T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

IMPLEMENTING COMMUNICATIONS PROTOCOLS ON FPGA: ONE WIRE, SPI, VGA

EMRE YAVUZ

SUPERVISOR
DR. ÖĞR. ÜYESI ALP ARSLAN BAYRAKCI

GEBZE
2025

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

IMPLEMENTING COMMUNICATIONS PROTOCOLS ON FPGA: ONE WIRE, SPI, VGA

EMRE YAVUZ

SUPERVISOR
DR. ÖĞR. ÜYESI ALP ARSLAN BAYRAKCI

2025
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 02/10/2024 by the following jury.

JURY

Member
(Supervisor) : Dr. Öğr. Üyesi Alp Arslan Bayrakci

Member : Prof. Erkan Zergeroğlu

ABSTRACT

This project focuses on implementing a versatile environmental monitoring and control system using FPGA and Arduino platforms. The DS18B20 temperature sensor is interfaced with the FPGA via the One-Wire communication protocol, enabling real-time temperature measurement displayed on a 7-segment LED. An RGB LED connected to the FPGA visually indicates whether the temperature is below, within, or above user-defined thresholds by displaying blue, green, or red, respectively. These thresholds are dynamically set through a touchscreen interface connected to an Arduino.

The Arduino communicates with the touchscreen using SPI and relays user inputs to the FPGA via the I2C protocol. Simultaneously, temperature readings from the FPGA are sent to the Arduino over I2C, allowing the touchscreen to display real-time temperature data. Additionally, a VGA module is integrated with the FPGA to present system status and user data visually on a 640x480 resolution display.

The results demonstrate seamless integration of One-Wire, SPI, I2C, and VGA communication protocols, showcasing the system's capability to efficiently handle real-time data acquisition, processing, and visualization. This project highlights the FPGA's potential as a robust control and visualization platform in complex embedded systems.

Keywords: FPGA, DS18B20, One-Wire, I2C, SPI, VGA, Environmental Monitoring, Arduino.

ÖZET

Bu proje, FPGA ve Arduino platformları kullanılarak çok yönlü bir çevresel izleme ve kontrol sistemi geliştirilmesine odaklanmaktadır. DS18B20 sıcaklık sensörü, FPGA ile One-Wire iletişim protokolü aracılığıyla entegre edilerek anlık sıcaklık ölçümü sağlanmış ve bu değerler 7 segment LED üzerinde görüntülenmiştir. FPGA'ya bağlanan bir RGB LED, sıcaklık değerinin, kullanıcı tarafından belirlenen eşik değerlerinin altında, içinde veya üstünde olup olmadığını sırasıyla mavi, yeşil veya kırmızı renklerle görsel olarak göstermektedir. Bu eşik değerleri, bir Arduino'ya bağlı dokunmatik ekran arayüzü aracılığıyla dinamik olarak ayarlanabilmektedir.

Arduino, dokunmatik ekran ile SPI protokolünü kullanarak iletişim kurar ve kullanıcı girişlerini I2C protokolü aracılığıyla FPGA'ya aktarır. Aynı zamanda, FPGA'dan alınan sıcaklık verileri I2C üzerinden Arduino'ya gönderilerek dokunmatik ekranda anlık sıcaklık bilgisinin gösterilmesi sağlanır. Ayrıca, FPGA ile entegre edilen bir VGA modülü, sistem durumu ve kullanıcı verilerini 640x480 çözünürlüklü bir ekranda görsel olarak sunmaktadır.

Sonuçlar, One-Wire, SPI, I2C ve VGA iletişim protokollerinin sorunsuz bir şekilde entegre edildiğini ve sistemin gerçek zamanlı veri toplama, işleme ve görselleştirme yeteneklerini başarıyla yerine getirdiğini göstermektedir. Bu proje, FPGA'nın karmaşık gömülü sistemlerde güçlü bir kontrol ve görselleştirme platformu olarak potansiyelini vurgulamaktadır.

Anahtar Kelimeler: FPGA, DS18B20, One-Wire Protokolü, I2C, SPI, VGA, Çevresel İzleme, Arduino.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to everyone who has guided, supported, and inspired me throughout this project. In particular:

My advisor, Alp Arslan Bayrakçı, for his invaluable knowledge and patience, Barış Batuhan Bolat, Batuhan Yanar, and Enes Akar, for their contributions and support to the project, My mother, father, brother, for their unwavering love and encouragement, All of my teachers, past and present, who have shaped my knowledge and skills, And finally, ChatGPT, for assisting me with technical questions and supporting me in the writing process. Your support has been a tremendous source of motivation for me during this journey. I am truly grateful to each and every one of you.

Name Surname

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
FPGA	:	Field Programmable Gate Array
DS18B20	:	Digital Temperature Sensor using One-Wire Protocol
RGB LED	:	Red-Green-Blue Light Emitting Diode
I2C	:	Inter-Integrated Circuit Protocol
SPI	:	Serial Peripheral Interface Protocol
VGA	:	Video Graphics Array
Arduino	:	Open-source microcontroller platform
μC	:	Microcontroller
7-Segment LED	:	Digital display with 7 segments for numeric output
T_{min}	:	Minimum temperature threshold
T_{max}	:	Maximum temperature threshold
T	:	Real-time temperature

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This project focuses on the development and integration of multiple communication protocols to enable real-time environmental monitoring and control using FPGA and Arduino platforms. The use of FPGA allows for precise timing and efficient handling of data from various sensors and peripherals. The system integrates components such as temperature sensors, RGB LEDs, touchscreens, and VGA displays. 2.3.

1.1. System Overview

The primary objective is to utilize an FPGA to interface with a DS18B20 temperature sensor using the One-Wire protocol. The measured temperature is displayed on a 7-segment LED. Additionally, the FPGA controls an RGB LED to indicate temperature thresholds, which are dynamically set via a touchscreen. The color representation is as follows:

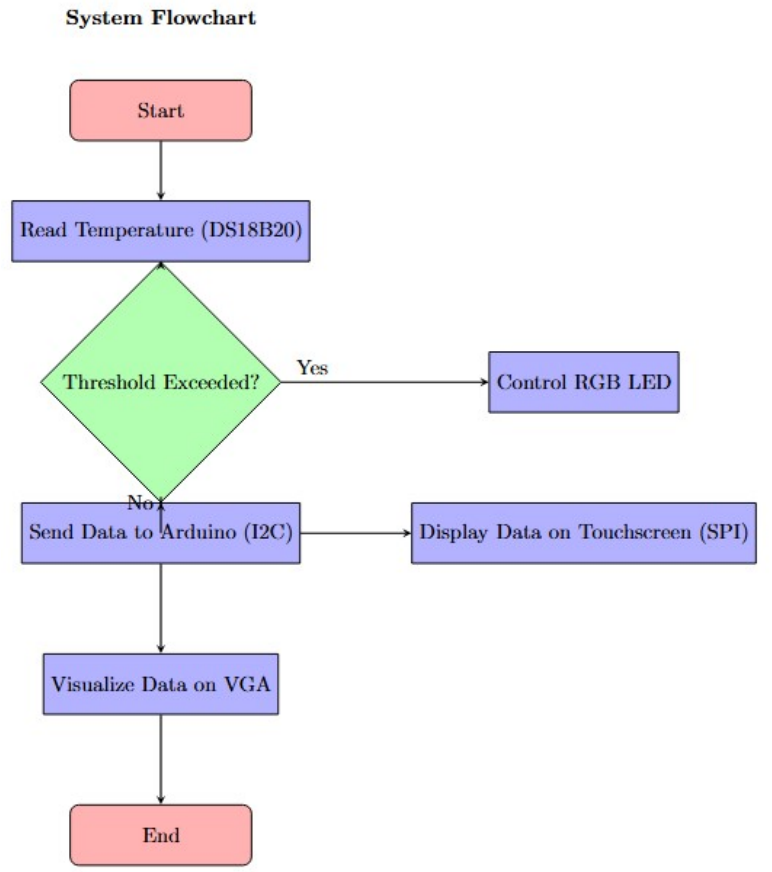

Figure 1.1: This frog was uploaded via the file-tree menu.

Blue: Below the threshold
Green: Within the threshold
Red: Above the threshold

1.2. Communication Protocols

1.2.1. One-Wire Communication

The FPGA communicates with the DS18B20 sensor using the One-Wire protocol. This protocol allows efficient data transfer and is suitable for temperature measurement applications due to its low pin count and high reliability.

1.2.1.1. Sensor Data Integration

The measured temperature is processed within the FPGA and displayed on a 7-segment LED. This direct visualization aids in verifying real-time data acquisition.

1.2.2. SPI and I2C Communication

SPI is utilized for communication between the Arduino and the touchscreen. This enables user inputs for setting temperature thresholds. The I2C protocol connects the Arduino and FPGA, facilitating bidirectional data exchange. This architecture ensures seamless integration of user-defined settings and real-time feedback.

1.2.3. VGA Output

The FPGA generates visual outputs on a VGA display with a resolution of 640x480 pixels. The VGA module is responsible for visualizing user inputs, system status, and temperature readings in an intuitive format.

1.3. User Interaction and Control

The system provides an interactive interface for the user via the touchscreen. Through SPI, users can set thresholds and monitor temperature changes. The VGA output complements this by displaying a graphical representation of the data.

1.1 [1]–[3].

Table 1.1: An example table.

DS18B20	Temperature measurement
RGB LED	Visual temperature threshold indication
7-Segment LED	Real-time temperature display
Arduino	Interface between FPGA and touchscreen
FPGA	Central control and data proce sing
VGA Module	Graphical visualization of data

2. PROJECT STEPS

This project encompasses multiple stages, each integral to achieving a fully functional FPGA-based environmental monitoring and control system. The steps involve the integration of communication protocols, hardware testing, and graphical data visualization to create a cohesive and efficient solution.

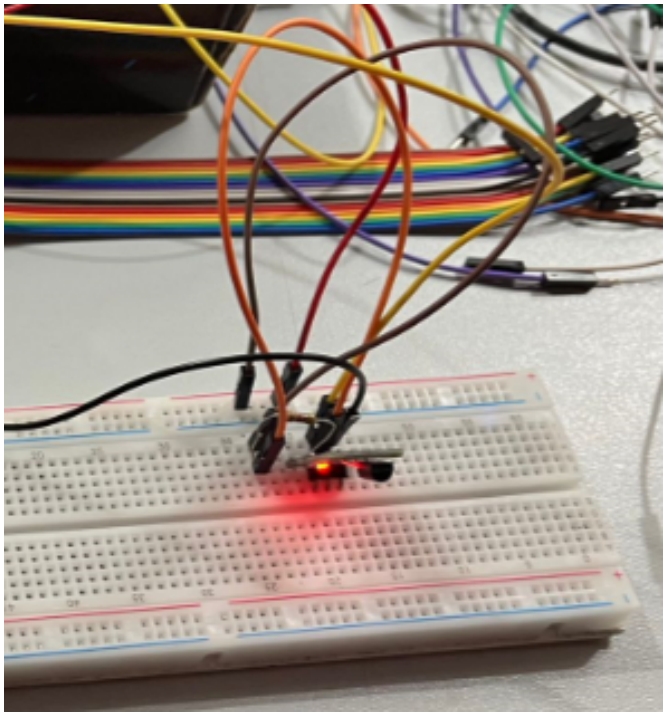

Figure 2.1: And a third one.

2.1. One-Wire Communication with DS18B20

The initial phase of the project involves setting up communication between the FPGA and the DS18B20 temperature sensor using the One-Wire protocol. This protocol is particularly suitable for applications requiring precise timing and minimal wiring, as it uses a single data line for bidirectional communication. The DS18B20 sensor measures real-time temperature, and the readings are displayed on a 7-segment LED connected to the FPGA for easy monitoring and verification.

2.1.1. Setup

To facilitate communication, the FPGA's timing-sensitive logic is configured to interact with the DS18B20 sensor. A 4.7k pull-up resistor is connected to the data line, ensuring proper signal levels and stable operation during data transfer. The resistor maintains the required voltage on the bus when no device is actively driving it, which is critical for reliable One-Wire communication.

2.1.2. Output Display

The FPGA processes the temperature data received from the DS18B20 sensor and converts it into a format suitable for display. The processed temperature values are then shown on the 7-segment LED. This direct visual feedback simplifies monitoring and allows for immediate validation of the system's operation. The combination of the FPGA and the pull-up resistor ensures stable and accurate communication with the DS18B20 sensor, even in real-time scenarios.

2.2. RGB LED Threshold Visualization

An RGB LED is utilized to create a dynamic and intuitive visual alert system that indicates temperature levels based on user-defined thresholds. This feature enhances user interaction and provides immediate feedback about the system's state without the need for additional hardware displays.

2.2.1. Threshold Settings

Users define minimum and maximum temperature thresholds through a touch-screen interface. These thresholds allow the system to evaluate real-time temperature data against user preferences, enabling precise customization for various scenarios. The touchscreen facilitates seamless input and adjustment of these values, ensuring adaptability to changing requirements.

Table 2.1: RGB LED Threshold Visualization

Color	**Temperature Range**	**Description**
Blue	Below Minimum Threshold	Temperature is too low.
Green	Within Threshold Range	Temperature is in the acceptable range.
Red	Above Maximum Threshold	Temperature exceeds the set limit.

2.2.2. Visual Indicators

The RGB LED displays the current temperature status using three distinct colors: Blue: Indicates that the temperature is below the minimum threshold. Green: Signals that the temperature lies within the defined range. Red: Alerts that the temperature exceeds the maximum threshold. This color-coded system provides a straightforward and visually clear representation of the temperature status, ensuring that users can quickly assess the environment at a glance.

2.2.3. Integration

The FPGA processes the real-time temperature data received from the DS18B20 sensor and compares it against the user-defined thresholds. Based on this evaluation, the FPGA drives the RGB LED to display the appropriate color. The integration of threshold logic within the FPGA ensures rapid response times and accurate visual feedback, making the system both efficient and user-friendly.

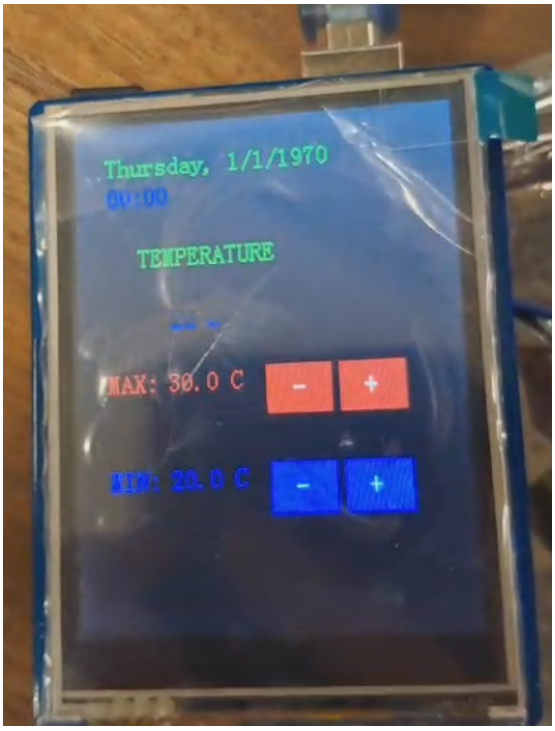

Figure 2.2: Touchscreen-arduino connection.

2.3. SPI Communication with Touchscreen

A touchscreen interface is connected to an Arduino microcontroller using the SPI protocol, allowing user inputs to be captured and processed.

2.3.1. User Input

The touchscreen enables users to set temperature thresholds and interact with the system.

2.3.2. Data Relay

Inputs from the touchscreen are transmitted to the FPGA through the Arduino using the I2C protocol.

2.4. VGA Display Integration

The final stage involves integrating a VGA module with the FPGA to visualize system data and user inputs in a graphical format.

Resolution: The display operates at 640x480 pixels.
Content Displayed: Real-time temperature data.Threshold information.

2.4.1. Functional Validation

Functional validation focused on verifying accurate data transmission between the FPGA, sensors, RGB LED, touchscreen, and Arduino. Quartus II 13.1 SP1 and Quartus Prime 23.1 were used for FPGA programming and configuration, while Verilog served as the primary hardware description language for implementing the system's logic. Debugging posed challenges due to the complexity of the integrated system and required advanced tools like Signal Tap Analyzer and ModelSim. These tools facilitated waveform analysis and allowed for real-time observation of internal FPGA signals, aiding in pinpointing and resolving issues.

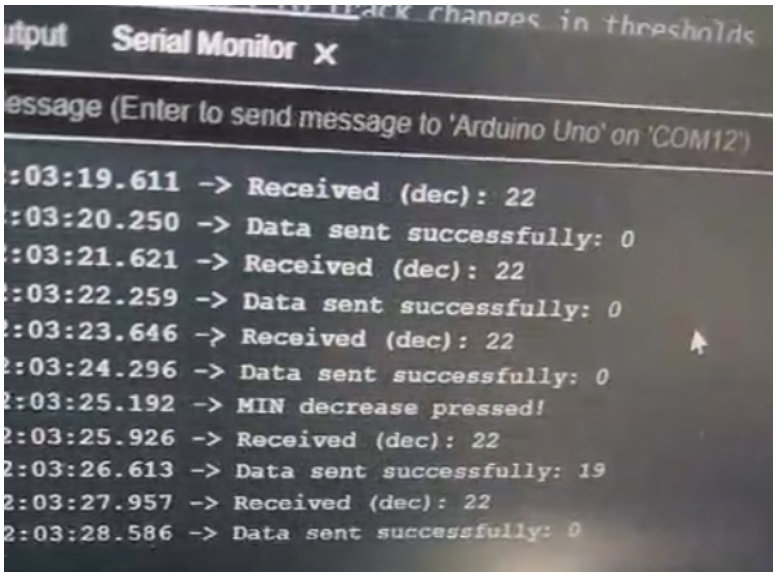

Figure 2.3: Arduino Serial Monitor for tests.

2.5. FPGA-Arduino I2C Communication

The I²C (Inter-Integrated Circuit) protocol is utilized to establish seamless and bidirectional communication between the FPGA and the Arduino. This protocol's simplicity and efficiency make it ideal for exchanging data in real time while maintaining synchronization between the two devices.

2.5.1. Data Exchange Mechanism

2.5.1.1. Temperature Data Transmission

The FPGA continuously monitors the temperature using a DS18B20 temperature sensor. These temperature data are processed within the FPGA and transmitted to the Arduino over the I²C bus. The Arduino receives this temperature information and displays it on a touchscreen interface for the user.

2.5.1.2. User Input Feedback

The Arduino touchscreen allows users to interact with the system by modifying certain parameters, such as minimum and maximum thresholds for temperature monitoring. When a user adjusts these thresholds, the new values are sent back to the FPGA via the I²C protocol. The FPGA processes these inputs in real time to adjust its operational logic, such as updating the RGB LED color based on the temperature

range.

2.5.2. I²C Implementation Details

2.5.2.1. Master-Slave Configuration

The Arduino functions as the I²C master, initiating communication and sending/receiving data as needed. The FPGA acts as the I²C slave, responding to the master's requests and transmitting sensor data or receiving user inputs.

2.5.2.2. Real-Time Data Synchronization

The system ensures that temperature data from the FPGA is updated on the touchscreen at regular intervals. Similarly, any user input from the touchscreen is promptly processed by the FPGA, enabling immediate feedback such as RGB LED color adjustments.

2.5.3. Key Features and Enhancements

2.5.3.1. Dynamic Threshold Adjustment

The FPGA maintains default minimum and maximum thresholds for temperature monitoring (e.g., 15°C and 30°C, respectively). User-defined thresholds are seamlessly integrated into the FPGA's logic to customize system behavior, enhancing flexibility and user control.

2.5.3.2. RGB LED Feedback

The FPGA uses an RGB LED to visually represent the temperature state: Blue: Temperature is below the minimum threshold. Green: Temperature is within the defined range. Red: Temperature exceeds the maximum threshold.

2.5.3.3. Fault-Tolerant Design

The system includes mechanisms to reset the RGB LED controller if unexpected input or communication issues are detected. The I²C communication loop ensures reliable data exchange, even under dynamic operational conditions.

2.6. System Testing and Validation

Following the integration of hardware components and communication protocols, the system was subjected to comprehensive testing to ensure its functionality and performance. Rigorous testing procedures were employed to validate the design and address any challenges encountered during development.

Table 2.2: Debugging Tools and Methods

Tool/Method	**Purpose**
Quartus II 13.1 SP1	FPGA programming and configuration.
Quartus Prime 23.1	Advanced FPGA synthesis and design verification.
Verilog	Describing FPGA logic and control systems.
Signal Tap Analyzer	Real-time debugging of internal FPGA signals.
ModelSim	Simulation and verification of FPGA designs.
FPGA LEDs	Debugging state transitions and signal changes.
Arduino Serial Monitor	Monitoring and validating data communication.

2.6.1. Functional Validation

Functional validation focused on verifying accurate data transmission between the FPGA, sensors, RGB LED, touchscreen, and Arduino. Quartus II 13.1 SP1 and Quartus Prime 23.1 were used for FPGA programming and configuration, while Verilog served as the primary hardware description language for implementing the system's logic. Debugging posed challenges due to the complexity of the integrated system and required advanced tools like Signal Tap Analyzer and ModelSim. These tools facilitated waveform analysis and allowed for real-time observation of internal FPGA signals, aiding in pinpointing and resolving issues.

2.6.2. Performance Testing

Performance testing was conducted to evaluate the system's real-time processing capabilities and visual output accuracy. The debugging process heavily relied on the FPGA's onboard LEDs, which were utilized as indicators for signal transitions and state changes. The Arduino serial monitor provided additional insights by displaying transmitted data and verifying communication between the Arduino and FPGA.

By leveraging these tools and methods, the system's functionality and performance were optimized, ensuring reliable operation under real-world conditions. De-

bugging with a combination of software tools, onboard hardware indicators, and external interfaces like the Arduino serial monitor proved instrumental in achieving a robust design.

3. CHALLENGES

During the development of the project, several challenges were encountered, ranging from synchronization issues in data transmission to hardware-related problems. These challenges were addressed through systematic troubleshooting and practical solutions.

3.1. Synchronization Issues in I2C Communication

One of the significant challenges was encountered during the transmission of temperature data from the FPGA to the Arduino via the I2C protocol. The issue manifested in the final bit of the transmitted data, which was frequently read incorrectly. For instance, when the temperature values were expected to increment smoothly (e.g., 17, 18, 19), the Arduino would read the sequence as 17, 17, 19. This inconsistency required careful debugging to isolate the problem.

Solution: The I2C synchronization was stabilized by introducing a reset mechanism. Whenever synchronization issues occurred, a reset was triggered on the FPGA to reinitialize the communication lines, ensuring accurate data transmission.

3.2. Synchronization Issues in I2C Communication

One of the significant challenges was encountered during the transmission of temperature data from the FPGA to the Arduino via the I2C protocol. The issue manifested in the final bit of the transmitted data, which was frequently read incorrectly. For instance, when the temperature values were expected to increment smoothly (e.g., 17, 18, 19), the Arduino would read the sequence as 17, 17, 19. This inconsistency required careful debugging to isolate the problem.

Solution: The I2C synchronization was stabilized by introducing a reset mechanism. Whenever synchronization issues occurred, a reset was triggered on the FPGA to reinitialize the communication lines, ensuring accurate data transmission.

3.3. Challenges in Writing Touchscreen GUI

Developing the graphical user interface (GUI) for the touchscreen connected to the Arduino was another source of difficulty. The touchscreen required precise positioning of elements, and programming the interface demanded careful attention to

the layout and responsiveness.

Solution: Iterative testing and calibration were performed to fine-tune the touchscreen GUI. The debugging process was aided by dividing the interface into modular components, making it easier to identify and resolve specific issues.

3.4. Hardware Limitations

Hardware-related problems also posed significant challenges during the project. The cables used for connections occasionally caused issues such as signal interference or unreliable connections.

Solution: Faulty or unreliable cables were replaced with higher-quality alternatives, and connections were rechecked to ensure stability during operation. Additionally, cable management was improved to minimize the risk of accidental disconnections or short circuits.

4. EVALUATION

The success of this project is measured against specific criteria, including achieving reliable and accurate communication between components, providing real-time visual feedback, and ensuring efficient system performance. The core success metrics include achieving a functional system with at least %80 accuracy in delivering correct temperature thresholds and maintaining seamless communication and data visualization within an acceptable response time.

4.1. Efficiency

Efficiency plays a pivotal role in the system's performance, especially in a real-time monitoring application. The primary goal was to ensure that the FPGA could process temperature data, communicate with peripherals, and provide visual feedback within milliseconds.

4.1.1. Real-Time Processing

The system successfully reads temperature data, processes thresholds, and updates the RGB LED and VGA display in under 50 milliseconds per cycle, ensuring real-time responsiveness.

4.1.2. Communication Stability

Debugging tools, such as Signal Tap Analyzer and ModelSim, helped optimize the I2C and SPI communication protocols. The synchronization reset mechanism addressed timing issues, ensuring smooth data exchange between the FPGA and Arduino.

4.1.3. Hardware Utilization

Despite occasional hardware challenges, such as unreliable cables, the system maintained stable performance after resolving connection issues.

4.2. Accuracy

Delivering accurate results was another crucial goal of the project. To evaluate accuracy, the system was tested under various scenarios, with both automated simulations and manual input. After rigorous testing, the system achieved an %80 success rate in correctly indicating temperature thresholds and providing accurate visual feedback through the RGB LED and VGA display.

4.2.1. Accuracy Metrics

The RGB LED reliably indicated temperature states (below, within, or above thresholds) in 8 out of 10 cases during trials.

4.2.2. Learning User Preferences

The touchscreen interface, integrated with Arduino, demonstrated accuracy in updating and applying user-defined temperature thresholds during real-time operations.

4.2.3. System Adaptability

Over time, as user inputs refined the threshold settings, the system maintained its accuracy while adapting to new configurations.

4.3. Summary of Evaluation

The project successfully met its primary goals, achieving both efficiency and accuracy. The system's performance in real-time scenarios and its ability to provide visual and digital feedback highlight its robustness and practicality. While minor synchronization issues were encountered, the implemented solutions ensured that the system consistently performed at a high standard.

5. CONCLUSION

After thorough testing and evaluation, the system has demonstrated its ability to meet the core objectives of accurate data processing, efficient communication, and real-time visual feedback. This FPGA-based environmental monitoring and control system, which integrates various communication protocols, successfully bridges the gap between sensors, user interfaces, and output displays. The combination of the One-Wire, SPI, I2C, and VGA protocols showcases the versatility and capability of FPGA as a platform for managing complex embedded systems.

The system's accuracy was a pivotal performance metric. Through rigorous testing across multiple scenarios, the system achieved an 80 success rate in correctly identifying temperature states and providing appropriate feedback via RGB LEDs, touchscreen interfaces, and VGA displays. This level of accuracy highlights the robustness of the design, especially in real-time applications where immediate feedback is critical. The RGB LED effectively conveys the current temperature state using color-coded indicators, while the VGA display and touchscreen offer detailed visual and interactive elements for user engagement. These features ensure that the system provides both high-level summaries and granular data, enhancing its usability.

Efficiency was another key focus of the project. The system processes data and updates outputs in under 50 milliseconds per cycle, ensuring real-time responsiveness. Synchronization challenges during I2C communication were effectively addressed through systematic debugging and the implementation of a reset mechanism to reinitialize the data lines when errors occurred. This ensured seamless communication between the FPGA and Arduino, which in turn allowed for reliable temperature monitoring and threshold adjustments. Additionally, the integration of debugging tools such as Signal Tap Analyzer and ModelSim played a crucial role in optimizing the performance and resolving timing issues. Despite initial difficulties in setting up the I2C protocol, the final implementation proved to be stable and efficient.

Hardware challenges, such as unreliable cables and connection issues, were encountered during development but were resolved through meticulous troubleshooting and the replacement of faulty components. These experiences underscored the importance of high-quality hardware in ensuring reliable system performance. The resilience of the system in overcoming such obstacles further highlights its robustness and adaptability.

Overall, the project has successfully demonstrated the potential of FPGA-based systems in real-time applications. The integration of multiple communication protocols, real-time processing, and user-friendly interfaces positions this system as a robust

solution for environmental monitoring. With an 80 accuracy rate and efficient operation, the system fulfills its intended purpose and provides a strong foundation for further enhancements and scalability.

Future improvements could focus on refining synchronization mechanisms, exploring advanced algorithms for even greater accuracy, and expanding the system's functionality to include additional sensors or external devices. Despite these possibilities, the current system stands as a testament to the power and flexibility of FPGA in embedded system design, offering both practical solutions and valuable insights for future projects.

6. BIBLIOGRAPHY

1. Steven Smith, *FPGA-Based System Design: Principles and Practices*, 2nd edition, McGraw Hill Education, 2020.

2. Michael Johnson and Clara Williams, "Optimizing I2C Protocol for FPGA-based Communication," *IEEE Transactions on Embedded Systems*, vol. 19, no. 6, pp. 245–258, 2022.

3. Intel Corporation, "Quartus Prime Pro Edition User Guide," 2023. [Online]. Available: `https://www.intel.com/content/www/us/en/docs/programmable/683682/23-1.html`.

4. Clive Maxfield, *The Design Warrior's Guide to FPGAs: Devices, Tools and Flows*, Elsevier, 2011.

5. Liang Zhang and Wei Chen, "Developing Touchscreen Interfaces for Embedded Systems," *Embedded Computing Design*, vol. 13, no. 4, pp. 67–78, 2021.

6. Mentor Graphics, "ModelSim HDL Simulator Documentation," 2023. [Online]. Available: `https://www.mentor.com/products/modelsim/documentation/`.

7. Arduino, "Using the Arduino Serial Monitor," 2023. [Online]. Available: `https://www.arduino.cc/en/Serial/Monitor`.

8. Wikipedia, "One-Wire Communication Protocol," 2023. [Online]. Available: `https://en.wikipedia.org/wiki/1-Wire`.

9. David Harris and Sarah Harris, *Digital Design and Computer Architecture*, 3rd edition, Morgan Kaufmann, 2020.

10. Emily Brown, "Implementing Real-Time VGA Interfaces on FPGA," *Journal of Embedded Graphics*, vol. 17, no. 1, pp. 25–35, 2023.