FPGA Based Smart Home Automation System

1st Ravindu Athukorala
dept. Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
ravindu.athukorala@stud.hshl.de

2nd Kajeepan Umaibalan
Department of Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
kajeepan.umaibalan@stud.hshl.de

3rd Lochana Abhayawardana
Department of Electronic Engineering)
Hochschule Hamm-Lippstadt
Lippstadt, Germany
lochana.abhayawardana@stud.hshl.de

***Abstract*—While the human race is evolving, they are tending to find an easier way to make their life easier. Modern days due to fast technical developments the automation of day-to-day activities was capable. During this project, we are targeting to automate and monitor attributes related to smart homes such as temperature, lightand detection of the movements inside the house. For such projects, microcontroller is the popular approach. Apart from using a microcontroller for the project, we used an FPGA because of its real-time processing capability, reconfigurability, and hardware reconfigurability. In this report, we will explain the approach we took, how found solutions to the challenges, and the future direction of this project.**

I. Introduction and Concept Description

Our plan for this project is to access the FPGA's internal temperature sensor to measure the room temperature and light intensity, detect the movements inside the house, and send the details via the Pmod Bluetooth module to the mobile phone in order provide security and monitor the surroundings.

II. Technologies

A. Hardwares

To achieve optimal performance, we did some research and found out that the following hardware items would be most suitable for our project.

1) FPGA: For the FPGA we used Nexys A7 100T which is considered a trainer board and has midrange performance and price. Furthermore, it is compatible with the tools we are using in the project like Xilinx Vivado and Xilinx Vitis. It has 15,850 Programmable logic slices, 4,860 Kbits of fast block RAM and the internal clock speed is 450 MHz. The Nexys A7 100T FPGA is equipped with 128MiB DDR2 serial flash memory. [1] For this project, we are using the onboard Temperature sensor and four Pmod connectors to connect additional sensors and modules.

2) Temparature Sensor: Nexys A7 FPGA has an inbuilt analogue temperature sensor ADT7420. This sensor has a maximum resolution of 16 bits and is capable of displaying temperature with an accuracy of 0.25 degrees Celsius. [1]

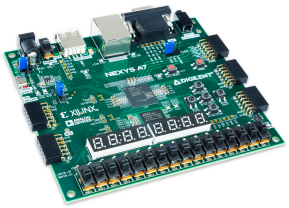

Fig. 1. Nexys A7 100T [1]

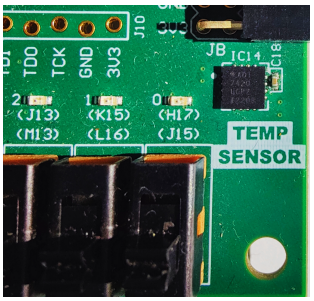

Fig. 2. The inbuild temperature sensor in FPGA [1]

3) Ultrasonic Range Finder: For the ultrasonic range finder we have used Pmod MAXSONAR by Digilent company. It has an effective detection range from 6" to 255". The sensor can detect an object, size of one inch, up to 20 feet away. Furthermore, it has free-run (continuous measurement) capability. The users have 3 methods of sending data. Those are UART, analogue and PWM. [1]

Fig. 3. ultrasonic sensor [1]

4) Ambient Light Sensor: For this project, we have used PmodALS as the ambient light sensor. This is also manufactured by Digilent. This sensor is capable of converting light data to digital data with 8-bit resolution. It has 6 pins including CS,(NC), MISO, SCK, GND and Vcc by using the SPI interface. Furthermore, this sensor follows Digilent Pmod Interface specification Type 2. [1]

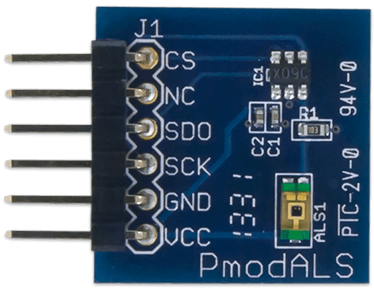

Fig. 4. Ambient light sensor [1]

5) Blue Low Energy module: In our project, the Bluetooth module acts as the communication device between the laptop and the mobile phone. We have used Digilent Pmod BLE which has 12 pins Pmod connector with UART interface. This sensor uses using Bluetooth 4.2 low-energy module which can be remotely configured over the air. Data is encrypted with AES128. Furthermore, it has ASCII command interface API and it is certified with FCC, IC, CE, KCC, NCC and SRRC. [1]

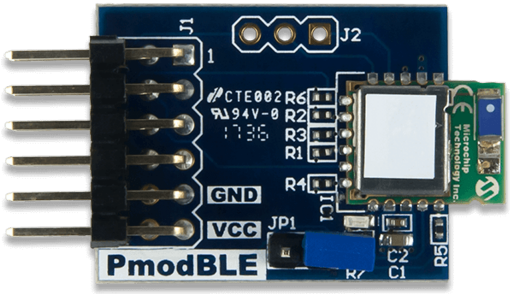

Fig. 5. Bluetooth module [1]

B. Softwares

Our project is build on the Nexys-A7-100t FPGA as the primary hardware platform for our smart home application. For hardware programming and project implementation, Xilinx devices use major software platforms.

C. Vivado

The main tool for hardware design and synthesis is Vivado, which makes it easier to create an FPGA design that includes all of the different parts of the smart home system. As the primary goal of this project, Vivado assists us in integrating Microblaze into Nexys-A7-100t digilent board. Its modular and customisable architecture enables the integration of various IP cores, including communication modules, sensor interfaces, and logic blocks specifically designed for smart home applications. Vivado assists in several ways during the project in the FPGA development process such as: [2]

- Hardware Description Language (HDL) Design
- Design Entry
- Synthesis
- Implementation
- IP Integration
- Debugging and Verification

Vivado helps our project to integrate MicroBlaze and allows us to connect Pmodsensors IP blocks in our project and also helps debugging to check implementation and synthesis. Fig 6 shows the user interface of Vivado 2021.2 version which was used during the project

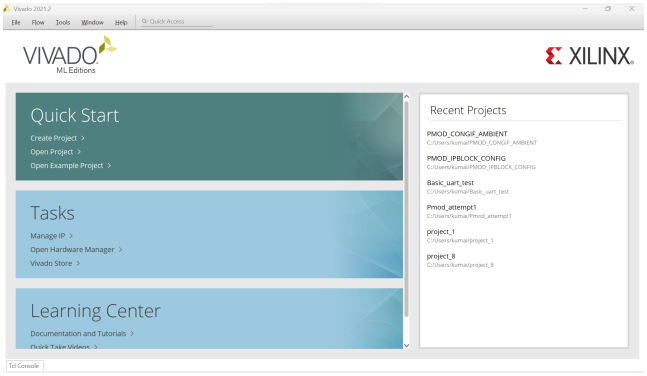

Fig. 6. Vivado

D. Vitis IDE

The integration of Vitis IDE becomes essential to achieving the goal of an intelligent living environment in our Smart Home Applications project. Using MicroBlaze as our Xilinx FPGA's embedded processors. Key uses of this platform during our projects are:

- Unified Development Environment: In our Smart Home project, the Vitis IDE acts as the main development environment for embedded software applications.
- Programming Language Support: Multiple high-level programming languages, including C and C++, in which

we used c programming to align with the programming requirements of MicroBlaze.

- Debugging and Performance Analysis: Vitis IDE includes powerful debugging tools and performance analysis capabilities, which help the project to ensure reliability and efficiency.

Figure 7 shows the user interface of Vitis IDE 2021.2 version.

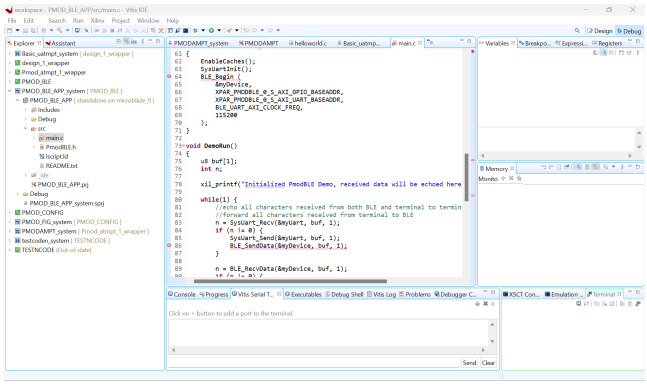

Fig. 7. Vitis

III. Implementation

A. Hardware

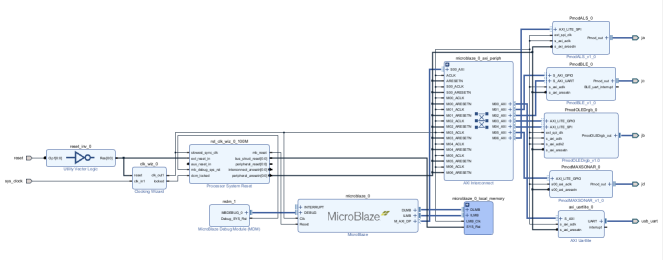

Fig. 8. IP Block Design

The first step of the implementation of the project would be creating the IP block. We use the Vivado platform for that. First, we added a MicroBlaze. Then run block automation. local memory 128kb. Recoustamize IP of clock: CLK-LN1: sys clock, EXT-RESRT-LN : custom output clocks: Active Low. Then ran the connection automation again. For that select all the connections, but change the ext-reset-ln to custom. Therefore we can remove the unwanted reset-o block. Then

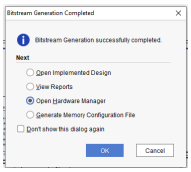

Fig. 9. Bit Stream Genaration

we added the UART protocol to the IP block, then we added necessary PMOD sensors such as Bluetooth, ultrasonic and ambient light sensors to the IP block. Meantime we assigned the necessary Pmod ports to the sensors and Bluetooth modules. Afterwards, we created HDL rapper and generated the bitstream. The next step would be launching the Vitis IDE to program the device.

B. Software

To program Microblaze, we utilised the Visual Studio IDE and the C programming language for software implementation. When bitstream generation is successful, hardware files can be exported in wrapper format. The xsa file format aids in the development of project platforms. The platform project's goal is to develop an application that takes advantage of the hardware that we have configured for debugging. Fig 8 shows the application project created and the files included in the application project.

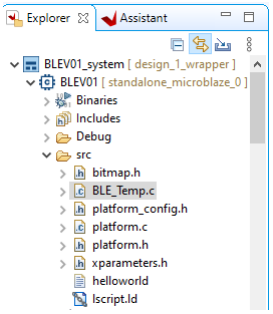

Fig. 10. Application project interface

The project's C program for debugging and testing is written in the primary source file, BLE.Temp.c. Before programming the code to the device, build project tool helps to check errors and gives a clear idea of where error occured in the Vitis IDE software platform, which aids in project construction and error checking.

Our Microblaze-based device primarily makes use of an external PmodBLE Bluetooth sensor and an integrated temperature sensor. We need to include library files like PmodBLE, xiic, and other libraries in order to configure and access the sensors and program it.

```
15 #include "xil_cache.h"
16 #include "xparameters.h"
17 #include "xil_cache.h"
18 #include "xparameters.h"
19 #include "PmodBLE.h"
```

Fig. 11. Libraries used

The "xparameters.h" file included in the code is a generated header file that contains macro definitions for various parameters and configuration settings related to the hardware platform. These kinds of uses are also made of other libraries.

A temperature sensor, Bluetooth connection module, and UART (Universal Asynchronous Receiver-Transmitter) configuration block were established in a project once the required libraries were configured. It adapts to diverse target platforms,

notably distinguishing between systems utilising the MicroBlaze CPU and those using other processors, such as ARM in Xilinx's Processing System (PS).

```
void DemoRun() {
    u8 recv_buffer;
    u32 num_received;

    u8 buffer[] = "AT+CIPSTART=\"TCP\",\"192.168.70.222\",80\r\n";
    u8 buffer_09[] = "AT+CIPSEND=8\r\n";  //send message to server


    int a = 39; // testing with a dummy value

    itoa(a, msg, 10); // converting the recieved temperature value to string to send over IP protocol to server

    int j = 0;
    int i = 0;
```

Fig. 12. DemoRun Function

An essential part of our Smart Home Automation project, intended for a MicroBlaze-based system, is the DemoRun() function in the code. This feature coordinates Bluetooth Low Energy (BLE) devices. It uses a continuous loop to manage data reception, conversion, and transmission. It initialises variables, such as receive buffers and AT instructions. Specific actions, including sending AT commands to establish a wireless connection and broadcasting temperature values over both BLE and UART interfaces, are triggered within the loop based on counters. These activities demonstrate how different parts of our smart home system are integrated. The precise implementation of communication functions may require further attention during development. Figure 10 shows the function code mentioned above.

In the DemoRun function of our Smart Home Automation code, when the loop counter 'i' hits 200,000, temperature data is obtained and converted to a string. The formatted message, including a temperature label, is then sent over Bluetooth Low Energy (BLE) and UART interfaces for real-time monitoring. Use of itoa command Strcat helps to transmit string with the temperature data. This concise yet crucial part of the code illustrates the seamless integration of temperature sensing and communication functionalities within our Smart Home Automation system, enabling efficient transmission to both BLE and UART endpoints. Fig 11 shows the code for the function to send string and temperature readings.

```
if (j == 100000) {

    j = 0;
    if (i == 100000) {
   //     BLE_SendBuffer(&myDevice, buffer_09, 14);

    }

    if (i == 200000) {
        print_temp_reading();
        itoa(global_temp, msg, 10);  //converting to string


        char temp_data_string[] = "\r\n ";
        char msg_temp01[] = " Temp - ";
        char msg_temp[] = " \n";
        char nxt_line[] = "\r\n";
        strcat(msg, msg_temp);  // concatenating temp to string
        strcat(msg, msg_temp01);  // concatenating temp to string
        int  n = BLE_SendData(&myDevice, msg, strlen(msg));
        SysUart_Send(&myUart, msg, strlen(msg));
```

Fig. 13. DemoRun Function used to transmit data

the temp-init function initializes a temperature sensor through the TempSensorExample function, retrieves the temperature value, and prints it to the console. The success or failure of the initialization process is indicated by the return status. This function is likely part of a broader temperature-sensing system within the project.

The main function acts as the project's central control hub, including other functions that need to access the uart Bluetooth and temperature sensor overseeing initialization and execution. It starts by configuring Bluetooth Low Energy (BLE) settings with DemoInitialize-BLE, performs general system setup with DemoInitialize, and initializes the temperature sensor subsystem through temp-init. The core project functionality is executed in DemoRun, likely related to home automation or monitoring. After ensuring a clean termination by disabling caches with DisableCaches, the function returns XST-SUCCESS. Commented-out code hints at potential future expansions. Finally, DemoCleanup is called for cleanup operations. Overall, the main function efficiently manages the project's key stages: initialization, execution, and cleanup

IV. Implementation Chanllenges

A. Getting used to the vivado platform.

The first problem we had was group collaboration using different vivado versions. As an example, one collaborator has Vivado 2021.2, and the other person has 2022.1. When we merge projects, it produces some errors. The solution to this challenge would be for all the collaborators to use the same version.

While using Vivado 2022.1, we could add Pmod blocks, but when we assign related Pmod ports to the FPGA, the connection ports are not displayed. That was the second reason to move to version 2021.2.

It was quite a difficult task to find the related IP library files for the Nexys A7 100T. At the beginning of the project, we used the board files of the Nexys 4 board; therefore we found some warnings and errors when generating bitstream. Later we found the IP library for the Nexys A7, and this problem was solved.

Sensors such as ambient light sensors and ultrasonic sensors have only six pins instead of 12 pins. Because of that, it was quite challenging to connect the sensors to the board. The external temperature sensor is also not compatible with the Nexys A7.

V. Future Discussion

Since the pins did not support the Nexys A7 board Ambient light sensors and ultrasonic sensors could not be implemented. Since we had limited time to finish the project, it was unable to be tested with the new matching sensors. In the future, the sensors mentioned before can be implemented. This prototype can be developed with more sensors, such as an accelerometer, to detect earthquakes.

In future versions the FPGA can be connected to Raspberry pi via Bluetooth and then with the help of the Raspberry pi, the system can be upgraded to an IOT device, therefore the system can be accessed over the Internet.

VI. Conclusion

In conclusion, our project integrates the Nexus A7 FPGA board with MicroBlaze and various IP blocks to achieve a comprehensive sensor-based system. The incorporation of an onboard temperature sensor, ambient light sensor, ultrasonic sensors, and a Bluetooth module as PMODs enhances the board's functionality and enables a diverse range of applications.

The use of MicroBlaze allows for efficient processing and control, enabling seamless communication between the FPGA board and the connected sensors. Through the integration of IP blocks, we have optimized the implementation of specific functions, ensuring a streamlined and effective operation of the system.

The onboard temperature sensor provides valuable environmental data, while the ambient light sensor enables the system to adapt to varying lighting conditions. The ultrasonic sensors enhance the board's capability to measure distance and detect objects in the surroundings. The inclusion of a Bluetooth module as a PMOD facilitates wireless communication, expanding the project's connectivity options.

In summary, our project showcases a well-integrated and versatile system that leverages FPGA capabilities, MicroBlaze processing, and a variety of sensors to create a robust and adaptable solution. This documentation serves as a comprehensive guide to understanding the architecture, functionality, and applications of our Nexus A7 FPGA-based project.

References

[1] "Digilent reference," accessed: 2024-01-10. [Online]. Available: https://digilent.com/reference/

[2] S. Alonso, J. Lázaro, J. Jimenez, U. Bidarte, and L. Muguira, "Evaluating latency in multiprocessing embedded systems for the smart grid," *Energies*, vol. 14, no. 11, p. 3322, 2021.

VII. Affidavit

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 you that this paper has not been part of a course or examination in the same or a similar version.

Athukorala Ravindu
Name,Vorname
Last Name, First Name

Lippstadt 12/01/2024
Ort,Datum
Location, Date

Unterschrift
Signature

Umaibalan Kajeeban
Name,Vorname
Last Name, First Name

Lippstadt 12/01/2024
Ort,Datum
Location, Date

Unterschrift
Signature

Abhayawardana Lochana
Name,Vorname
Last Name, First Name

Lippstadt 12/01/2024
Ort,Datum
Location, Date

Unterschrift
Signature