Documentation
Smart Weather Monitoring System
12.01.2024

1 Team members (and Who did What)

a. **Moataz Elbayaa**
    - a. Labs:
        - i. Lab_4 (Both Exercises)
        - ii. Lab_6 (Both Exercises)
    - b. Final Project
        - i. Adding the MicroBlaze to the block design and setting it up.
        - ii. Adding and integrating the UART IP.
        - iii. Adding and integrating the Pmod TMP3 IP.
        - iv. Adding and integrating the Pmod CLS IP.
        - v. Getting the synthesis report, the Utilization Synthesis Design, the clock interaction report, Power summary report, and the Design timing summary.
        - vi. Getting the Implementation reports are in the repository, including the Place Design, DRC Opt Design, IO Place Design, and the Utilization Place Design.
        - vii. Writing the code to measure the temperature via Pmod TMP3 temperature sensor and printing it via the UART.
        - viii. Writing the code of sending the temperature which has measured via the Pmod TMP3 and the humidity which has been measured via the Pmod HYGRO via SPI protocol and display them on the Pmod CLS.
        - ix. Also, Writing the code of integrating the value which has been measured via the Pmod PIR and using it to turn off the Pmod CLS and the 7-segments to save power.

b. **Yashodhan Vishvesh Deshpande**
    - a. Labs:
        - i. Lab_2
        - ii. Lab_3 (Both Exercises)
        - iii. Lab_7
    - b. Final Project
        - i. Project concept idea, block diagram and project specification document.
        - ii. Adding Microblaze processor design to block design and the AXI interface.
        - iii. Writing the VHDL code for 2-digit 7 segment display. Synthesis, implementation and bitstream generation.
        - iv. Creating the AXI display ip from the vhdl code. Changing the AXI ip top file and bottom file to make the ip adapt AXI interface.
        - v. Connecting the AXI display ip with the microblaze block diagram successfully and testing.
        - vi. Researching about the Pmod HYGRO IP. Found functionality information for relevant functions along with examples. The examples helped understand the IP functions making it easier to code.

vii. 2 Flowcharts in the documentation.

c. **Emirkan Sali**

a. Final Project

i. Adding the full Microblaze microprocessor to the block design and setting other component IPs up.
ii. Working on the seven segment Display IP and setting up in the block design.
iii. Setting up Pmod HYGRO and Pmod PIR in the block design and connecting it to the Microblaze microprocessor.
iv. Writing C code to operate the seven-segment display as intended.
v. Writing C code to operate the Humidity sensor Pmod HYGRO and read temperature and humidity values.
vi. Writing C code to operate and read the PIR sensor values and setting up its functionality.

2 Introduction

In Chip design IP-Cores and IP-design in general is a useful technology to save development time and costs by packaging integrated circuits or parts of ICs into smaller IP cores that can be interconnected for full functionality. These IP-Cores can be memory modules, processor modules, cache or interface modules of a bigger system. Companies can also produce IPs for specific functions and license and sell those to other manufacturers or people. With all that being said, IP-Cores are used in a vast amount of modern chip designs and in FPGA development. Especially the usage in FPGA prototyping is useful since it is a programmable logic device. Using an FPGA and a library of IPs, one can develop almost every system they have in mind. For our project, we wanted to develop a smart weather monitoring system which can primarily measure the outside temperature and display it for the user and secondarily also the humidity. All this will be implemented using the MicroBlaze processor IP and more IP blocks for the advanced functionality.

3 Concept and Task description

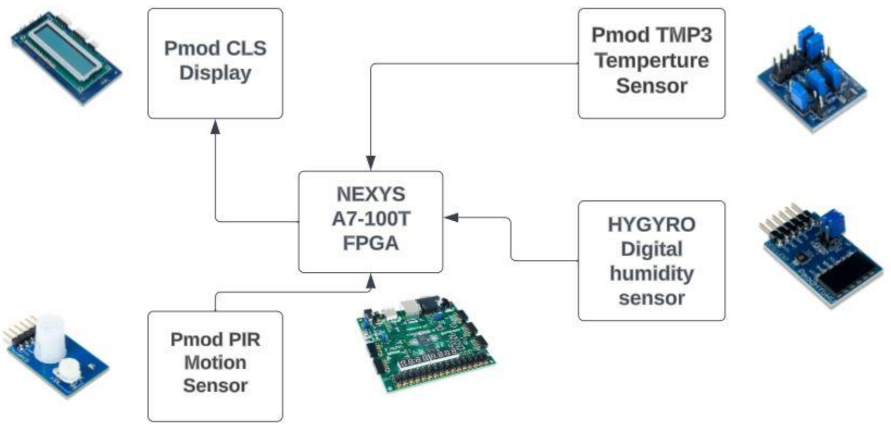


Figure 1 Block diagram of the system concept

A smart weather monitoring system that is designed to be an Internet of things (IoT) application on an FPGA board. For this project, the NEXYS A7 100T FPGA board is used. The finalized project should be able to measure the room temperature using a Pmod TMP3

external sensor and display it on the internal seven segment display. Also, display it on the Pmod CLS.
Also measuring the humidity via Pmod HYGRO and displaying it on the 7-segments and the Pmod CLS. In addition to using Pmod PIR to turn off the 7-segments and the Pmod CLS to save power in case of that no movement around which means that there is no user near the display to read the values and in the same way the Pmod PIR can be used in different applications.

4 Project/Team management

To realize our project concept, we had to coordinate by dividing tasks into doable with the FPGA board and doable without the FPGA board since there is only 1 board to work with. Tasks like creating diagrams for the system concept, creating an initial block design in Vivado without testing and documenting progress would be done by group members who don’t have the board. Tasks like testing C code and work with the sensors would be distributed to the group member who currently had the FPGA or be done at a meetup at the university together.

Divided tasks for each member at the very beginning of project:

1. Moataz: Creating a MicroBlaze block and connecting temperature sensor to the FPGA board and linking it correctly with the Microblaze processor to obtain the data.

2. Yashodhan: Creating a microblaze block design and most part of the concept design, project specification document, 7-segment display AXI IP and the VHDL code. Flowchart for VHDL code. Assisted in realization of the humidity sensor functionality. Block diagram.

3. Emirkan: Working on microblaze block design, testing and writing C code to use obtained data for user feedback on 7-segments.

At the end, who did what of the whole project is mentioned at the beginning of the documentation.

5 Technologies

For our project, different technologies were used to implement the system on an FPGA:

- VHDL: VHDL is a hardware description language (HDL), which makes it possible to describe digital systems on transistor level. Together with the Verilog HDL it is one of the most used HDLs in the world. VHDL mostly describes the behavior of certain electronic modules that its user wants to create. Furthermore, it allows for testing and simulation and synthesis into a netlist to create actual electronic systems from. However, it is important to note that not every VHDL code is not synthesizable. VHDL was used to create custom IP-Cores to be added into our final microblaze design.
- FPGA: FPGA, short for Field Programmable Gate Array is a device often used in prototyping, containing a vast amount of digital building blocks with which many different circuits can be realized. It also contains, depending on the board, different onboard sensors, displays, interfaces and I/Os also GPUs like FPGAs used in Automotive Applications to aid in prototyping and development by providing potentially needed hardware in one programmable package.

- Pmod TMP3: The Digilent Pmod TMP3 is a temperature sensor module built around the Microchip TCN75A. With a resolution of up to 0.0625°C, users can effortlessly measure a broad spectrum of ambient temperatures, ranging from -40°C to +125°C.

  In our project, Pmod TMP3 has been connected via I²C communication protocol which is connected over the header J1.

  This requires having a connected jumper for JP4 and JP5. A connected jumper over A0 and GND in JP1. A connected jumper over A1 and GND in JP2. And a connected jumper over A2 and GND in JP3.

- Pmod CLS: It is a 16×2 liquid crystal character display which allows wide range of instruction functions and has up to 32 user definable characters and allows connection via multiple communication options including UART, SPI, and I²C.

  In our project, it has been connected via SPI communication protocol which is connected over the header J1.

  Selecting the SPI as the used communication protocol is done by setting the mode jumpers MD0, MD1, and MD2 of JP2 on the board. A missing jumper is represented by 0 and a connected jumper is represented by 1.

  And selection SPI requires MD2 = 1, MD1 = 1, MD0 = 0. which means a connected jumper for MD2, also connected jumper for MD1, and missing jumper for MD0. Also, for JP1 there are no connections.

- Pmod HYGRO: it is a relative humidity sensor with an integrated temperature sensor for highly accurate measurements at low power. It has a relative Humidity Accuracy ±2% and Temperature Accuracy ±0.2°C. Also, it provides an Excellent Stability at High Humidity. It is connected via I2C communication Interface.

- Pmod PIR: the Pmod PIR motion sensor is a Passive Infrared Sensor for monitoring movement at low power. It can detect movement up to 5 meters away. It has 6-pin Pmod connector with GPIO interface.

- Xilinx Vivado Design Suite: The design program for AMD adaptive SoCs and FPGAs is called Vivado. Design Entry, Synthesis, Place and Route, and Verification/Simulation Tools are among its features. With system-to-IC level tools and a shared scalable data model, it is an integrated design environment (IDE) with a common debug environment. Vivado comprises standards-based IP stitching and systems integration of all kinds of system building blocks; standards-based IP packaging of both algorithmic and RTL IP for reuse; electronic system level (ESL) design tools for synthesizing and verifying C-based algorithmic IP; and the verification of blocks and systems. Designers have access to a constrained version of the design environment with Vivado Web PACK Edition, which is available for free.

6 FPGA Implementation Results

The Implementation Results including the synthesis report, the Utilization Synthesis Design, the clock interaction report, Power summary report, and the Design timing summary are included in the repository.

Also, Implementation reports are in the repository, including the Place Design, DRC Opt Design, IO Place Design, and the Utilization Place Design.

For our project, we first had to create a microblaze microprocessor in the Vivado IDE as a foundation to handle all the code and data that needs to be handled. We installed some Ram from the internal memory of the FPGA to our board and a clocking Wizard that can output differently paced clock signals if we need them for later components as seen in figure 2.

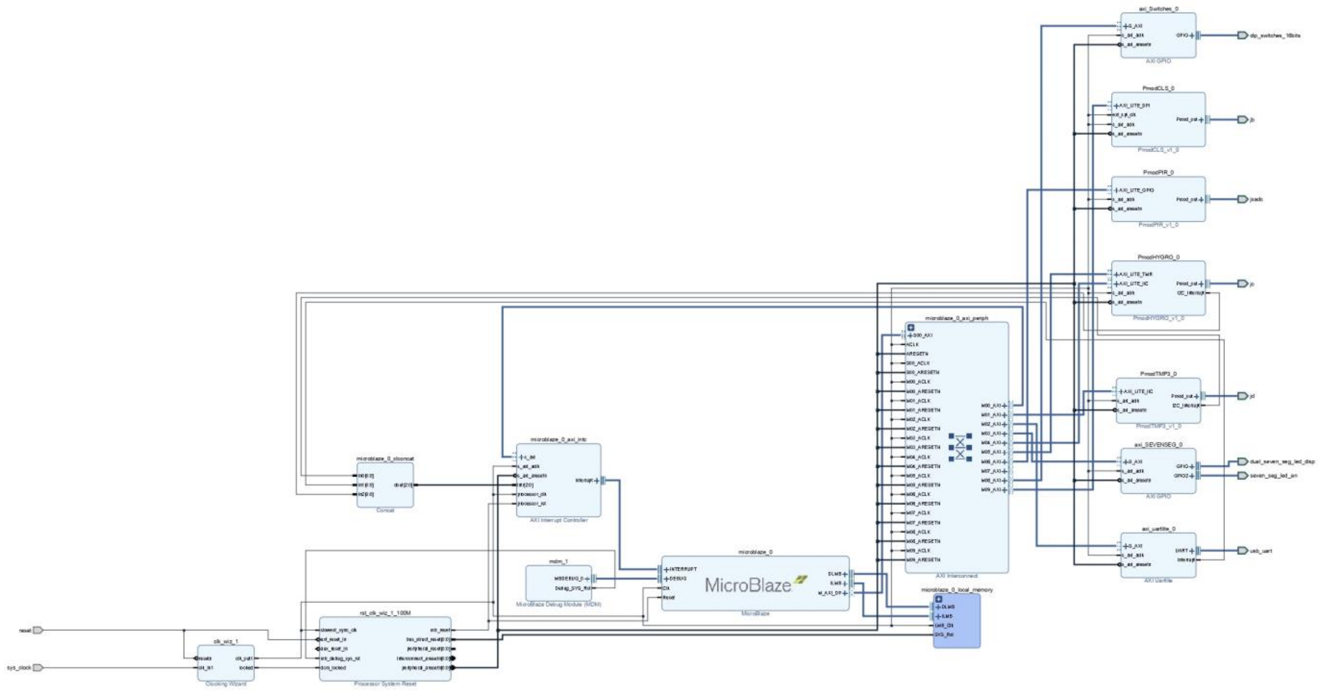
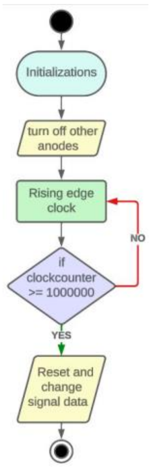
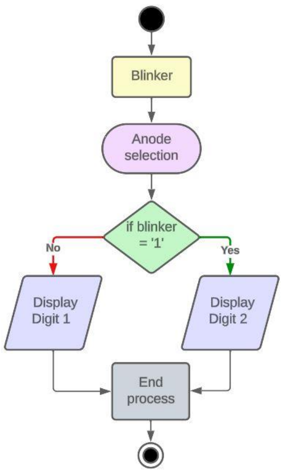


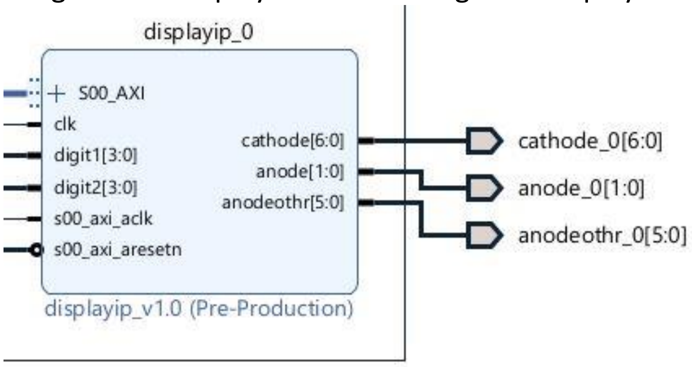
Figure 2 block design of our project

The custom AXI IP for the 7-segment display was created in VHDL programming language. The initializations for the IP were designed initially to include the registers for anodes, cathodes, input of digits and the clock. The IP was designed by first creating a process for internal clock that would be used for timing purposes to ensure the anodes of the seven-segment display function coherently. After 1000000 clock cycles the value of the signals blink and blinkers will be toggled and the clock counter signal will be reset. The following functionality of the IP is included in the Picture 4 flowchart. Then another process starts that is triggered by the toggling of the blink signal. This process is designed to select the anodes which would choose exactly which of the 8 seven segment displays will be used. The blink and blinker signals will choose the anodes while the unused anodes are switched off. Only 2 displays are used out of the eight hence only two anodes will need to be coded. One anode will be off while the other will be on. These two anodes will switch to display their respective display values as per the clock signals. To display the digits the input signal will be given to the cathodes by selecting the 4-bit binary value of the digits to be displayed. The following functionality is displayed in figure 5 of the flowchart.

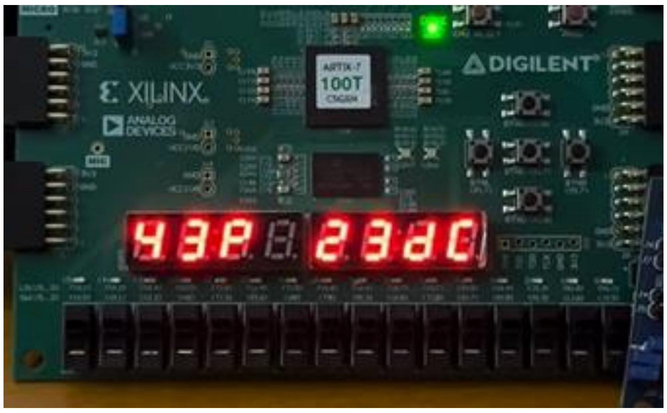




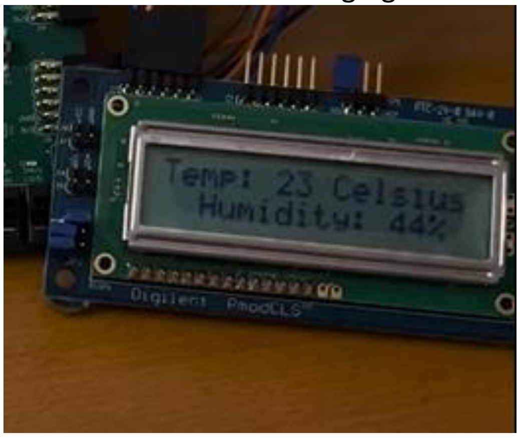
The AXI IP was then created in vivado by using the create IP tools. The necessary changes in the AXI top file and the source file were carried out to declare ports, signals and logic. Upon finishing this step, the IP was then added to the catalog and connected to the MicroBlaze processor. The figure 6 show the image of the IP block that was created. The digit1 and digit2 ports were mapped in I/O planning to the switches to test the functionality of the whole microblaze block design including the AXI IP. The test was successful and the selected 2 digits were displayed on the 7-segment display.



Having added and connected the seven-segment display as an IP in the block design, the next step was to write C code for the foundation of our final project state and the operation of the seven-segment display. The communication with the seven-segment display would be handled by our design's gpio ports, so we needed the library xgpio.h in our C code. Generally, the xgpio.h library is a header file that provides an interface for working with gpio peripherals in Xilinx FPGA designs. It serves as a driver for ther AXI GPIO IP core. With the information that the seven-segment display in the Nexys A7 100t is a common cathode multiplexed seven-segment display, we needed to build our code around that functionality. To display more than one digit or symbol at the same time we needed a loop with built in inbetween each symbol that is to be displayed. With xgpio, firstly we needed to establish the gpio channels of the anodes and kathodes. In channel 1 we send the appropriate bit sequence to show the symbol or digit we want and in channel 2, we decide on which section of the display it will be shown. After figuring out the hexadecimal values for the sections of the display and the bit sequences for displaying each digit and symbol, we wrote the code to send the data to the internal seven-segment display. Using XGpio_SetDataDirection(), XGpio_DiscreteWrite() and some functions for proper orocessing of the values to be displayed, we set up the communication and sent the data in a way that the desired result is on the seven-segment display as shown in the following figure:



In the next step, we added the Pmod HYGRO and Pmod PIR to our block design as sensors for getting the temperature and humidity data that we want to show to the user on the display. Having added the IPs from the digilent library and connected everything with the other IPs of our design, we moved on to write the necessary C code to make both sensors deliver the data that we needed. Using the PmodHYGRO.h and the PmodPIR.h libraries in the C code helped in achieving the needed functionality. To gather the data, we simply needed to initialize both sensors in the C code and use the library functions to get the desired temperature and humidity values. These were implemented into the main loop to show the gathered data on the seven-segment display as shown in the previous figure. The low power infrared sensor was used to determine movement infront of the device and activate the displays and sensors of the device. If no movement is detected, the displays will turn off and the sensors will not read any data to ultimately save electricity from being wasted.
From there on, we also added an optional LCD display to show more clearer values to the user and also show the temperature in Celsius and Fahrenheit as desired by the user showcased in the following figure:



7 Conclusion and Evaluation

In summary, our project successfully integrated the Pmod TMP3 temperature sensor, Pmod HYGRO Humidity sensor, Pmod CLS display, 7-segment display, and Pmod PIR motion sensor. We achieved our goals of accurately measuring temperature and humidity, displaying the data efficiently, and adjusting the display based on detected motion.

In conclusion, our project demonstrates our ability to deliver innovative solutions in IoT, paving the way for future developments in smart environments.

8 Sources/References

[1] https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual
[2] https://digilent.com/reference/pmod/pmodcls/start
[3] https://digilent.com/reference/pmod/pmodhygro/start
[4] https://digilent.com/reference/pmod/pmodpir/start
[5] https://digilent.com/reference/pmod/pmodtmp3/start
[6] https://digilent.com/reference/learn/programmable-logic/tutorials/pmod-ips/start

Code and sources will be found on GitHub:
https://github.com/emirkanS/Special_Emph_B_Lab_Yash_Moat_Emir.git
A video of the project can be found on YouTube with this link:
https://youtu.be/Jx6yT_UQWvM

A Video of the Implementation of Lab 2: Lab 2 - Implementation Video
A Video of the Implementation of Lab 3 Exercise 1: Lab 3 - Exercise 1
A Video of the Implementation of Lab 4: https://youtube.com/shorts/0z0wbmVGFrk
A Video of the Implementation of Lab 6: https://youtube.com/shorts/oCFRxPuv4JY
A Video of the Implementation of Lab 7: Lab 7 - Implementation Video

Lab 7 exercise has been changed directly to the implementation of custom IP for 7-segment display instead of PWM custom IP. Since it was allowed in the exercise to **create any custom** AXI IP with the microblaze block diagram.