

REAL-TIME TEMPERATURE MONITORING USING FPGA-BASED MICROBLAZE PROCESSOR WITH TCP AND HTTP PROTOCOLS

Abeeb Opeyemi Nureni
Electronic Engineering Department
Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
abeeb-opeyemi.nureni@stud.hshl.de

Saikot Das Joy
Electronic Engineering Department
Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
saikot-das.joy@stud.hshl.de

Nirojan Navaratnarajah
Electronic Engineering Department
Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
nirojan.navaratnarajah@stud.hshl.de

Habeeb Rilwan Giwa
Electronic Engineering Department
Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
habeeb-rilwan.giwa@stud.hshl.de

Abstract—This project explores the design of digital systems utilizing a hardware description language and FPGAs (Field Programmable Gate Arrays), while also taking into account the restricted availability of financial resources and scientific advancements. In order to do this, the VHDL hardware description language was thoroughly investigated, and a brief user guide was created, containing the essential elements of the language to be utilized. The Xilinx Vivado software suite, a comprehensive circuit design and gateway implementation (FPGA) environment, was then put to the test. The design and simulation processes were thoroughly explained. The circuits were finished and tested using a Xilinx Artix-7 series FPGA on a Digilent Nexys A7 development board. The project was further developed utilizing the Pmod ESP32, the integrated temperature sensor, and Microblaze processor on this board as samples.

Index Terms—Microblaze, IPs, Synthesis, Block Automation, Block design, TCP, HTTP, FPGA, IDE, ISE

I. INTRODUCTION

Re-programmable devices have been created as a result of technological advancement, totally altering the way digital systems are designed. These devices are the FPGAs, which first debuted about 1980 and are now widely used for applications ranging from simple to complicated.

FPGAs are digital integrated circuits with programmable interfaces and digital logic modules. The logic functions that explain how a digital circuit works are implemented by programming the logic parts. Due to the versatility and speed they offer, FPGA applications are typically developed using hardware description languages like VHDL and/or Verilog. These applications span a number of disciplines, including biology, military systems, and cryptography.

This dissertation will examine one such tool since the ever-increasing complexity seen in the design and building of digital systems necessitates the usage of design tools that streamline this process. The project offers the reader the chance to learn more about the Xilinx VIVADO design environment, FPGA implementation, and the VHDL language. This project's goal was to integrate a Microblaze CPU into a Nexys A7 development board and transmit sensor data to a computer running a TCP/IP server. The TCP client is a Pmod ESP32 linked to the Local WiFi and to the Microblaze using the UART Protocol. The temperature sensor is connected to the Microblaze using the I2C protocol.

II. CONCEPT DESCRIPTION

As seen in figure 1, a microblaze processor is implemented in an FPGA board (Nexys A7) and used to read temperature data from a sensor. The microblaze processor is equipped with a TCP/IP stack, which allows it to send and receive data over a network. A TCP client is created on the microblaze processor to connect to a server listening on port 80, and the temperature data is transmitted to the server using this connection.

This system allows for remote monitoring and control of the temperature data, as the server can receive updates from the microblaze processor in real-time and potentially display or store the data for further analysis. The use of the microblaze processor and FPGA board enables the system to be highly customizable and scalable, as additional sensors or other peripherals can be easily added to the system.

III. PROJECT / TEAM MANAGEMENT

Project development has always been a plan-driven process, but with the advent of Agile Methodologies, project system development is now taking a more adaptable tack. The most critical demand for software systems today is frequently rapid development and delivery. Agile development methodologies, which we used in the creation of our project, first appeared in the late 1990s with the goal of drastically reducing the delivery time for systems.

The work on the program requirements, design, and implementation for our Very Large Scale Integration project was integrated. Every team member participated in the system specification and evaluation for the temperature monitoring utilizing the Microblaze CPU and ESP32, and automated testing was performed throughout the entire project's development.

Finally, the use of GitHub repository to keep the records of our works can not be underestimated.

IV. TECHNOLOGIES

Technologies are the tools, techniques, and processes that are used to create, produce, and use goods and services. These can include both physical technologies, such as machines and equipment, and intangible technologies, such as software and processes. Technologies are always changing and getting better, and they are a major force behind advancement and innovation across a wide range of industries. Each technology we used in this project is outlined in more detail below:

A. COMMUNICATION PROTOCOL

An information flow between devices or systems through a communication channel is governed by a set of rules and conventions called a communication protocol. Communication protocols can be used to enable communication between objects or systems that are physically close to one another or to enable communication between objects or systems over large distances, such as the internet. Communication protocols come in a wide variety and are each designed to address a particular set of requirements. The types of data that can be transferred, how it is formatted and encoded, the strategies employed to guarantee data transmission reliability and accuracy, and the systems for error detection and correction are some common characteristics of communication protocols. In embedded systems, communication protocols are frequently employed to enable interaction between various components and with outside devices. Using a communication protocol, for instance, a microcontroller may send data to a sensor or receive data from a display. Following is a list and description of the communication protocols we used in this project:

- Universal Asynchronous Receiver/Transmitter (UART)
- Inter-Integrated Circuit(I2C)
- Transmission Control protocol (TCP)
- Hyper Text Transfer Protocol (HTTP)

1) *Universal Asynchronous Receiver/Transmitter (UART):*

A form of communication protocol called UART (Universal Asynchronous Receiver/Transmitter) enables devices to send and receive data asynchronously, i.e., without the use of a fixed clock signal. UART is frequently used in embedded systems to enable communication between components and with outside devices like computers. In a UART communication, data is delivered and received as serial bits, with each bit denoting a single data element (such as a letter or a number). The timing and format of the data bits, as well as the methods for sending and receiving data, are all specified by the UART protocol. Two wires are normally needed for UART communication: one for sending data and one for receiving it. To regulate the data flow, some UART devices could also have extra control signals including flow control and interrupt signals. A popular and widely used protocol, UART is frequently combined with others to speed up device connection, including TCP/IP and USB. In our project, Temperature data is sent to esp32 module with UART protocol. UART protocol is also used to establish a connection between the computer and the Nexys A7-100T board.

2) *Inter-Integrated Circuit(I2C):* I2C (Inter-Integrated Circuit) is a communication protocol that allows devices to exchange data over a two-wire bus. It is frequently used in embedded systems to let devices communicate with one another and with external devices like computers. Each bit in an I2C transmission represents a single data element, and data is transmitted and received in the form of serial bits (such as a letter or a number). The I2C protocol defines the methods for transmitting and receiving data, as well as the time and format of the data bits. Two wires are needed for I2C communication: one for the clock signal (SCL) and one for the data signal (SDA). Each device on the I2C bus has a specific address, and when communicating, the devices send and receive data to and from these addresses. I2C is a popular and easy-to-use protocol that is frequently combined with others, like TCP/IP and USB, to help devices communicate with one another. The microblaze processor in our project uses the I2C communication protocol to receive data from the on-board temperature sensor of the Nexys A7-100T board.

3) *Transmission Control protocol (TCP):* TCP (Transmission Control Protocol) is a communication protocol that is used to transmit data over the internet and other networks. It is a transport layer protocol, which means that it resides in the transport layer of the Internet Protocol Suite (often referred to as the TCP/IP architecture). With the help of TCP, two systems or devices can connect in a dependable, stream-oriented manner. This is accomplished by establishing a connection between the two devices, segmenting the data that is being transmitted, and transferring the segmented data to the target device. The final step is for the destination device to recombine the segments back into the initial data stream after acknowledging receipt of them. TCP can request that any missing or damaged segments be sent again, ensuring that the data is carried accurately and entirely. TCP is frequently used

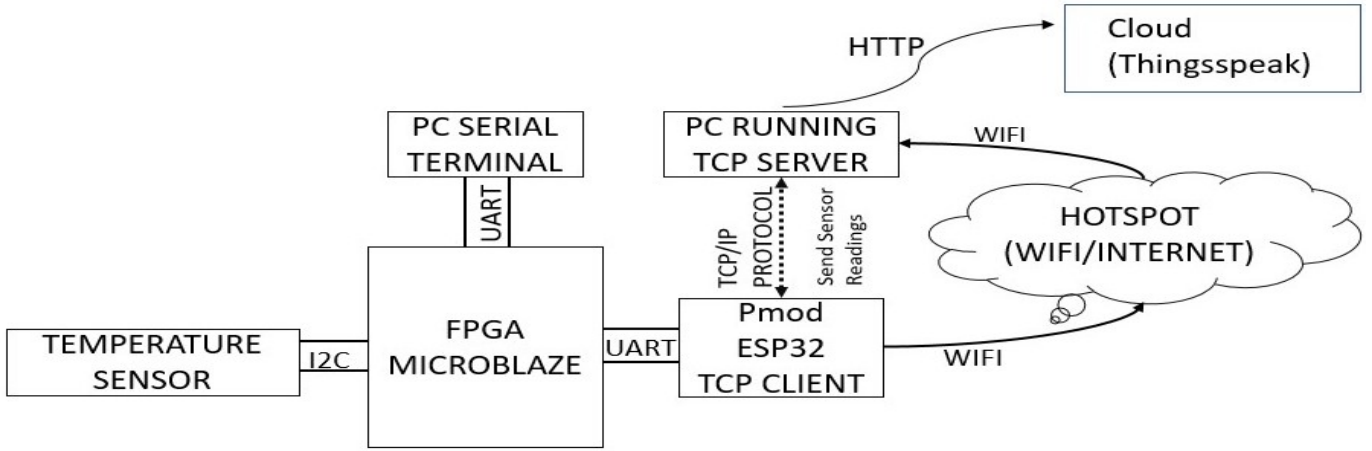


Fig. 1. Concept description

in systems that demand a dependable connection, including file transfers, email, and web browsing. It frequently works in tandem with Internet Protocol (IP), which is in charge of directing the data packets over the network. In this Project , We used this protocol to send the temperature to a TCP Server running on another PC(IoT).

4) *Hyper Text Transfer Protocol (HTTP)*: HTTP is a protocol for sending and receiving messages over the internet. It is the foundation of the World Wide Web and is used to transmit data between a server and a client. HTTP allows clients (e.g., web browsers) to send requests to servers (e.g., web servers) and servers to send responses back to clients. HTTP uses a simple, text-based format and is designed to be extensible, reliable, and efficient. It is widely used to transmit data such as HTML documents, images, and other types of media. In this project, the temperature data transferred from Microblaze to the ESP32 is sent over this protocol to the Thingspeak Cloud.

B. VIVADO DESIGN SUITE

Vivado Design Suite is a comprehensive and powerful software tool suite for designing and implementing digital logic circuits on FPGA (Field-Programmable Gate Array) devices. It was created by Xilinx, a top supplier of FPGA technology, and is widely utilized by experts in the digital design and embedded systems industries. To generate, implement, and debug digital designs for FPGAs, designers can use a variety of tools and functionalities in the Vivado Design Suite. These tools consist of a graphical design environment for constructing and configuring digital circuits, a synthesis tool for transforming the design into an FPGA-implementable format, and a place-and-route tool for placing the design on the unique resources of an FPGA device. The suite also includes a number of IP (Intellectual Property) building blocks that are simple to incorporate into a design, along with simulation and verification tools. The Vivado Design Suite also provides a variety of other features and tools to support the design process in addition to these basic tools, such as support for high-level design languages, reuse and version control for designs, and integration with

external tools and flows. We utilized Vivado Design Suite to create the hardware part of our project.

C. INTELLECTUAL PROPERTY (IP)

In Vivado, IP (Intellectual Property) refers to a pre-designed, pre-verified functional block that is simple to incorporate into an FPGA design. When implementing common functionalities like memory interfaces, processors, and communication protocols, IP blocks are frequently utilized. These blocks can be altered to match the unique requirements of a particular design. By allowing designers to reuse functional blocks that have already been tried and tested rather than having to create and validate these blocks from scratch, using IP blocks can accelerate the design process. A large range of pre-made IP blocks are available in Vivado, and designers can also buy or request the creation of unique IP blocks from outside vendors. Before employing IP blocks in Vivado, designers must first add them to their designs using the "Create and Package IP" flow or by dragging and dropping IP blocks from the IP catalog. After then, the designer can modify the IP block to suit the particular requirements of the design, such as choosing the proper interface standards or adjusting the memory amount. The IP block can then be integrated into the broader FPGA implementation by connecting to other blocks in the design. In our project, we modified and properly connected already-existing IPs to design the hardware component.

D. VITIS

Vitis is a platform developed by Xilinx for developing and deploying high-performance, software-defined systems on Xilinx devices. It is a comprehensive and efficient set of tools and libraries that can be used to develop, implement, and optimize systems for a variety of applications, including computer vision, machine learning (ML), artificial intelligence (AI), and more. Developers can build and deploy software-defined systems on Xilinx hardware with the aid of a variety of tools and resources available on the Vitis platform. These technologies include the Vitis Integrated Environment, a software

package for creating digital logic circuits on FPGA (Field-Programmable Gate Array) devices, and the Vitis AI libraries, a collection of streamlined libraries and tools for creating AI and ML applications. Additionally, the platform offers a variety of tools for high-level synthesis and hardware-software co-design to improve system performance. The Vitis platform also includes a number of extra features and resources to support the creation and deployment of software-defined systems, such as support for a variety of programming languages and frameworks, integration with third-party tools and libraries, and a variety of design examples and tutorials. These extra features and resources are in addition to the platform's core tools and resources. In our project, we used this platform to integrate software into the hardware that was created and to validate the design uploading to an FPGA board (Nexys A7-100T)

E. NEXYS A7-100T

The Nexys A7-100T is a FPGA (Field-Programmable Gate Array) development board produced by Digilent. It is based on the Xilinx Artix-7 FPGA, which is a mid-range FPGA device that is suitable for a wide range of applications. The Nexys A7-100T development board is designed to be a flexible and effective platform for exploring and analyzing embedded systems and digital design. It has a variety of features and add-ons to assist customers in getting started with FPGA development, including a microSD card slot for data storage, an HDMI interface for video output, and a number of expansion ports. The Vivado Design Suite from Xilinx, which may be used to design and construct digital circuits on the FPGA, is one of many software tools that are supported by the Nexys A7-100T development board. Additionally, a number of libraries and programming languages, including as C/C++, Python, and VHDL (VHSIC Hardware Description Language), support it, enabling users to develop original ideas and applications for the board. To upload the design and review it, we used this board for our project. Additionally, we used this board's on-board temperature sensor. A general image of this board is shown in Fig 2

F. ADT7420 TEMPERATURE SENSOR

Analog Devices makes the ADT7420, a low-power, high-accuracy temperature sensor IC (Integrated Circuit). It can be used to measure temperatures in a variety of applications, and it can be readily interfaced with the Nexys A7-100T FPGA development board utilizing either the I2C or SPI communication interfaces. The ADT7420 must be connected to the proper communication interface on the Nexys A7-100T board in order to be used with the board. Next, a digital design must be created that reads the temperature data from the sensor and processes it as necessary. The ADT7420 has a wide temperature range, from -40°C to $+125^{\circ}\text{C}$, making it suited for usage in a variety of situations. It also has a resolution of 16 bits, which enables it to detect temperature with a high degree of accuracy. Due to its low power consumption, the

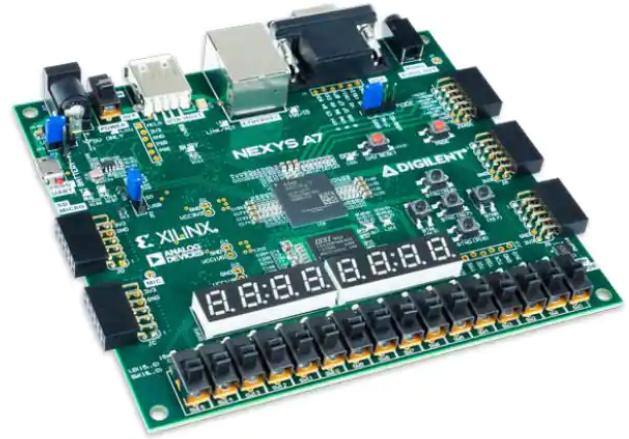


Fig. 2. NEXYS A7-100T FPGA Board

ADT7420 can be used in battery-powered or energy-sensitive applications. We utilized the Nexys A7-100T board's integrated ADT7420 temperature sensor for our project.

G. Pmod ESP32

The Digilent Pmod ESP32 is a Peripheral Module (Pmod) that enables quickly integrate Wi-Fi connection into the designs. It is based on the ESP32 microcontroller, a high-performance, low-power microcontroller with built-in Bluetooth and Wi-Fi. The 6-pin Pmod connector on the Pmod ESP32 can be used to connect it to a host board, and it is designed to be readily integrated into a variety of designs. In order to support Wi-Fi networking, it has a variety of features and add-ons, including an antenna, a power amplifier, and a low-noise amplifier. The Pmod ESP32 also works with a variety of libraries and programming languages, including as Arduino, Python, and C/C++, enabling users to quickly develop unique applications for their creations. The Pmod ESP32 can be used with a variety of host boards, including the Nexys A7-100T FPGA development board. The Pmod ESP32 should be connected to the relevant Pmod connectors on the Nexys A7-100T board before being used with the board. The ESP32 microcontroller will then be used to implement Wi-Fi connection in a digital design. The design can then be implemented on the FPGA and tested using the Vivado Design Suite's tools, and the Pmod ESP32's Wi-Fi capabilities can then be utilised. In our project, we attached this module to the Nexys A7-100T board's peripheral. For our design, we also used Pmod ESP32 IP. In Fig 3, a general picture of Pmodesp32 is shown.

- AT commands are used to control a modem, and are generally used to set up and configure a wireless modem. Here are a few common AT commands used with ESP32 PMOD:
 - AT+CWJAP=<ssid>,<pwd>: Connect to a WiFi network
 - AT+CIPSTART=<type>, <addr>,<port>: Start a connection over TCP or UDP



Fig. 3. Pmod ESP32

- AT+CIPSEND=<length>: Send data over the connection

H. MICROBLAZE

Microblaze is a soft processor core that can be implemented in an FPGA (Field-Programmable Gate Array) and configured to meet the specific needs of a particular design. It was created by Xilinx and is frequently used in embedded systems to build specialized microcontroller-based systems and manage the different peripherals and devices attached to the FPGA. Microblaze is a RISC (Reduced Instruction Set Computing) processor designed specifically for FPGAs. Its small size, light weight, and low resource utilization make it an excellent choice for designs with limited resources. To satisfy the unique requirements of a particular design, Microblaze can be built to support a variety of instruction set architectures (ISAs) and be customized with a variety of peripherals and memory interfaces. The Microblaze processor and any other required peripherals and devices must be included in the digital design in order to utilize Microblaze in a design. The design can then be implemented on an FPGA using the Vivado Design Suite tools or other digital design tools, and the performance of the Microblaze processor and the customized embedded system can then be tested. Internet of things (IoT) systems, industrial automation, embedded control, and other applications all frequently use Microblaze. In figure 4, a core block diagram of Microblaze Processor is given.

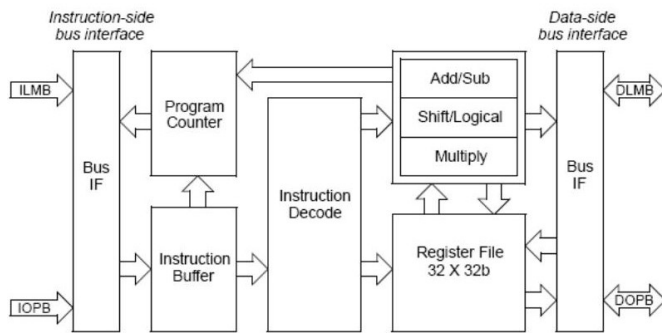


Fig. 4. Microblaze Core Block Diagram

V. HARDWARE IMPLEMENTATION

The hardware, being the bedrock of any design, we have employed in this project the Nexys A7-100T FPGA board as the hardware device for our design, replacing an external temperature sensor, a UART AXI IP block, and a PmodEsp32 IP block in our design with the temperature sensor on the board. Each of these served as the foundation for the design's development. To better understand the process on how we have put together the hardware for the design, the subsequent sections give an insight to this.

1) *Hardware Design Process:* To create the hardware design for our project, we first created a new RTL project in Vivado and selected the Nexys A7-100T board as the target hardware platform. We then added the Microblaze processor to our design using the IP catalog in Vivado. We also added a number of other IP blocks to our design, including an AXI UARTLite IP block and a PmodEsp32 IP block. The PmodEsp32 IP block was not available in the default IP directory, so we downloaded it from the internet and added the IP directory path to the Vivado settings.

2) *Configuration of IP Blocks:* We made a number of changes to the default settings of the Microblaze processor and other IP blocks in our design. For example, we ran the block automation for the Microblaze processor and selected a local memory size of 128Kb and enabled the interrupt controller. We also double-clicked on the Clocking Wizard IP block and edited the settings in the board tab. We selected the interface of Clk_IN1 as system and the interface of Ext_reset_In as reset. On the output tab, we selected a reset type of active low. We also double-clicked on the Concat IP block and selected the number of ports to be four (4).

3) *Configuration of IP Blocks:* In addition to the Microblaze processor and IP blocks, we also added a temperature sensor and an accelerometer to our hardware design. We dragged and dropped these components onto the design from the board tab in Vivado. We also added a board component, a **jb** connector, to the Pmod out on the design.

We connected the various hardware components together using the interconnects and buses available in Vivado. We also ran the connection automation to ensure that everything was properly connected. Figure 5 displays the project's complete IP block diagram.

4) *Validation, Synthesis, and Implementation:* After connecting all of the hardware components in our design, we validated the design to ensure that it was correct. We then synthesized and implemented the design, resolving any issues that arose during this process. Our IP block Design is successfully validated, as shown in figure 6

5) *Generation of Bitstream and Export of Design:* Finally, we generated the bitstream for our design and exported it as an .xsa file, including the bitstream. This allowed us to program the Nexys A7-100T board with our hardware design.

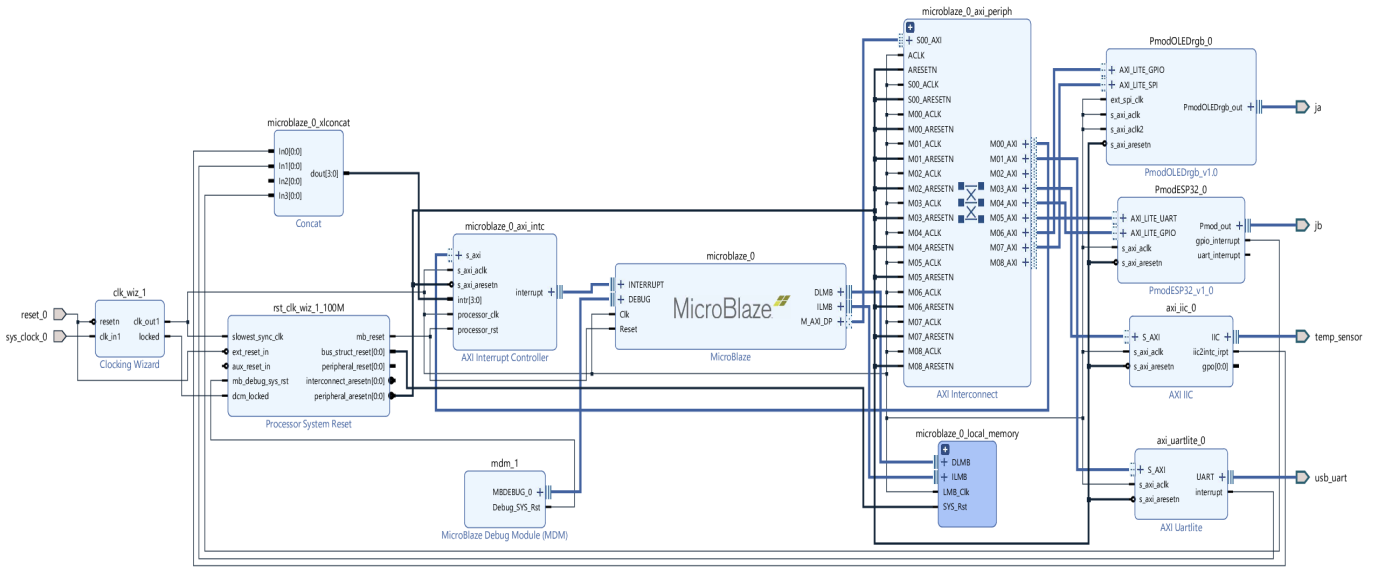


Fig. 5. IP block diagram

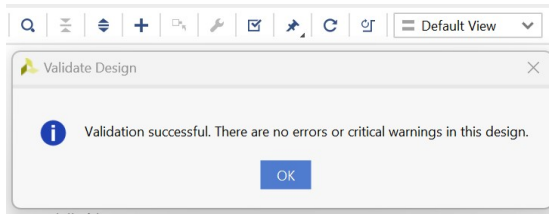


Fig. 6. Design Validation

The successful generation of the bitstream for our system is shown in figure 7.

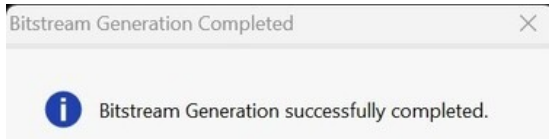


Fig. 7. Bitstream generation window

VI. SOFTWARE IMPLEMENTATION

1) *HyperTerminal / Tera Term*: HyperTerminal is a terminal emulation program that can be used to communicate with other devices through a serial port or a Telnet connection. It is often used in Internet of Things (IoT) projects to monitor and control devices remotely.

Here are some examples of how HyperTerminal could be used in an IoT project:

- **Debugging**: HyperTerminal can be used to send commands to an IoT device and display the responses in a text window. This can be useful for troubleshooting issues with the device or testing out new features. Mini

- **Monitoring**: HyperTerminal can be used to display real-time data from an IoT device, such as sensor readings or status updates. This can be useful for monitoring the performance of the device and ensuring that it is functioning properly.
- **Control**: HyperTerminal can be used to send commands to an IoT device to perform certain actions, such as turning on a light or adjusting the temperature. This can be useful for remotely controlling the device from a computer or other device.

Overall, HyperTerminal plays an important role in many IoT projects by providing a way to communicate with and control devices remotely.

A. ThingSpeak

ThingSpeak is a cloud platform that allows users to collect, store, and visualize data from devices and sensors in the Internet of Things (IoT). It provides a simple way for users to create IoT applications by providing tools for data analysis, visualization, and integration with other applications and services. ThingSpeak offers both a free and a paid version, and it is used by a wide range of users, including researchers, students, makers, and businesses.

ThingSpeak allows users to create channels to store and visualize data, and provides APIs for sending data to and receiving data from channels. It also offers a range of integration options, including integration with other cloud platforms and services such as AWS, Azure, and Google Cloud, as well as integration with various programming languages and platforms such as Python, Arduino, and Raspberry Pi.

B. Sending Data to the cloud (ThingSpeak)

The Python code running on Pycharm creates a TCP/IP socket and listens for incoming connections on port 80 (in this case).

When a connection is established, the code receives temperature data from the ESP32 in small chunks and sends the data to Field 2 of a ThingSpeak channel using the ThingSpeak API. A brief overview of how the data sending process happens in the python code:

- The code creates a TCP/IP socket and binds it to port 80. The code listens for incoming connections and accepts a connection when one is received.
- The code sends the received data to Field 2 of a ThingSpeak channel using an HTTP POST request to the ThingSpeak API.
- The ThingSpeak API receives the request and processes it. The API stores the data in Field 2 of the specified channel.
- The ThingSpeak API responds to the request with a success or error message.
- The code checks the status code of the response and prints a message to the console if the request was successful and continues to receive data from the client in a loop until there is no more data to be received.

```
Status = TempSensorExample(IIC_DEVICE_ID, TEMP_SENSOR_ADDRESS,
                           &TemperaturePtr);
xil_printf("Temp :%d °C \r\n",TemperaturePtr);
```

Fig. 8. Temperature code

VII. EVALUATION AND RESULTS

Here are the specifics of the processes taken in the evaluation in order to evaluate this project's work and determine whether the system we designed is capable of meeting the requirements:

- Utilizing the "validate the design" tools in the Vivado design suite after the design block was finished successfully demonstrates that the design is error-free. The hardware was then exported to Vitis IDE after that.
 - The C code is written here in Vitis IDE, and we were able to identify any issues in the code by using the debugging tools to check it.
 - The Vitis terminal wasn't adaptable enough to handle some external commands, such "AT COMMAND" from Espressif, which will be required to communicate information from the PC to the ESP32, which is why we chose a TeraTerm terminal as seen in figure 9.
- Tera term has its own restrictions, and using this terminal to send data to the cloud was not an option. This obstacle was subsequently overcome by writing a Python script as seen in figure 10 that is adaptable enough to send various commands from the ESP32 to the cloud, and it was ultimately successful.
- Finally, the sent data from the temperature sensor of the Nexys7 FPGA board through the ESP32 to the cloud using a ThingSpeak IoT platform was successfully implemented and the results were graphically displayed as seen in figure 11

Below are snippets of the data interpretation in the GUI as seen in figure 9, figure 10, and figure 11

```
COM6 - Tera Term VT
File Edit Setup Control Window Help
Temp :14 °C
Successfully ran IIC tempsensor on board
Requesting if ESP32 is ready to send Data
Connecting to TCP Server Rostock
VIFI CONNECTED
CONNECT
AT+CIPSEND=8
OK
>Temp :14 °C
busy p...
Recv 8 by
DAT+CIPSEND=8
OK
>Temp :14 °C
busy p...
Recv 8 by
SNAT+CIPSEND=8
OK
>Temp :14 °C
busy p...
```

Fig. 9. Results obtained from the external terminal "TeraTerm"

```
Run: C:\Users\miniro\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:\Users\miniro\PycharmProjects\pythonProject2\main.py
starting up on ('', 80)
waiting for a connection
connection from ('192.168.173.127', 50561)
VLSI Project Team Rostock
Message from esp32 connected to microBlaze
Received data: 13 °C
send to cloud ok
Received data: 13 °C
send to cloud ok
Received data: 13 °C
send to cloud ok
Received data: 13 °C
send to cloud ok
Received data: 13 °C
send to cloud ok
Received data: 13 °C
send to cloud ok
```

Fig. 10. Results obtained from the python script to establish the connection

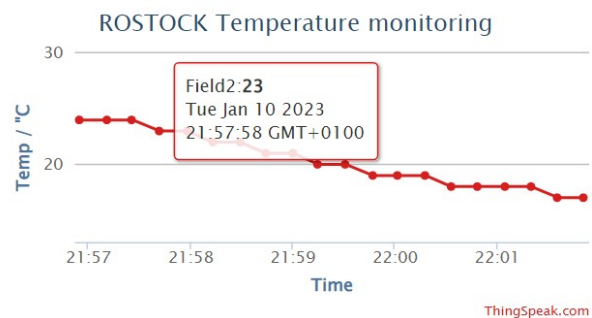


Fig. 11. ThingSpeak graph

VIII. CONCLUSION

As was already mentioned, the purpose of this project is to use the Microblaze CPU and ESP32 to communicate real-time

temperature data to the cloud. Writing C code on the Vitis IDE to operate the designed system followed the design and use of customized IPs on an integrated synthesis environment (ISE). To establish the communications between the system's components (Microblaze, ESP32), an external terminal was used. TCP was crucial in ensuring that the data being received from the temperature sensor of the FPGA was not lost and that the intended system could be realized.

As a result of our ability to program the processor to suit our design, FPGA has shown to be a good hardware option that offers a great degree of design freedom. On a ThingSpeak IoT platform, the anticipated temperature data was visualized.

Future developments of the microblaze will be able to connect a variety of sensors, much like Arduino, and allow for programming without the restrictions present in today's ordinary microprocessors.

IX. AFFIDAVIT

AFFIDAVIT - ABEEB OPEYEMI NURENI

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.



Abee Nureni
Lippstadt, 13.01.2023

AFFIDAVIT - NIROJAN NAVARATNARAJAH

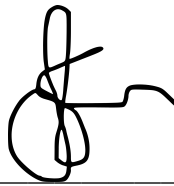
I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.



Nirojan Navaratnarajah
Lippstadt, 13.01.2023

AFFIDAVIT - HABEEB RILWAN GIWA

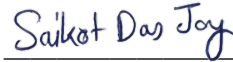
I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.



Habeeb Rilwan Giwa
Lippstadt, 13.01.2023

AFFIDAVIT - SAIKOT DAS JOY

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.



Saikot Das Joy
Lippstadt, 13.01.2023

REFERENCES

- [1] K. -c. Leung, V. O. k. Li and D. Yang, "An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges," in IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 4, pp. 522-535, April 2007, doi: 10.1109/TPDS.2007.1011.
- [2] U. Nanda2 and S. K. Pattnaik, "Universal Asynchronous Receiver and Transmitter (UART)," 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), 2016, pp. 1-5, doi: 10.1109/ICACCS.2016.7586376.
- [3] D. Levshun, A. Chechulin and I. Kotenko, "A technique for design of secure data transfer environment: Application for I2C protocol," 2018 IEEE Industrial Cyber-Physical Systems (ICPS), 2018, pp. 789-794, doi: 10.1109/ICPHYS.2018.8390807.
- [4] <https://www.xilinx.com/support/documentation/navigation/design-hubs/dh0003-vivado-designing-with-ip-hub.html>
- [5] <https://ebics.net/xilinx-vivado/>
- [6] <https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual>
- [7] <https://digilent.com/reference/pmod/pmodesp32/start>
- [8] <https://www.xilinx.com/products/design-tools/microblaze.html>
- [9] <https://reference.digilentinc.com/programmable-logic/nexys-a7/reference-manual>
- [10] <https://www.xilinx.com/products/design-tools/vivado.html>
- [11] <https://www.seas.upenn.edu/~ese171/vhdl/vhdlprimer.htm>