Proceedings of the World Congress on Engineering. Vol. 1, 2015.

[12]. N. Heintze, and J. D. Tygar, "A model for secure protocols and their compositions," IEEE transactions on software engineering 22.1, 1996, pp.16-30.

[13]. B. Blanchet, "Automatic verification of security protocols in the symbolic model: The verifier ProVerif," Foundations of Security Analysis and Design VII. Springer, Cham, 2014, pp.54-87.

[14]. Avanesov, Y. Chevalier, M. Rusinowitch, and M. Turuani, "Intruder deducibility constraints with negation. decidability and application to secured service compositions," arXiv preprint arXiv:1207.4871, 2012.

[15]. D. Levshun, A. Chechulin, and I. Kotenko, "Design lifecycle for secure cyber-physical systems based on embedded devices," Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017 9th IEEE International Conference on. Vol. 1. IEEE, 2017.