FPGA Based Weather Sensor Interfacing

Andrew Meesseman, Andrew Galczyk, Adam Marszalek, August Lile
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
adammarszalek@oakland.edu, atgalczyk@oakland.edu, ameesseman@oakland.edu, alile@oakland.edu

***Abstract*— This project has been created to interface with an external weather sensor, the BME280, via an FPGA acting as the host using VHDL to implement the circuit. This external weather sensor has temperature, humidity, and pressure measuring capabilities which are collected periodically. Sensor values that are captured will be reformatted and displayed on an external Hitachi HD44780 LCD.

I. Introduction

This report will cover the process that was developed for the experimentation and testing of this project. Vivado was used to write the VHDL that was used to control the FPGA and create all external interfaces. Many logic components were used when designing the weather sensor system such as registers, shift registers, logical computation units (adders, subtracters, multipliers), multiplexers, decoders, and several finite state machines (FSM). Results of the project will also be discussed in detail to understand future project viability. How these elements were created, simulated, and tested will be further discussed below.

II. Methodology

The system will consist of an interface between the BME280 and FPGA, a circuit to handle the weather data and reformat according to how it will be displayed on the LCD, and the interface between the LCD display and FPGA. The FPGA interfaces with a BME280 sensor in order to receive temperature, humidity, and pressure data. This data can then be reformatted accordingly and displayed on the Hitachi HD44780.

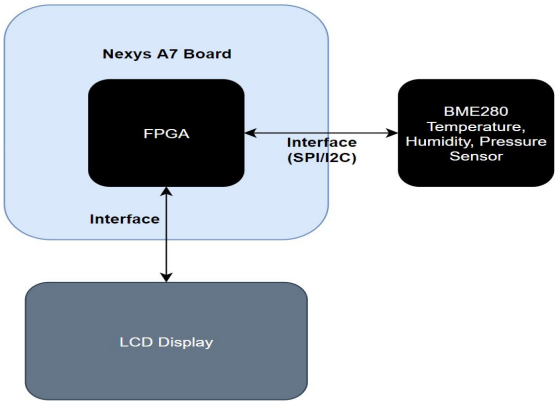

Figure 1. High Level Design

The high level design of the Weather sensor is fairly simple. The FPGA will communicate with the BME280 weather sensor via Serial Peripheral Interface (SPI) in order to write to registers on initialization and to read data from different data registers. Temperature, pressure, and humidity data that is collected indefinitely will then be processed and displayed on the LCD, a Hitachi HD44780.

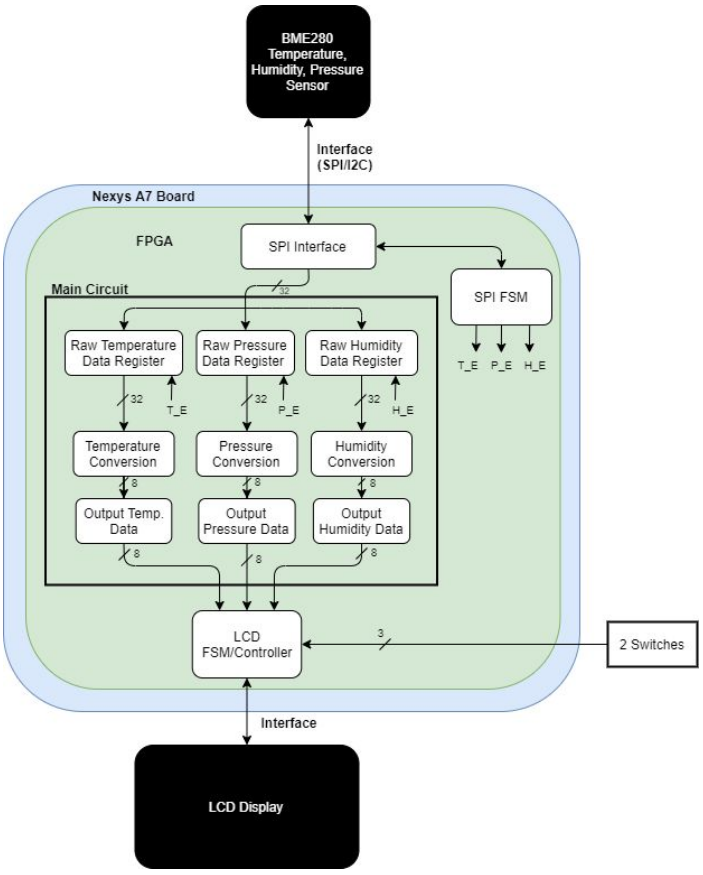

Figure 2. System Diagram

The system consists of three main portions. There is a circuit designed to interface with the weather chip over SPI and buffer in raw temperature, pressure, and humidity data periodically. This circuit consists of a state machine designed to first retrieve values that do not change and need to be used for calculations.

Following this, raw temperature, pressure, and humidity data will be collected continuously and get buffered in. The next circuit is the main circuit, used to convert the raw data into a format that can be read. The final circuit is designed to interface with the LCD and display one of four formats: temperature, pressure, humidity, and welcome screen. The circuit can multiplex what is being displayed with switches or a timer circuit and using weather values in registers that are periodically being updated in the other portions of the circuit.

In the final system it is desired to implement all of the functionalities provided by the BME280: temperature, pressure, and humidity readings. The major portion of implementing these was converting the values that are received from the sensor into compensated data. All values received from the sensor are output ADC values. As an example, the subsequent figures show the circuits that are necessary to convert the raw data into a compensated value that can be displayed. In order to design the circuit, the datasheet provides C code functions to convert the respective collected ADC values into either decimal or Q-format fixed-point numbers, which is what the system was designed around. The steps of the code were converted into an asynchronous circuit.

A. SPI Interface

In order to communicate with the sensor, a standard 4-wire SPI controller was implemented. The standard 4-wire version includes outputs of chip select, a clock, and MOSI (master out, slave in), or SDI. The input of the controller is MISO (master in, slave out), or SDO. The architecture of this controller includes two FSM's: one for controlling the PHY SPI communication and another as an overlay controlling the flow of what data is to be sent to the sensor. The latter FSM is detailed below.

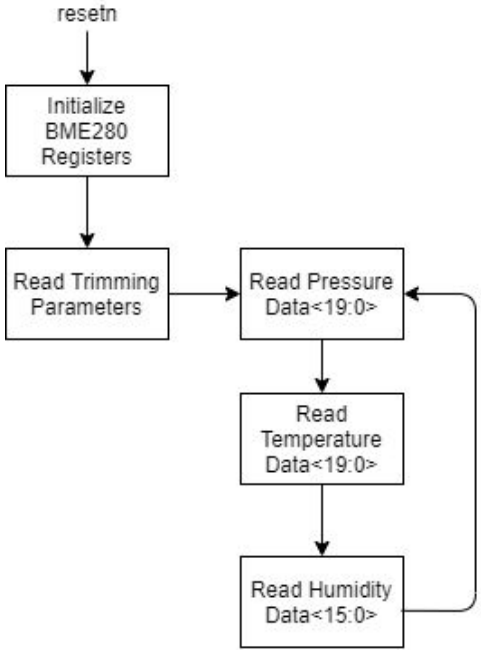

Figure 3. High Level SPI FSM Design

When the system boots up, the SPI controller writes to several registers on the sensor in order to set up some values for the sensor to abide by. For example, a Power-on Reset (PoR) is performed, the data polling rate is set to 500ms, and oversampling and filtering modes are set.

In the next sequence, 32 values that are stored in the sensor's non-volatile memory (NVM) are read and stored once. These values are stored as the trimming parameters and used by respective conversion circuits to convert the ADC values into physical data.

The final sequence of this FSM includes a loop that continuously reads out the ADC values for temperature, pressure, and humidity from the sensor. This is the most important data, and the values that are collected by the sensor periodically to update the weather data.

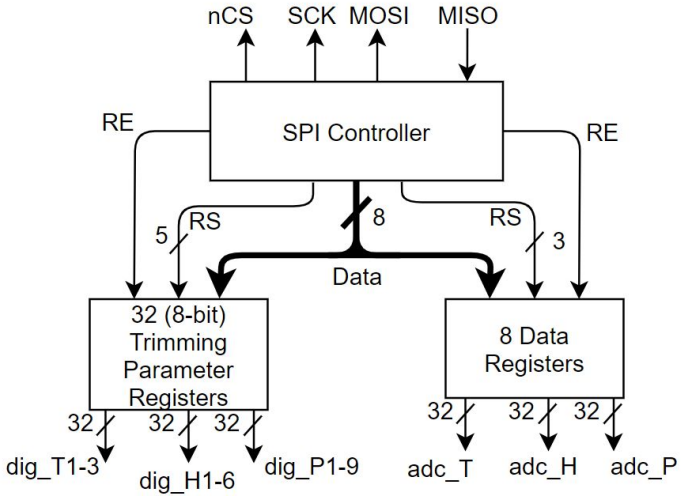

Figure 4. SPI Interface Architecture

The diagram above shows, from an architectural standpoint, how the SPI interface was set up. As mentioned earlier, the interface has two FSM's. Both of these are in the block labeled "SPI controller." The two separate blocks contain registers for the 32 8-bit registers for the trimming parameters and 8 8-bit registers for the ADC values. These values are mapped to different conversion circuits depending on which circuits need these values as inputs.

B. Conversion Circuits

Section 4.2.3 (rev. 1.6) of the sensor's datasheet gives a way to convert the raw ADC values for temperature, pressure, and humidity into data that can be displayed [1]. It provides example C functions to implement these conversions, along with an API. Given that a microcontroller is not being used, it was necessary to convert the logic in these functions into a series of logic gates that would essentially perform the same operations and output the results as the functions would. Given the rather complex conversions needed for some of the data, these conversion modules were arguably the biggest part of the circuit. They use proprietary data types understood to be signed or unsigned 32 or 64 bit integers. For VHDL, these were implemented as logic vectors of equivalent size. Taking sign into account, temperature

may be negative, but pressure and humidity cannot be negative values.

a. *Temperature*

The function to convert the temperature receives data in 32-bit signed format. When this value is converted to decimal, the two least-significant digits are considered to be the decimal points of the temperature. For example, a value of '5123' would equal 51.23 degrees Celsius, hence why the circuit converts the binary output to BCD in order to determine the two fractional digits. The temperature conversion circuit also creates the value t_fine, which is used in the pressure and humidity conversion circuits as well. The humidity and pressure rely on a highly precise temperature measurement to interpret physical values for those parameters. The circuits developed to convert temperature are shown in Figures 5-7.

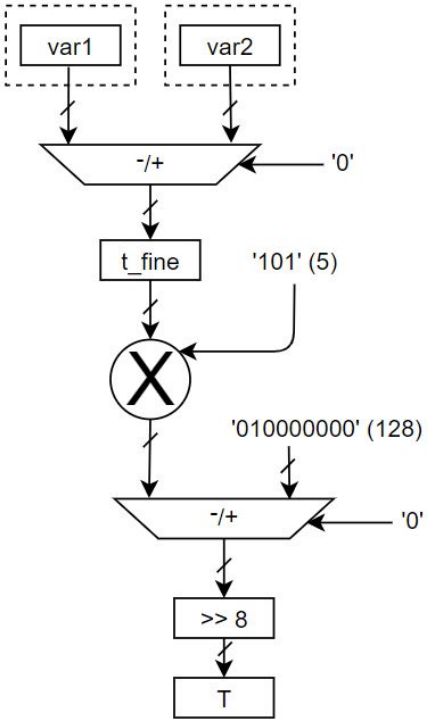

Figure 5. Temperature Conversion Final Calculation

The image above shows a block diagram of the circuit that was developed from the temperature conversion function. The values 'var1' and 'var2' at the top are not black boxes or values collected from the sensor, but rather values from different circuits used in the calculation of temperature. These two circuits can be seen in Figures 6 and 7, below.

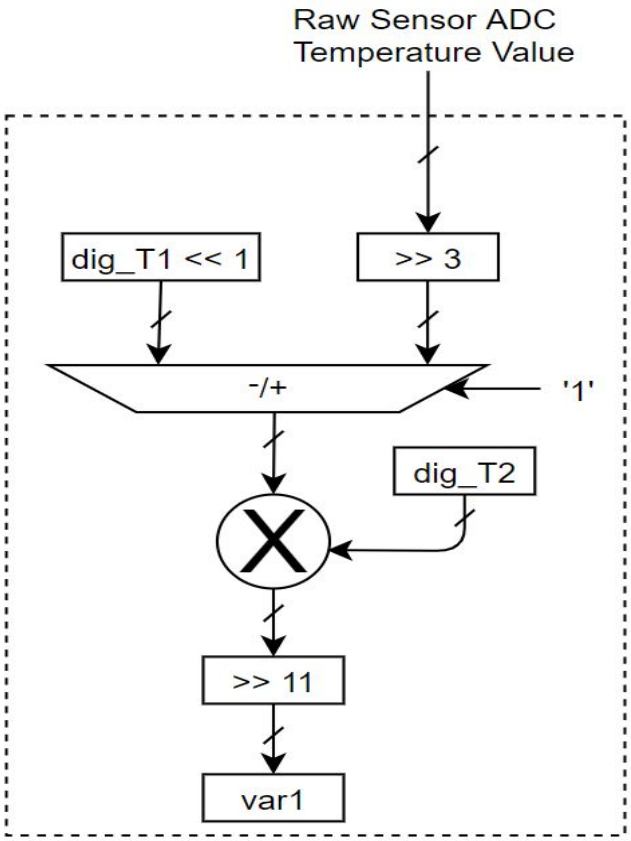

Figure 6. Intermediate Temperature var1 Calculation

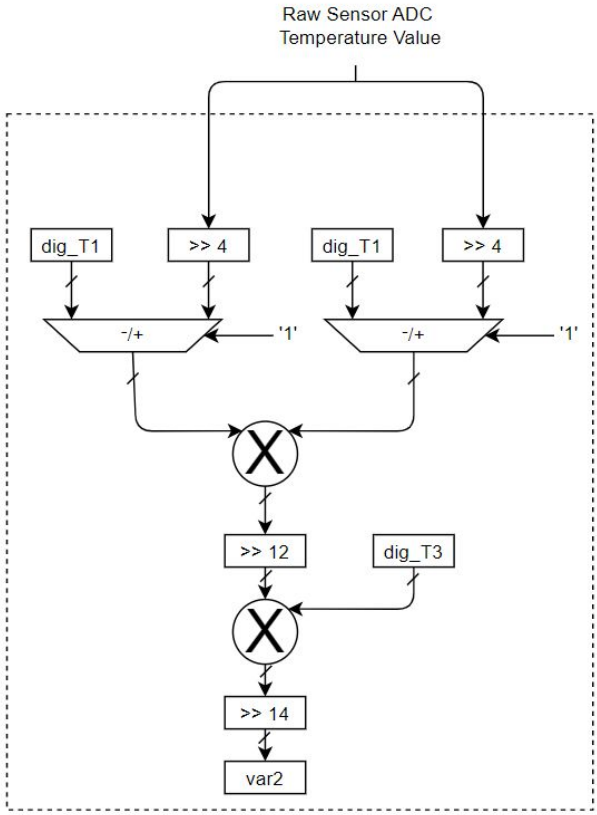

Figure 7. Intermediate Temperature var2 Calculation

b. *Humidity*

The function to convert humidity receives data in 32 bit signed integer format. It then converts the data into a 32bit value saturated to the range [0, 419430400]. The output value can then be interpreted by dividing by 1024, shifting the data 10 bits to the right. This value then corresponds to the value of %RH (relative humidity) between 0% and 100%, with 3 decimal points of decimal precision. The circuit that is derived from the humidity conversion function is shown below in Figure 8.

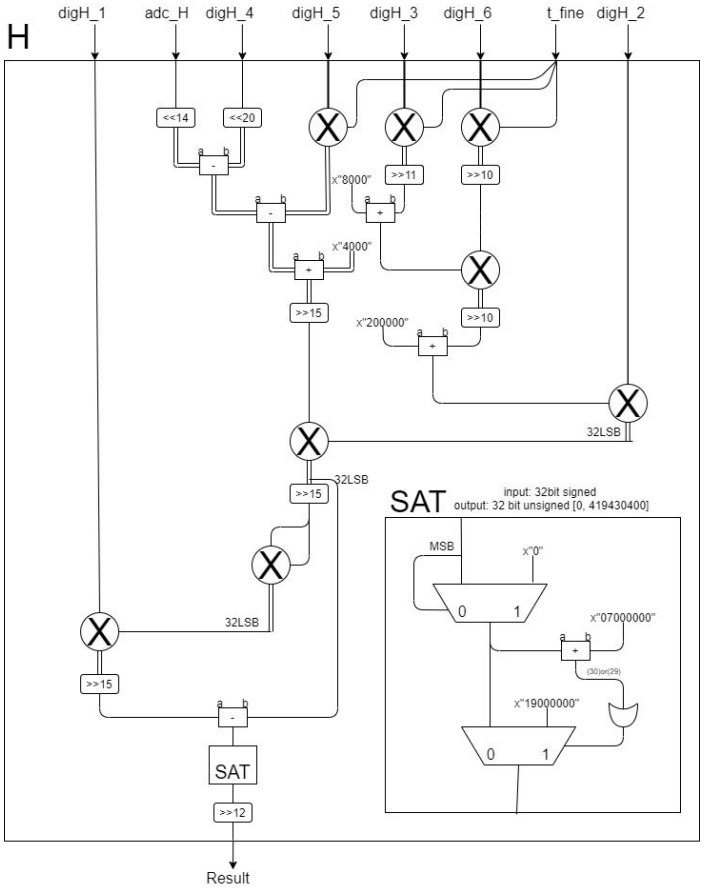

Figure 8. Humidity Conversion Digital Circuit

c. *Pressure*

The function to convert pressure receives data in 64bit signed value format and returns the converted value as a 32bit unsigned integer in Pascals with one decimal precision. It is more common to interpret pressure data in hectopascals, standard atmospheric pressure is 1013.25 hPa. The circuit that was developed to convert the pressure is shown in Figure 9.

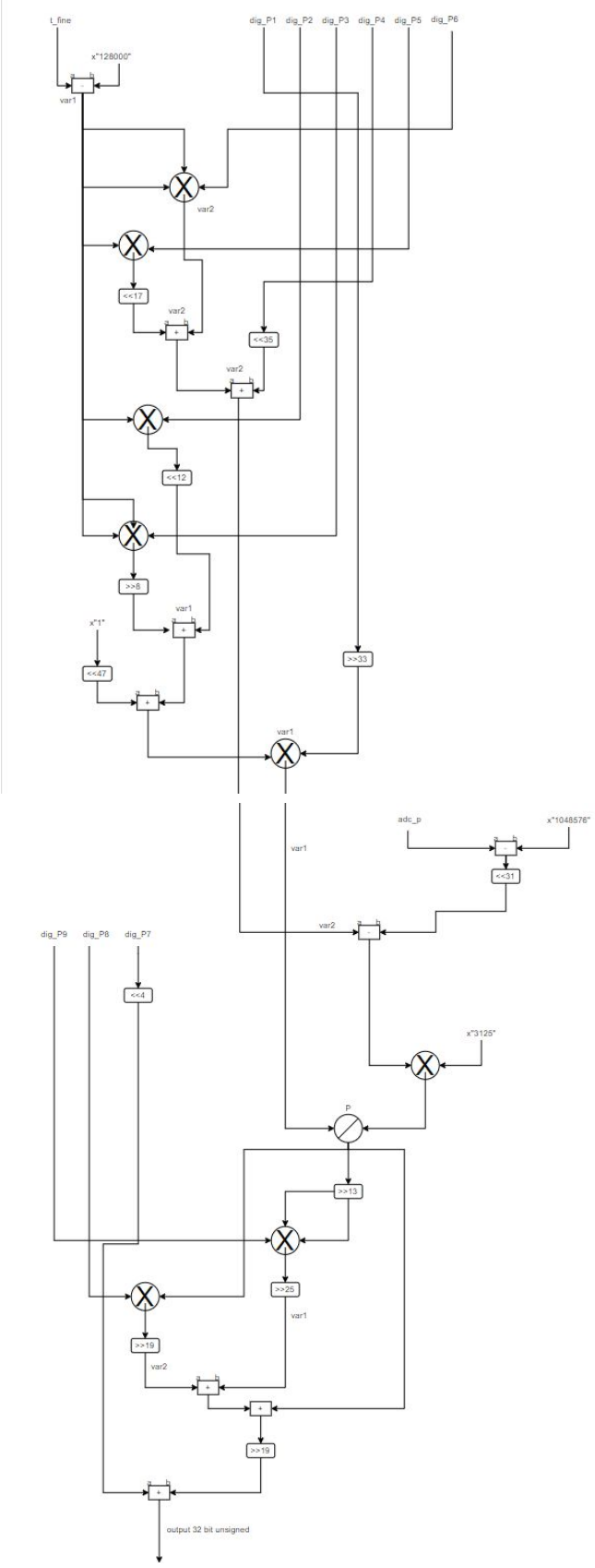

Figure 9. Pressure Conversion Digital Circuit

C. LCD Interface

The LCD uses 8 switches as inputs and a register select and display enable pins. Characters are able to be displayed using the character codes on page 17 of the LCD datasheet [4]. Upon startup, the SPI interface retrieves the values and the internal logic converts the raw data and then cyclically retrieves the raw sensor data. It is also necessary to clear the LCD as this is needed to switch between the welcome screen and temperature, pressure, and humidity data. Each of these messages are hard coded to be unique, except for the data from the sensor. This data is input into a 16to1, 8 bit mux which has its select line tied to a counter through and FSM. This FSM has an internal counter for waiting to output each character with a given delay. The output from the mux and the enable can be seen coming out of the 4 blocks on the left side of Figure 10.

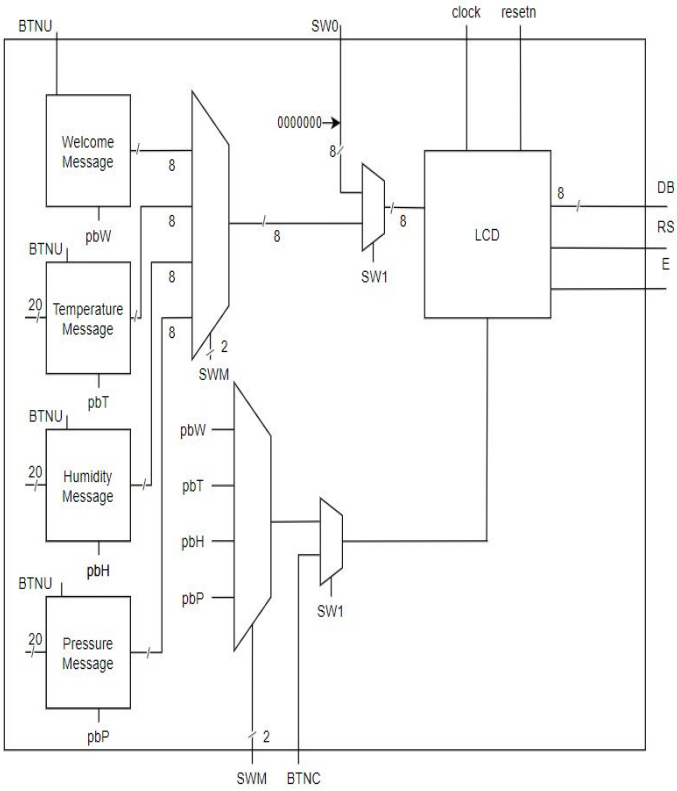

Figure 10. Pressure Conversion Digital Circuit

D. Finite State Machines

The project uses three finite state machines to control event timing such as reading from the data registers on the sensor and writing to the LCD. Therefore these are described here as FSM_LCD which controls the output and FSM_SPI which controls the input. FSM_LCD also handles the user input from the switches.

III. Experimental Setup

In order to test and make sure that each component is functioning correctly, testbenches were created. Now that all the components have been created, they can be tested on the FPGA. The version of Nexys board used was the Artix 7 - 100t. In order to test it, the sensor needed to be safely subjected to varied "weather" conditions. This can be accomplished at home by carefully subjecting the board to testing to alter the data output from the sensor. To alter the temperature is easy enough, one only needs to place their finger on it to change the temperature near the device. The pressure could be altered by carefully using a hair dryer or vacuum cleaner to change the air pressure around the device. Humidity is the most difficult to test as it could very easily be disastrous for the device if it were to get wet in the process. To alter the humidity condition in a home environment would be done most easily by using a shower to fill a bathroom with steam. The device could then be brought into the room to detect a change in the humidity. This test is the most dangerous of the bunch and is not likely to be performed due to the risk.

IV. Results

The results obtained show that the circuit functioned as intended. However, due to issues with calibration, it was not possible to obtain accurate results for some data. The implemented design was able to read data from the sensor for temperature and pressure. For humidity, because of the narrow range of values allowed out of the saturation component and the errors in calibration, the only output value shown was 0.00% RH. Due to the additional difficulty in testing for different humidity conditions, it was not feasible to obtain different results for this measurement within the timeframe the team had to work on the project.

Conclusions

This project has concluded that it is possible to interface an external component with the FPGA and export the measured data to an LCD. Given the complexities of some of the conversions, some of these functionalities may be better implemented on a microcontroller with dedicated hardware for floating or fixed point operations as intended for some of the conversions as it would yield more accurate results in the end.

One possible way this project could be improved is calibration of the readout. It was proven that output taken from the sensor could be converted in such a way that it could be displayed on the LCD. However, the data does not show accurate results corresponding to instantaneous weather conditions.

A real challenge faced by the group was operating during the COVID-19 pandemic. The group made regular contact during the development period, but the inability to meet and work on our parts together created a notable latency as well. Members of the group developed their parts separately, but for testing and observation, a download and upload cycle was needed. This created a huge delay and was detrimental to the project overall.

References

[1] Link to information pertaining to the sensor.
https://www.digikey.com/catalog/en/partgroup/bme280/62478

[2] Link to the board that was purchased for the project
https://www.dfrobot.com/product-1606.html?gclid=EAIaIQobC
hMIrL6F4Jui6AIVWv7jBx26BwmrEAkYCiABEgJxAvD_BwE

[4] Link to download the LCD datasheet.
https://www.sparkfun.com/datasheets/LCD/HD44780.pdf