**FPGA-Based Pulse Oximetry Device**

Matthew Capuano

Technical Report Submitted in Partial Fulfillment of the

Senior Electrical Capstone Proposal Course at Penn State Harrisburg

EE 406 – 001

Spring Semester, 2020

Submitted 5/6/2020

**ABSTRACT**

An FPGA-based pulse oximetry device has been designed in order to expand on the existing embedded system being used for a brain-machine interface by Penn State’s Brain Science and Neural Engineering group. The FPGA-based design creates an embedded system that can interface with multiple peripherals simultaneously, process data in real time and in parallel, and also send data wirelessly to a cloud server. The FPGA contains an FFT-based blood oxygen and heart rate processing algorithm and interfaces with a mixed-signal front-end chip for data acquisition, as well as a WIFI module for wireless transmission.

**Table of Contents**

Table of Figures 4

1. Introduction

A. Introduction 5

B. Technical History 5

C. Technical Literature 6

D. Lifecycle of Similar Products . 10

2. Experimental Method

A. Engineering Requirements Development . 11

i) Customer Requirements . 11

**ii) (done)Environmental and Safety Concerns…………………………………**

**iii) (done)Legal, Ethical or Political Concerns…………………………………**

**iv) (done)Sustainability……………………………………………………..**

v) Constraints . 12

vi) Standards . 12

B. **(done)** Engineering Requirements. 13

C. **(done)** Level 1 and 2 Functional Decomposition 14

**D. (done) Subsystem Development…………………………………………………………….**

**3. SYSTEM INTEGRATION, TESTING, AND RESULTS…………………………………**

4. PROJECT MANAGEMENT

A. Project Work Breakdown Structure (WBS) . 17

B. **(done)**Project Schedule (Gantt Chart) . 18

C. **(done)**Project Budget . 19

4. Summary and Conclusion   
 **A. (done) Project After-Action Report…………………**

**B. (done)Future Work…………………………………**

List of References 20

**Appendix A: Final Schematics, Layout, Parts List**

**Appendix B: List of Data Sheets Used  
Appendix C: Code**

**Appendix D: Calculations and Simulations**

**Appendix E: Technical Papers**

**Appendix F: Engineering Requirements Development**

**Table of Figures**

1: Absorbance spectra of red and IR light…………………………………………………………………………………5

2: Finger probe used for pulse oximetry devices……………………………………………………………………...6

3: AFE4490 front end chip………………………………………………………………………………………………….…….7

4: Maxim Integrated’s design for front end LED driving system………………………………………………..8

5: Typical lifecycle of biomedical devices……………………..…………………………………………………………10

6: Hierarchy representation of marketing requirements…..…………………………………………………….11

7: Level 1 Functional Decomposition……………………………………………………………………………………….14

8: Level 2 Functional Decomposition……………………………………………………………………………………….15

9: Work Breakdown Structure for overall design of FPGA system……………………………………………17

10: Timeline for overall system design……………………………………………………………………………………..18

**TableS**

1: SPO2 processing algorithms………………………………………………………………………………………………….9

2: HR Processing methods…………………………………………………………………………………………………………9

3: Marketing requirements………………………………………………………………………………………………………13

4: Functional Decomposition Level 2……………………………………………………………………………………..…15

5: Cost analysis chart used for determining budget……………………………………………………………….…19

**INTRODUCTION**

**INTRODUCTION**

An FPGA-based pulse oximetry system will allow for researchers to maximize the amount and quality of data they receive from peripheral devices they are using to conduct tests. This device is intended for use in research and laboratory settings, as opposed to clinical settings. An existing system currently has a microcontroller bussing information to a computer for post-processing. However, in order to expand on this system to increase functionality and number of peripherals, the FPGA-based device will be designed. This will allow for parallel, real-time processing, with the ability to send large amounts of information to a cloud server as needed. Many FPGA’s are also low-power, small form factor devices that can be considered well suited to function as brain-machine interface chips.

**TECHNICAL HISTORY**

First developed in the early 1900’s, pulse oximetry devices have become known as ‘the fifth vital sign’ due to being incredibly useful in clinical settings for detecting both blood oxygen levels and heart rate in patients, especially those undergoing anesthesia.

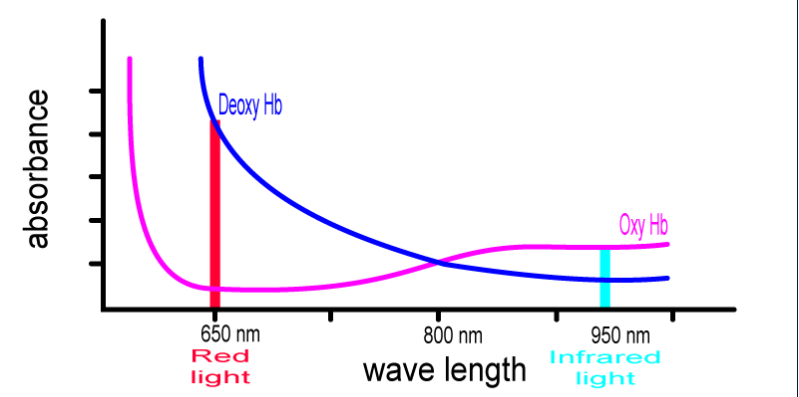
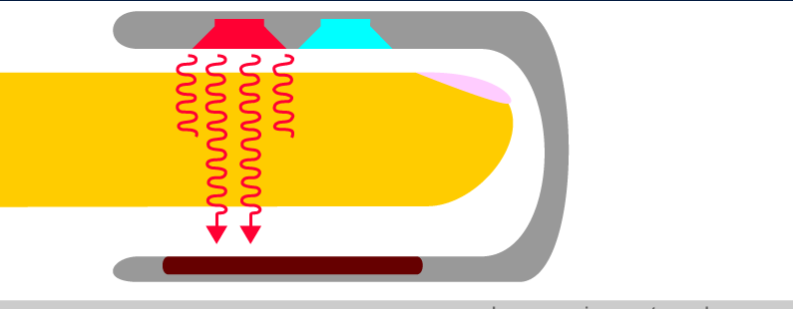
The first device to ever measure oxygen saturation levels was created by Karl Matthes in 1905. It was a two-wavelength oxygen-saturation meter that used red and green filters. This device was used on the ear to measure saturation. In the 1940’s Glenn Allan Millikan expanded on this concept by creating an oxygen saturation system that could be used to warn pilots when their blood oxygen levels were getting low during aerial combat. This device was also attached to the earlobe, and contained an incandescent light bulb, a set of red and green filters, and a selenium barrier level photocell. Millikan believed that the green light absorbance was independent of blood oxygen level, but this was later disproved, and the green light was replaced with infrared light.

Fig. 1: Absorbance spectra of red and infrared light [1]

In 1972, the foundation of what could be considered modern day pulse oximetry was created by bioengineers Takuo Aoyagi and Michio Kishi. Their device measured the ratio of red to infrared light being absorbed by pulsating components at a measuring site. It was this device that factored in that changes in arterial blood was causing ‘pulses’ in the generated signal, and that these changes could be used to measure arterial oxygen saturation without need of zero

calibration. When a pulse is not present, the signal would represent a baseline absorption[2].

Fig. 2: Finger probe used for pulse oximetry devices, with red and infrared LED’s emitting light through the finger to the brown-colored photodetector [1]

From 1974 onward, pulse oximetry devices became commercialized using fingertip probes with fiber optic cables. In the late 1970’s American manufacturers began creating their own specialized version of the pulse oximeter. Improvements in diode technology at this time also led to improved accuracy. These devices then became a common measurement device for clinical applications, especially anesthesiology [2].

**TECHNICAL LITERATURE**

Modern day pulse oximetry devices are broken down into three parts: the analog front end, the processing unit, and the processing algorithm. How the system displays results can also be considered.

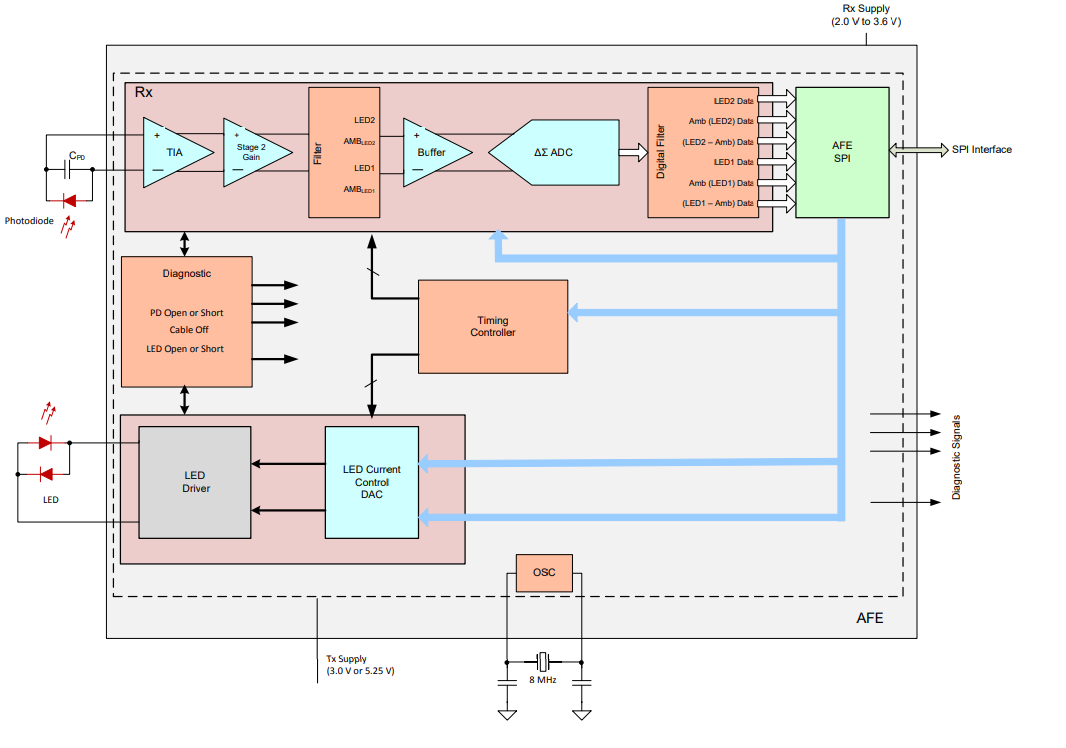
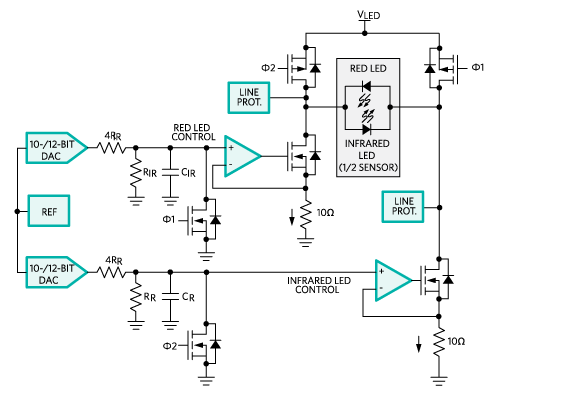
The analog front end is the part of the unit that typically involves a measuring probe, a photodetector, preamplifier, buffer, and DC component filter [3]. The AFE4490 functions as an analog front-end system completely encapsulated in a single chip. 

Fig. 3: AFE4490 Front end chip[4]

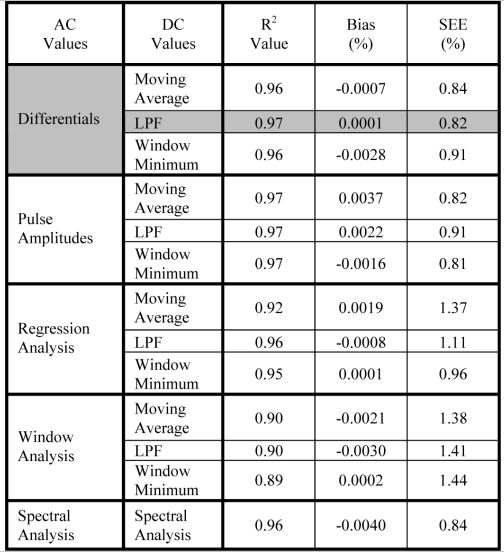
This device functions to drive the LEDs of the probe, receive light on the other end of the finger, convert the light into a current and then pass the current into the chip. The chip then runs through a transimpedance amplifier, active lowpass filter, and analog to digital converter. It therefore provides every role of the front of end system.

The processing unit is then introduced. Typical processing application units are microcontrollers. These devices function to pull data from the front end and process it. More specifically, the entire process from the measuring of data to displaying and recording of results occurs on the microcontroller [3]. How a microcontroller is selected depends on the degree of analog functionality and precision needed [4]. If the data going into the microcontroller is already digital, then low cost and low power microcontrollers such as 32-bit ARM microcontrollers suffice [4].

 Fig. 4: Maxim Integrated’s design for front end LED driving system [5]

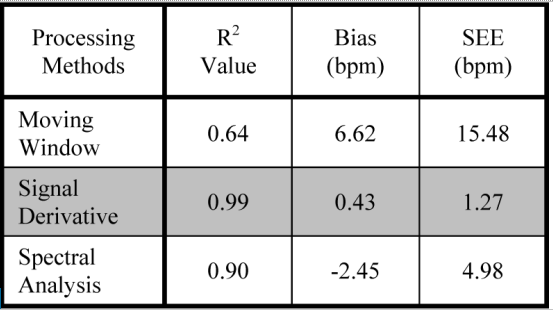
The last part of the implementation is the processing algorithm. Several algorithms for processing the data exist, each with their own benefits and drawbacks concerning computation time and processing accuracy. All processing algorithms must find AC and DC components of signals, but how it finds these components can be highly variable. To generalize the processing algorithms, they can either be performed by moving averages and sorting algorithms, or by spectral analysis.

Table 1: Table of SPO2 processing algorithms



The DC Values column displays what technique is used to find the DC component of the signal, with the other columns displaying how accurate the computation is [6].

Table 2: Table of Heart Rate Computation Methods [6]



For this project, the pulse oximeter being designed is to function as a proof of concept for a larger embedded brain machine interface system. Because it is part of a larger system, the ability for the pulse oximeter to perform parallel, real time computations is of high importance. Therefore, instead of a typical microcontroller processing unit, an FPGA is to be used. Since an FPGA is being used, a front-end device that already performs analog to digital conversion is desired, and a processing algorithm using spectral analysis is also preferred. This covers the design aspects for the pulse oximetry device that is to be implemented, and why it deviates from what could be considered typical design aspects of commercial pulse oximetry devices.

**LIFECYCLE OF SIMILAR PRODUCTS**

Biomedical devices in general undergo a systematic life-cycle process with emphasis on risk analysis, validation, and verification. This is due to there being large negative consequences, up to and including patient death, if the designed device does not work as intended. Regulatory and legal requirements must also be applied in each phase [7].

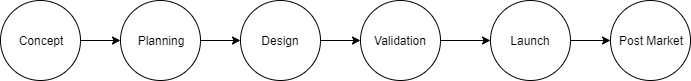


Fig. 5: Typical Lifecycle of biomedical devices

The phases involved with the product lifecycle can be broken down as seen in figure 7. Each phase can then be further explained as follows:

Concept: Exploring ideas that solve a medical problem while taking into consideration factors such as cost and safety.

Planning: Collecting user needs, creating engineering requirements, and design of basic prototyping systems.

Design: Commencement of product design and documentation.

Validation: Commencement of clinical validation to ensure safety and effectiveness. The product is submitted to regulatory organizations for review.

Launch: Device is sold to health care providers, hospitals, ect.

Post Market: Monitoring of sold devices to ensure product is functioning as intended, review of reports from clinicians regarding malfunctions or general feedback.

**EXPERIMENTAL METHOD**

**ENGINEERING REQUIREMENTS DEVELOPMENT**

**-Customer Requirements**

As per customer request, it has been decided that the fundamental design aspects needed for this design include:

1. Using the FPGA to do real-time, parallel processing.
2. Implementing WIFI communication that allows for large amounts of data to be sent remotely from the FPGA to a server.
3. A PCB design that meets a predefined embedded design for a brain-machine interface along with a PCB design for basic, noninvasive human use.
4. Use of AFE4490 front end chip in design.
5. Proper/extensive calibration and testing to occur to ensure accurate measurements and computations.

Less important, but still possible design criteria include:

1. Remote FPGA control, two-way communication from FPGA to remote device via cloud.
2. Use of visual display to display signal in real time.
3. On-chip statistical data analysis of computed heart rate and blood oxygen values.

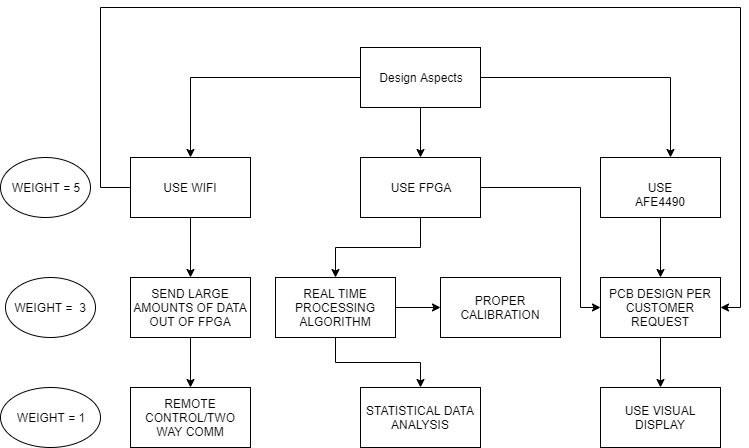


Fig. 6: Hierarchy Representation of Marketing Requirements (Importance from top to bottom)

**-Environmental and Safety Concerns**

This project complies with all applicable environmental, health, and safety laws and regulations included in the IEEE code of ethics. It poses no potential risk to harming the environment or harming organisms.

**-Legal, Ethical, or Political Concerns**

This project used Verilog code (SPI module and UART module) written by Russell Merrick, and licensed under the MIT license. The license allows for permissive and free use of his code, given that copyright and license notices are preserved. This project also used Verilog code (Square root Module) written by Schuyler Eldridge, and licensed under the Apache License 2.0. This is a permissive license allowing for free distribution and use. This project also used an unlicensed FFT module from Riddhi Padariya, a recent graduate student from Penn State. The module was used with her permission. The PCB design for this project was done by Charlie Wischner, and all credit goes to him.

No ethical or political concerns arose or currently exist in the design and implementation of this project.

**-Sustainability**

This project does not use energy or resources in a way that compromises the environment or future generations.

**-Constraints**

The following system-level, cost, and time constraints are issued for the design:

System constraints:

* Must interface with AFE4490 Front-End chip
* Must have data processing algorithm on the board (as opposed to post-processing)
* Must use WIFI communication to the send data to server
* Accuracy of algorithm computation must meet clinical standards
* Design must be on FPGA, primarily Intel/Altera due to Penn State funding

Cost constraints:

* The costs to execute this project are minimal since the design focuses around programming a single FPGA, which is a low-cost device. Prototyping devices and PCB design will have a cost, but funding for the project through Penn State’s REU program should cover it.

Time constraints:

* The project must be completed in the Capstone Design Project timeframe.

**-Standards**

The following standards will be implemented into the design:

Interoperability: Serial Peripheral Interface (SPI), WIFI, Universal Asynchronous Receiver-Transmitter (UART), AT commands (modem communication)

Consumer Safety: May investigate a clinical standard for accuracy of measurements

SPI communication is used for interacting between the FPGA and the AFE4490 chip. UART communication is used for sending data to the WIFI module. WIFI communication is used for sending data from the WIFI module to a server wirelessly. AT commands are the commands sent on the UART line in order to communicate information from the FPGA to the WIFI module.

**ENGINEERING REQUIREMENTS**

The following table takes the previously stated requirements and shows how these requirements will be met through engineering and design.

Table 3: Marketing requirements that lead to engineering requirements and the justification of design choice

|  |  |  |
| --- | --- | --- |
| **Marketing Requirements** | **Engineering Requirements** | **Justification** |
| 1, 9, 11, 13, 16, 17 | Use Intel Max 10 FPGA | FPGA’s are programmable devices capable of parallel, real time processing; MAX 10 series chip has flash memory to keep designed system non-volatile |
| 2, 9, 11, 14 | Use ESP8266 WIFI module | Simple WIFI module that is highly configurable for wireless data transmission |
| 3, 16, 17 | Master board PCB design, general PCB design | Master board design for use in embedded brain-machine interface system.  General PCB design for human use |
| 4, 9, 11, 12 | Use AFE4490 mixed signal chip | Chip specifically designed by Texas instruments for pulse-oximetry |
| 5, 10, 15 | Extensive calibration process involving simulation and testing with MATLAB | Processed/computed data must be highly accurate, so a systematic calibration process must be implemented |
| 6, 9, 11, 14 | Modification of web server/wifi module system to accommodate for two-way/remote system control | Allows for the functionality of the FPGA to be controlled wirelessly through implemented WIFI system. This allows for control of sampling rate, data streaming time, resets, ect. |
| 7, 17, 18 | Thin-film-transistor (TFT) display driven by FPGA using VGA interface | System capable of displaying the pulse signal in real time |
| Customer Requirements:   1. Using the FPGA to do real-time, parallel processing. 2. Implementing WIFI communication that allows for large amounts of data to be sent remotely from the FPGA to a server. 3. A PCB design that meets a predefined embedded design for a brain-machine interface along with a PCB design for basic, noninvasive human use. 4. Use of AFE4490 front end chip in design. 5. Proper/extensive calibration and testing to occur to ensure accurate measurements and computations. 6. Remote FPGA control, two-way communication from FPGA to remote device via cloud. 7. Use of visual display to display signal in real time. 8. On-chip statistical data analysis of computed heart rate and blood oxygen values.   Standards:   1. Interoperability: Serial Peripheral Interface (SPI), WIFI, Universal Asynchronous Receiver-Transmitter (UART), AT commands (modem communication) 2. Consumer Safety: May investigate a clinical standard for accuracy of measurements 3. SPI communication is used for interacting between the FPGA and the AFE4490 chip. UART communication is used for sending data to the WIFI module. WIFI communication is used for sending data from the WIFI module to a server wirelessly. AT commands are the commands sent on the UART line in order to communicate information from the FPGA to the WIFI module.   Constraints:   1. Must interface with AFE4490 Front-End chip 2. Must have data processing algorithm on the board (as opposed to post-processing) 3. Must use WIFI communication to the send data to server 4. Accuracy of algorithm computation must meet clinical standards 5. Design must be on FPGA, primarily Intel/Altera due to Penn State funding 6. The costs to execute this project are minimal since the design focuses around programming a single FPGA, which is a low-cost device. Prototyping devices and PCB design will have a cost, but funding for the project through Penn State’s REU program should cover it. 7. The project must be completed in the Capstone Design Project timeframe. | | | |

**LEVEL 1 AND 2 FUNCTIONAL DECOMPOSITION**

The following decompositions explain the designed system functionality on varying levels of abstraction.

Level 1 decomposition:

* Data acquisition system
* Data processing system
* Data output system

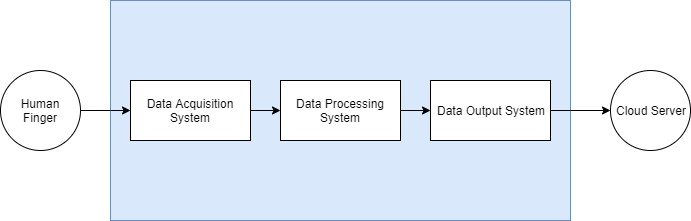


Fig. 7: Level 1 Functional Decomposition

Level 2 decomposition:

Data acquisition:

* Probe connection to finger
* Filter/process raw samples
* Analog to digital conversion of samples

Data processing:

* Process digital samples
* Compute Heart Rate
* Compute Blood Oxygen

Data Output:

* Format data so that it can be wireless transmitted
* Send computed data and digital samples out to cloud server

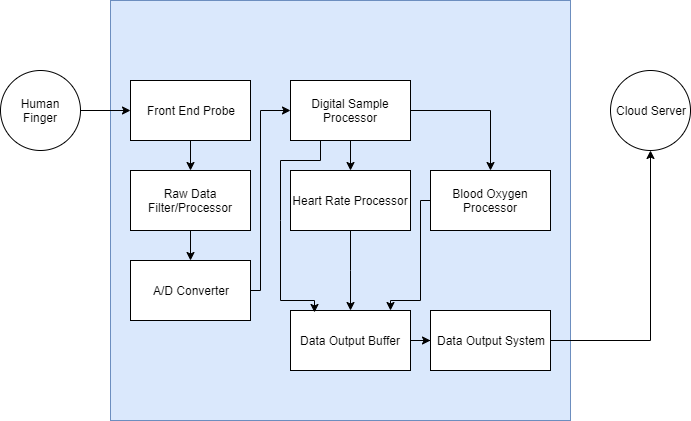


Fig. 8: Level 2 Functional Decomposition

The following table further explains the I/O and functionality of the modules encompassed in the level 2 decomposition

Table 4: Further Analysis of Functional Decomposition Level 2

|  |  |
| --- | --- |
| **Module** | **Front End Probe** |
| Inputs | Human finger |
| Outputs | Raw Data processor |
| Function | Placed on human finger to receive light values proportional to blood oxygen saturation, photodiode converts light to current |

|  |  |
| --- | --- |
| **Module** | **Raw Data Processor** |
| Inputs | Front end probe |
| Outputs | A/D Converter |
| Function | Receive current, convert to voltage, low pass filtering |

|  |  |
| --- | --- |
| **Module** | **A/D Converter** |
| Inputs | Raw Data Processor |
| Outputs | Digital Sample Processor |
| Function | Convert analog values to digital samples |

Table 4 continued

|  |  |
| --- | --- |
| **Module** | **Digital Sample Processor** |
| Inputs | A/D Converter |
| Outputs | Heart Rate Processor, Blood Oxygen Processor, Data Output Buffer |
| Function | System to store samples from A/D converter and prepare them for processing |
| **Module** | **Heart Rate Processor** |
| Inputs | Digital Sample Processor |
| Outputs | Data Output Buffer |
| Function | Perform algorithm to compute heart rate |

|  |  |
| --- | --- |
| **Module** | **Blood Oxygen Processor** |
| Inputs | Digital Sample Processor |
| Outputs | Data Output Buffer |
| Function | Perform algorithm to compute blood oxygen |

|  |  |
| --- | --- |
| **Module** | **Data Output Buffer** |
| Inputs | Digital Sample Processor, Heart Rate Processor, Blood Oxygen Processor |
| Outputs | Data Output System |
| Function | Receive and store both digital samples and computed values, format information to be sent out of system wirelessly |

|  |  |
| --- | --- |
| **Module** | **Data Output System** |
| Inputs | Data output Buffer |
| Outputs | Cloud Server |
| Function | System that connects to a cloud server and sends samples and computed information to it |

**SUBSYSTEM DEVELOPMENT**

**DATA ACQUISITION SYSTEM** – The data acquisition system is considered to include the front-end probe connecting to the AFE4490 as well as the AFE4490 connecting to the FPGA. The front-end probe is a Nellcor DS-100A and uses a DB9-F connector to route signals into the AFE4490. The probe has a red and an IR LED that are both driven by the AFE4490. On the other end of the probe is a photodetector, which generates a current that is proportional to the amount of light it is receiving from either of the two LEDs, and sends it into the AFE4490.

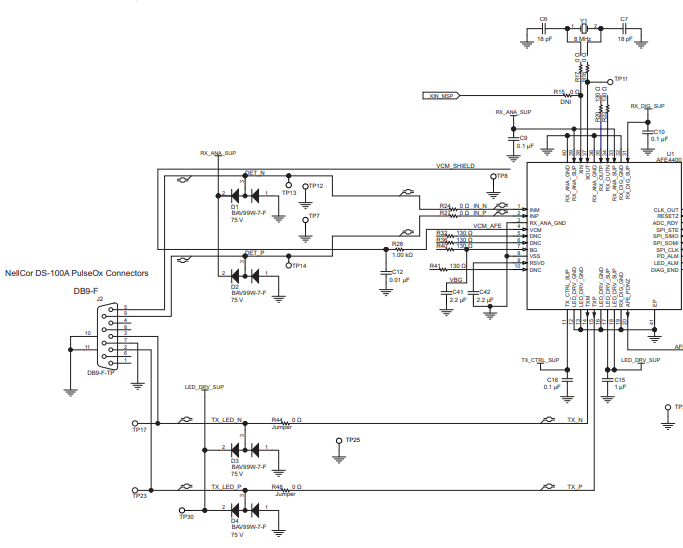


Figure X: Example schematic showing interconnect between front end probe and AFE4490

The AFE4490 receives the current from the finger probe and passes it through a transimpedance amplifier in order to convert to a voltage, an active lowpass filter in order to remove noise, and an analog-to-digital converter in order to create a digital sample. It stores the digital sample in registers that can be accessed through SPI communication. The AFE4490 controls the rate at which the LEDs oscillate back and forth, and thus controls the sampling rate of the information passed into it.

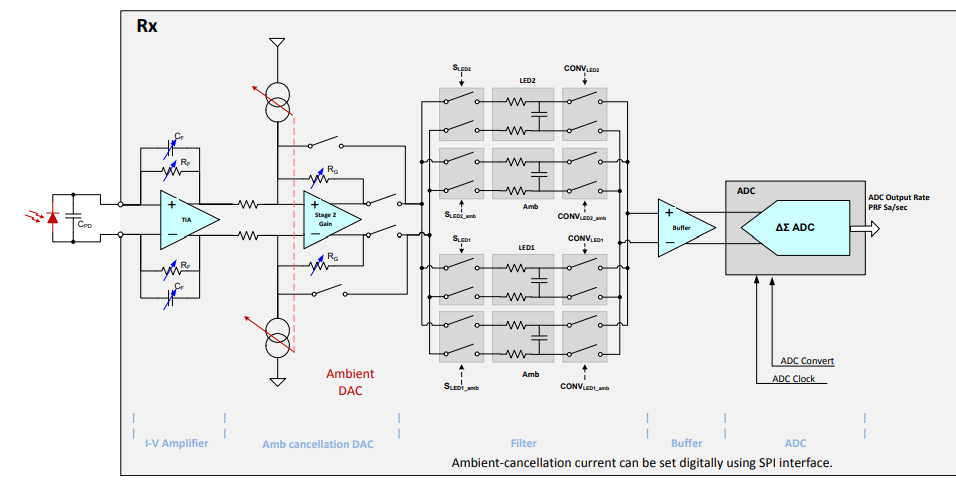


Figure X: Schematic showing the signal path through the AFE4490

In order to drive the processes of the AFE4490 and receive digital samples from it, Verilog code must be written and synthesized onto the FPGA in order to send and receive information using an SPI communication protocol to registers in the AFE4490.

A flowchart of the processes that must occur in order for the AFE4490 to function was created (figure x). From that flowchart, a system of Verilog modules was designed with a state machine implementing control and determining what state from the flowchart the system was in (figure x). The system involved a FSM, address selector, read RAM, write RAM, and SPI module. The process of the FPGA system was as follows:

1. FMS sends control signals to all other modules based on what state it is in.
2. Given control information, address selector sends out addressing information to RAM modules.
3. Given control and addressing information, the write RAM sends addressing information and data to SPI module (if in a write state)
4. Given control, data, and addressing information, the SPI module formats address and data for specified SPI format (SPIMODE 0, 32 bit words, 8 bytes per send, uS delays between bytes, ect.)
5. Data received from AFE4490 to SPI module would be passed to read RAM (if in a read state)

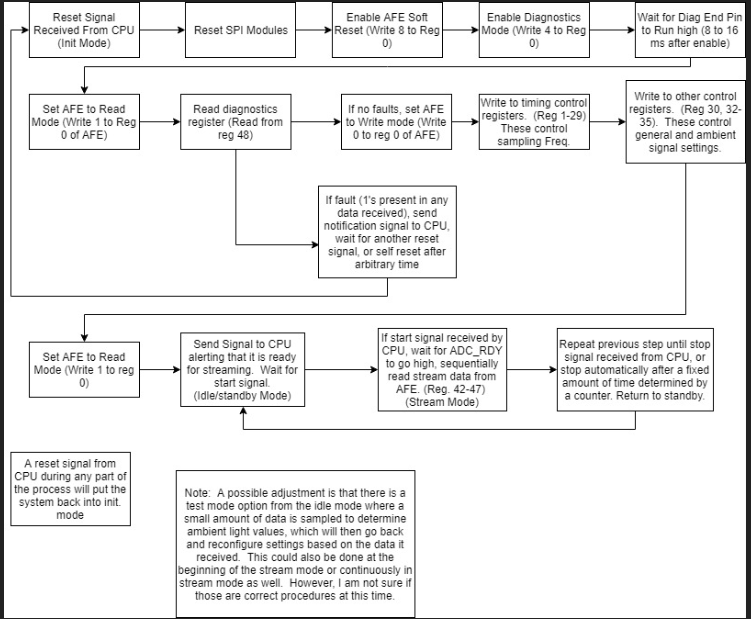


Figure X: Early flowchart design describing how the FPGA would configure and then stream from the AFE4490

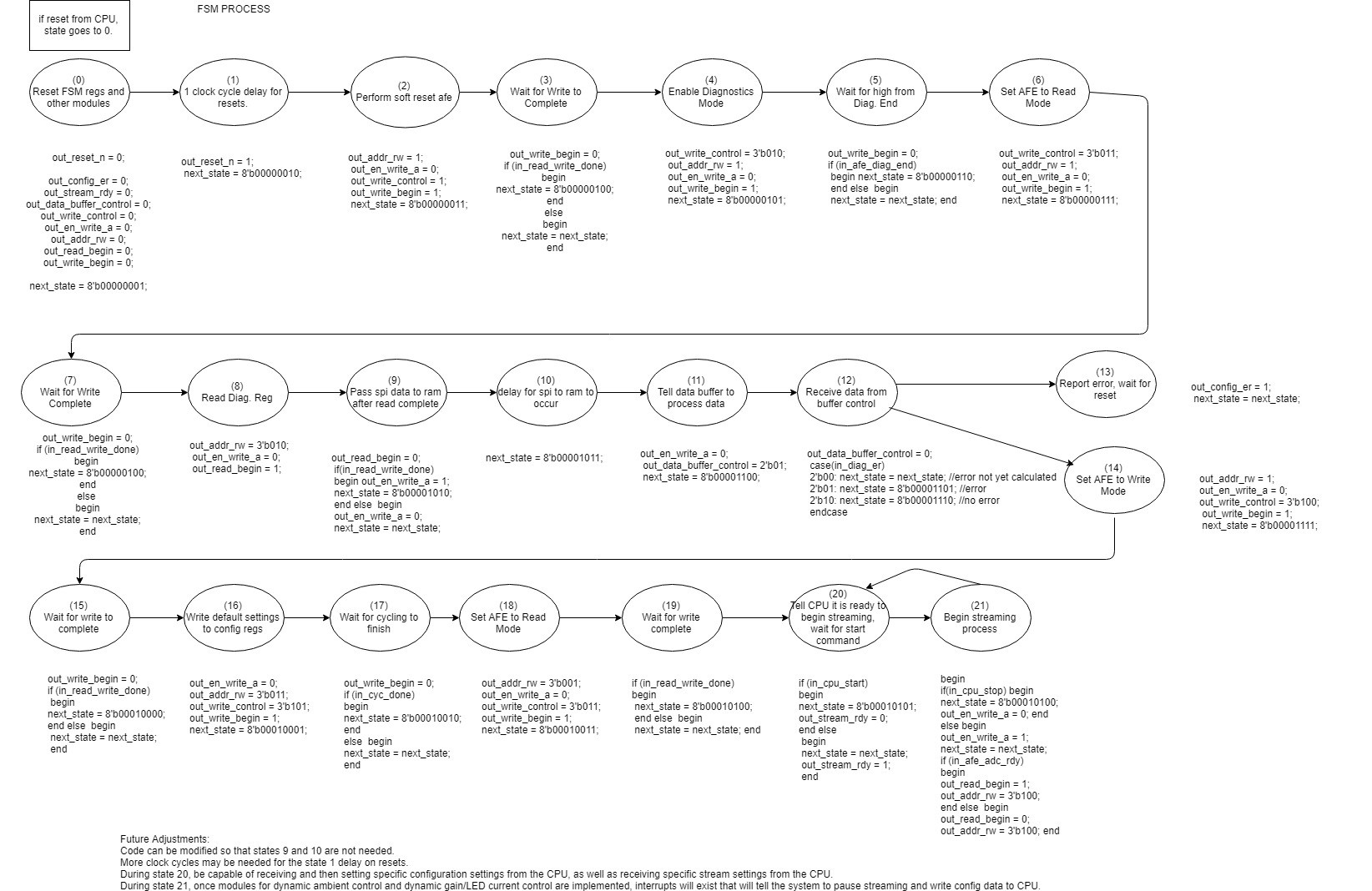


Figure X: Early flowchart describing FSM output signals associated with each state

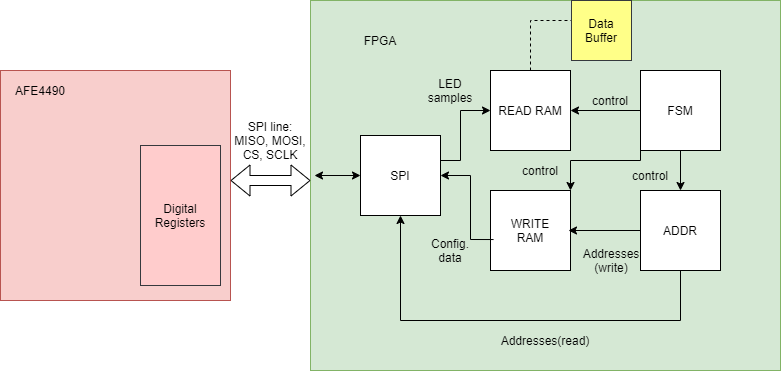


Figure X: Block diagram of interconnect system involved with the FPGA communicating to the AFE4490, as well the module interconnect

**DATA PROCESSING SYSTEM** – After raw samples are received by the FPGA, they must be processed in order to determine SPO2 and HR in real time. The data processing system is considered the subsystem within the FPGA responsible for computing these values. As stated in the technical literature portion of the report, there are multiple methods of computing heart rate and blood oxygen. Multiple methods were attempted in the design of the real-time computational system, and all of them used the following equation for calculating SPO2:

Where

This means that each signal generated by a group of samples from the red LED and IR LED contains an AC component and a DC component whose value is used to compute SPO2. Heart rate is then extracted by taking the AC component and multiplying by 60 in order to convert from Hz to BPM.

Spectral analysis using an FFT appeared to be an efficient method, since FFT’s are common implementations in an FPGA, and can yield accurate results when finding AC and DC components. This was verified through a MATLAB simulation using actual samples extracted from the AFE4490. The simulation process is explained in detail in the system integration section. A system of Verilog modules was written in order to implement the data processing algorithm with an FFT. In order to maintain clinical accuracy within the computation, a resolution of around .04 in the FFT needed to be achieved, where

The AFE4490 was sampling at 500 sps, which means that a sample size of 12500 samples would be needed in order to achieve .04 resolution. A sampling size this large would be too resource and computation heavy for the FPGA. This size can be scaled down to a more standard 1024 size if the sampling rate is also scaled down from 500 to 40 sps. However, 25.6 seconds of data would then need to accumulate before being passed into an FFT for a new computation to be performed. In order to minimize time between accumulations, a circular buffer was designed and implemented. This allowed for a new computation to occur whenever a new sample was passed into the circular buffer (once every 25ms). Data is output from the FFT in real and complex values. Only magnitude is relevant for this application, so the real and complex values must be squared, added, and square rooted. Since negative frequencies are not relevant, only the first 512 of 1024 samples of the output are processed. Since the sampling rate is 40 samples per second, the 512 samples represent values from 0 to 20 Hz, incrementing at around .04 Hz per sample. A post data buffer system functions to perform the magnitude calculations, and data is then stored in RAM, where a sorting algorithm extracts the AC and DC components. This system is implemented twice, in parallel, since one is needed for the red and IR LEDs. These components are then sent to a final computation module, where the ratios of the red and IR LEDs are computed, multiplied by 25, and subtracted from 110.

The data flow was as follows:

1. Data passes from read RAM to data buffer.
2. Data buffer passes samples to circular buffer.
3. Circular buffer performs downsample and accumulates data into RAM.
4. Circular buffer passes data into FFT
5. FFT performs computation and outputs values into post data buffer.
6. Post data buffer groups first 512 samples, runs through square, add, square root system.
7. Sorting algorithm extracts AC and DC component, passed to final computation module.
8. Final computation module takes ratios of components and computes SPO2.

The sorting algorithm also sent the address associated with the AC component to a LUT module, where a heart rate would be output.

This implementation yielded mixed results, and it was determined that some part of the design was at fault either with incorrect implementation or with a bug. A further explanation is included in the system integration section.

An alternative method that was implemented was a custom time domain peak detection algorithm. This algorithm found peaks and troughs in a group of samples. It then calculated the DC component by finding the average between a peak and trough and the AC component by finding the magnitude of the peak to trough value. Frequency and subsequently heart rate are found by calculating the number of samples between a peak and trough and factoring in sampling rate. For a sampling rate of 500 sps, heart rate would be

**DATA OUTPUT SYSTEM –** Once an SPO2 and HR computation is made, those values must be sent out of the FPGA to a cloud storage system. The data output system is considered the system of modules within the FPGA responsible for interacting with an ESP8266, an ESP8266 WIFI chip, and a Raspberry PI that emulates a cloud storage device. Within the FPGA, Processed data is passed to a FIFO which acts to perform clock domain crossing to a WIFI and UART module. The UART module operates at half the clock frequency of the rest of the FPGA (from 50MHz to 25Mhz) and this is because it must further divide down to a baud rate of 115200bps to drive the UART system, something that isn’t able to be achieved with 50Mhz, given the design of the UART module. Within the WIFI module, a series of AT commands are generated that pass to the UART and to drive the firmware on the ESP8266. The series of commands are as follows:

1. System Reset (AT+RST)
2. Set WIFI module as station (AT+CWMODE =1)
3. Connect to Server (AT+CWJAP = “Server Name”)
4. Create Single Connection (AT+CIPMUX = 0)
5. Create TCP connection using specified IP address (AT+CIPSTART=\"TCP\",\"192.168.42.1\",80")
6. Specify data width for HTTP GET command (AT+CIPSEND=68)
7. Write to PHP file via GET command ("GET /index.php?heart=56&bldox=23 HTTP/1.1\r\nHost: 192.168.42.1\r\n\r\n")

The WIFI module uses these commands to connect to the Raspberry PI. Using DHCP and HOSTADP packages downloaded and configured through the Linux Kernel, the Raspberry PI can be configured to function as a wireless access point, assigning IP addresses to systems that connect to it. An Apache web server was implemented onto the device, and a PHP file and text file were created on the server. When a device connects to the server, it can either choose to connect to the text file or the php file. When connecting to the php file, a device can send data to it. The php file was coded so that it takes this data and places it in the text file. A device that connects to the text file is able to read the information in it. Apply this concept, the WIFI module in the pulse oximetry is system connects to the PHP and sends data to it. Then, a remote computer can simultaneously connect to the server, access the text file, and read what the WIFI module is sending.

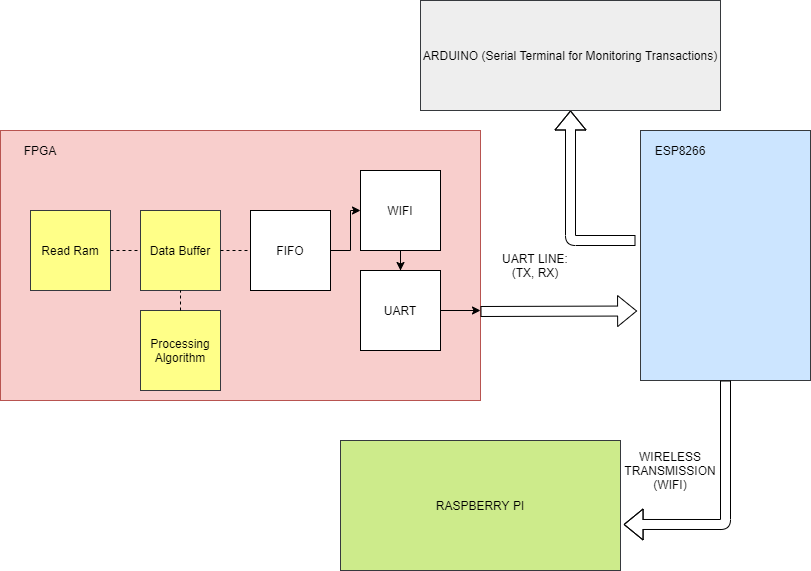


Figure X: Wireless data transmission system

The version of the ESP8266 used for this project comes with open-source, preloaded firmware called NodeMCU. When flashed, this firmware can receive AT commands from the chip’s uart line and use them to connect and send data to other devices.

**SYSTEM INTEGRATION, TESTING, AND RESULTS**

**SIMULATIONS AND TESTING –** There were three types of simulations/tests used in this project: Verilog Simulation/Testbenches, raw data testing with MATLAB, and system integration testing using ICs on breakout boards.

**Verilog:** In order to verify the functionality of a Verilog module, simulations in Quartus and Modelsim were used. Test stimulus is applied to the input of the simulation, and it can be seen when output signals trigger high or low. Some examples of testbenches used in this project can be found in appendix XYZ.

**MATLAB:** A verification process was performed in order to ensure that the AFE4490 was being driven properly, and that raw data being transferred from the AFE to FPGA was correct. The verification process involved extracting raw data and running it through a MATLAB simulation using an FFT within the MATLAB function library. Through inspection, heart rate and blood oxygen could be calculated. This simulation would function to both verify data acquisition from the AFE4490, as well as verify the functionality and accuracy the data processing algorithm intended to be implemented in the FPGA.

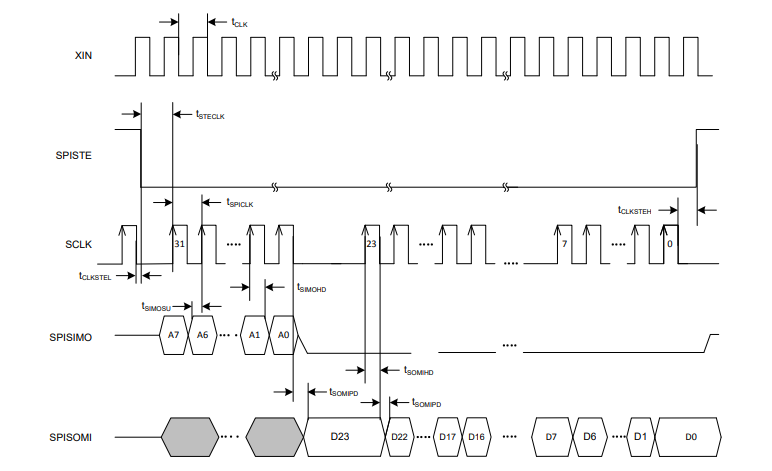


Figure 1: Timing diagram for reading data from the AFE4490. The FPGA handles driving the SPISTE, SCLK, and SPISIMO lines, while the AFE4490 outputs the SPISOMI line.

A Tektronix Logic Analyzer was applied to all four of the SPI lines connecting the AFE4490 to the FPGA. 10 seconds of data was recorded and saved into a CSV file within a memory card on the analyzer. This information was then transferred to a computer.

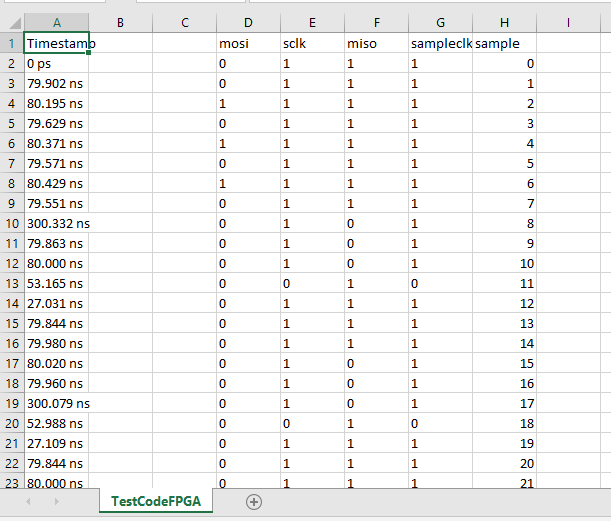


Figure 2: CSV File showing the recording of first 23 samples from the scope, opened in Excel. The system should trigger on every sclk high (column G), however, it can be seen that there are some errors with triggering (row 20 has a 0 and must be removed).

The file was imported into MATLAB where each column became a vector/variable to be manipulated. The column of data of interest is column F, the MISO line. After code was written to remove the errors with triggering, code was written to run through the values of the MISO line, throw out values associated with addressing, and keep values associated with output data. Bits were grouped in 24 bit words, and LED1 words were separated from LED2 words. The MISO line is still activating when the MOSI line is targeting the address, but the information it is sending has no value, only the 24 bits sent after the address requested.

The samples were then converted from binary to decimal values, thus representing the actual voltage that the 22 bit values were mapped to. The simulation was eventually ran using both the raw values and voltage-equivalent values since in the FPGA, everything will obviously still be in bits.

At this point, the samples can be plotted in the time domain for analysis.

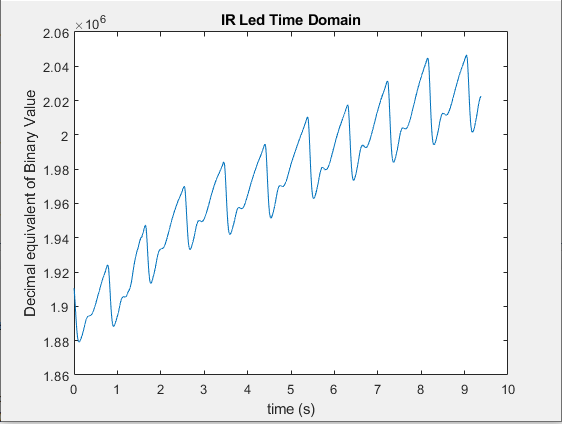
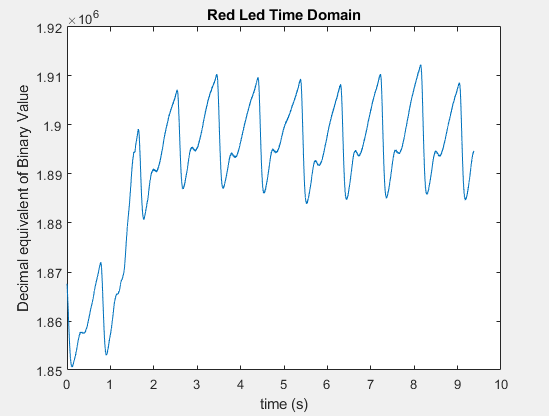


Figure 5: Time domain plot of samples in MATLAB.

Analysis of figure 5 shows that there is potentially some noise for the first 2 seconds of data collection, and the DC component for the IR LED appears to be variable as well. The simulation of the data processing algorithm was implemented regardless of this.

The data processing algorithm implements the following steps:

1. Use FFT to convert to frequency domain for both LED signals.
2. Extract AC and DC components from each.
3. The Frequency bin associated with the AC component of either FFT is the heart rate / 60.
4. SP02 = 110 – 25 \* (ACred / DCred) / (ACir / DCir)

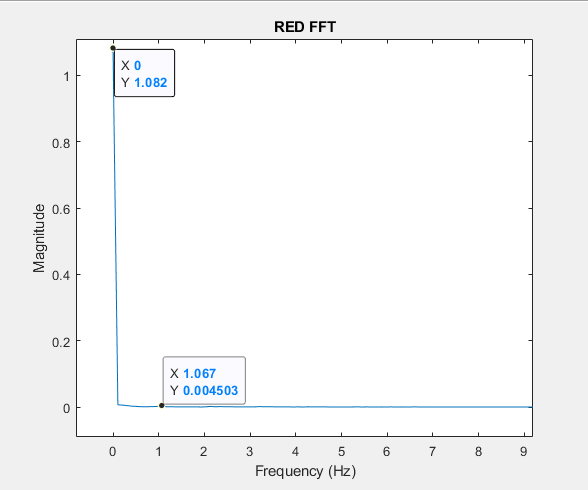
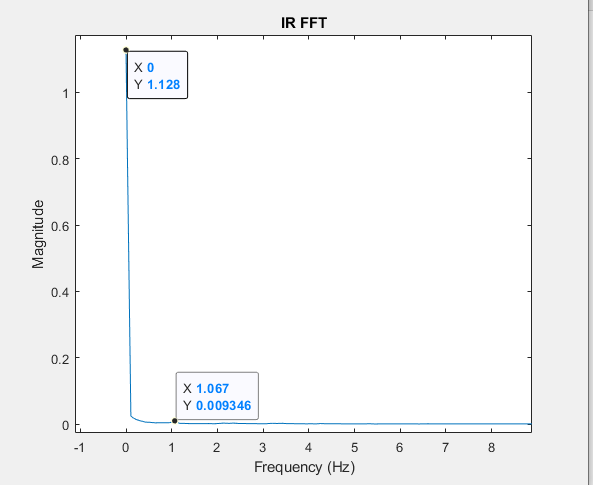


Figure 7: ACred = .004503, DCred = 1.082, ACir = .009346, DCir = 1.128 Frequency = 1.067 Hz

From the data of figure 7, and following the algorithm stated previously, the heart rate can be determined as 64.028 beats per minute. SPO2 can be determined as 97.442%. Given that this was data extracted from my finger, these values can be considered valid and acceptable, even given that there appears to be noise in the time domain. The final data results prove that both the raw data being driven from the AFE into the FPGA is valid, and that the intended algorithm to be implemented into the FPGA should work accurately and as intended. MATLAB code used for this simulation can be found in appendix XYZ.

**Breakout Boards:** In order to verify the overall functionality of each IC and if they communicated with each other, breakout boards were used. A breakout board existed for the AFE4490, Intel MAX 10, and ESP8266 WIFI chips. The Raspberry Pi was also on its own breakout board. After each system was developed, test data was passed from the finger probe through the entire system and out to the text file in the server on the Raspberry Pi. As the data processing system developed, real time verification of the processed data was analyzed through this text file.

**SYSTEM INTEGRATION**

System integration can be broken down into two domains: Verilog system integration within the FPGA and IC integration between the breakout boards. The latter has been described in detail both in the subsystem design section and breakout board subsection of simulation/testing in this report. For the former, the system hierarchy for the FPGA breaks down into its own sections based on each subsystem, as seen in figure XYZ.

Figure X: System hierarchy and categorization of modules in FPGA: Red-AFE4490 Interface, Green-Data Processing System, Purple-WIFI Interface

Each category had to be integrated in with others, and each category had to be designed in a way that allowed for simple changes to be made if needed. For example, multiple data processing algorithms were attempted, so all modules in green in figure XX had to be quickly and easily swapped in and out.

**RESULTS** (desing and pcb)

The system interface between the FPGA and AFE4490 was designed first. This subsystem was design and integrated successfully. The there were multiple attempts through the entirety of the project to successfully design the data processing subsystem, but it repeatedly failed to output successful results. However, it appeared to be integrated into the rest of the system without issue or error. The data output system with the FPGA-WIFI-Raspberry Pi interface was successfully implemented and integrated. However, data transmission is not at an optimal rate due to how the system was design.

The PCB design work was done by Charlie Wirth. Charlie’s goal was to implement each of the 3 ICs needed for this design as their own separate breakout board. Then he would interface them and, if working properly, would do a final PCB design with all three IC’s on one PCB. Charlie has successfully implemented the PCB design for the AFE4490, arguably the most complex of the three IC boards to design. The following figures show his design work.

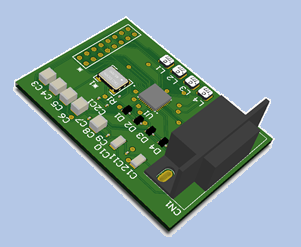


Figure 2: Early Design of AFE4490, without coin slot battery implementation

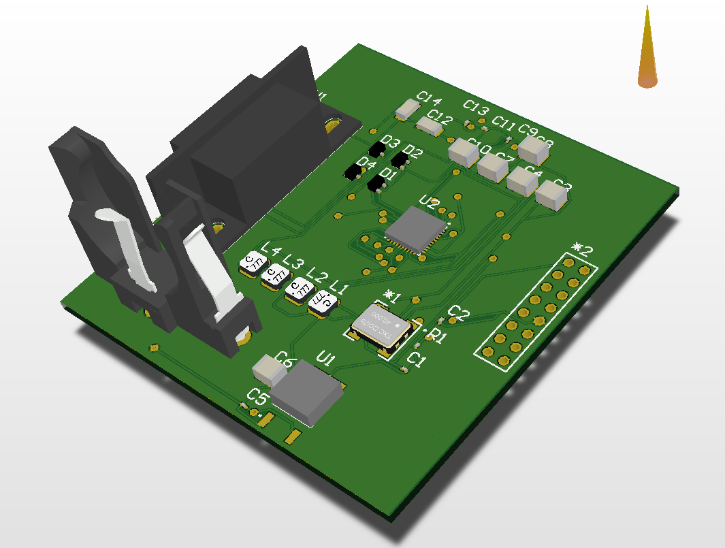


Figure 3: New Design with coin slot battery holder.

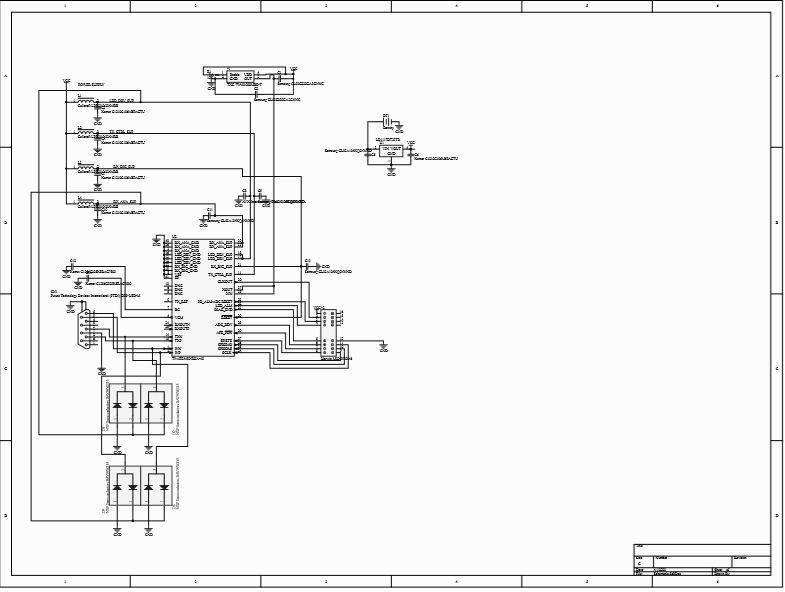


Figure 4: Schematic of AFE4490 PCB design

**PROJECT WORK BREAKDOWN STRUCTURE (WBS)**

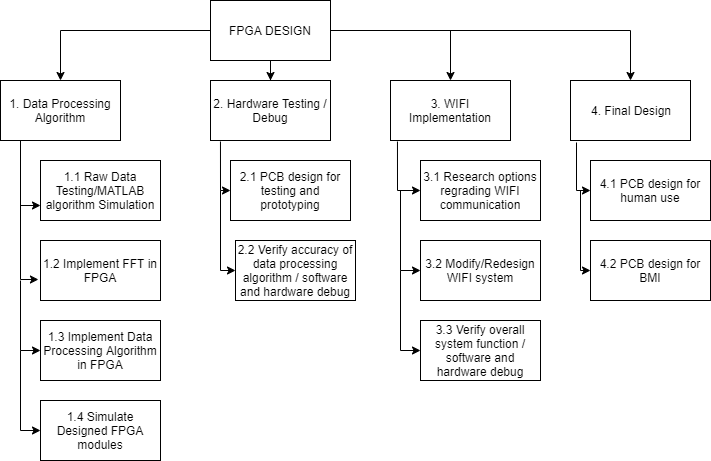
All tasks are assigned to Matthew Capuano, with potential assistance in PCB design from Charlie Wirth and Steve Grosse. Figure 9 elaborates on the processes needed for design completion.

Fig. 9: Work Breakdown Structure for overall design of FPGA system

1 – Finish algorithm simulation using raw data in MATLAB

2 – Implement FFT algorithm in FPGA and modify existing architecture as needed

3 – Implement data processing algorithm from FFT

4 – Perform simulation of designed modules in Quartus

5 – Design PCB with AFE4490 and FPGA for testing purposes

6 - Verify accuracy of heart rate and spO2 computation using PCB

7 – Research possible options regarding improvements to WIFI implementation

8 – Modify or redesign existing WIFI system

9 – Conduct overall performance testing

10 – Final PCB design for human purposes

11 – Final PCB design as a brain-machine interface

**PROJECT SCHEDULE (GANNT CHART)**

Figure 10 provides a breakdown of when the project tasks are to be implemented. This timeline will ensure that all tasks are completed within the timeframe provided in the capstone project.

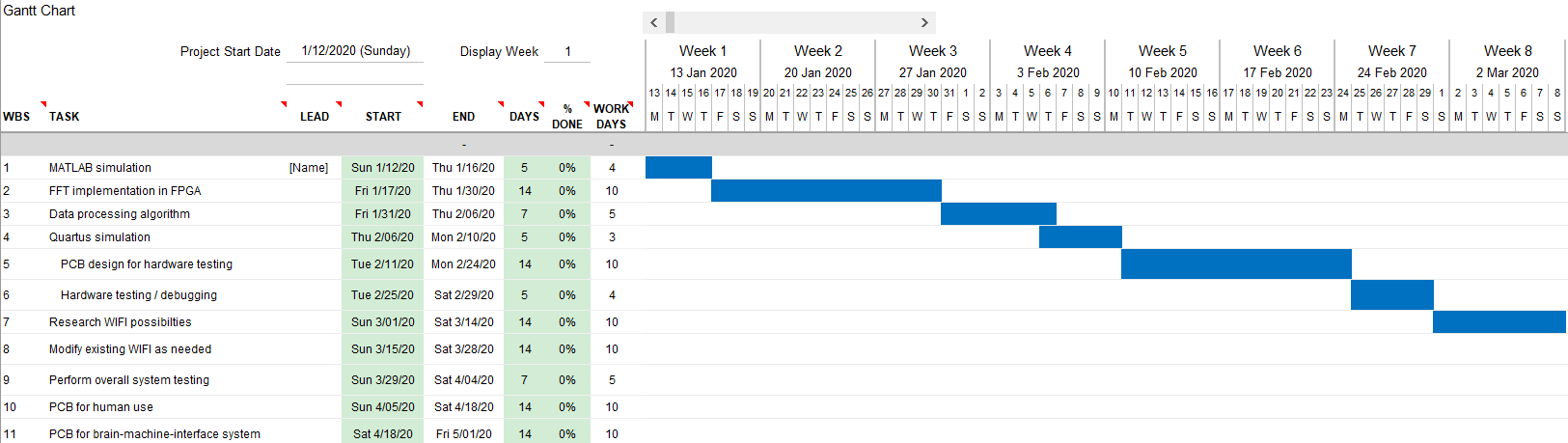
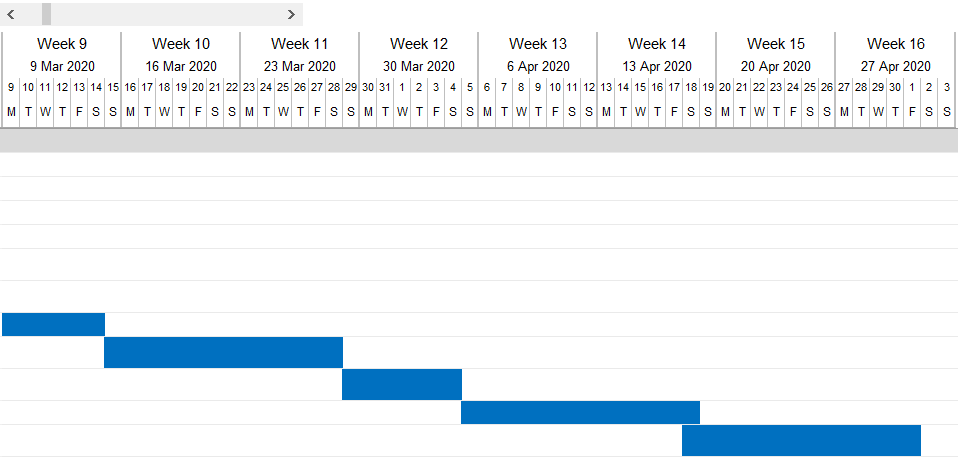


Figure 10: Timeline for overall system design

Timeline of Spring 2020: 13th of January (week 1) to 1st of May (week 16)

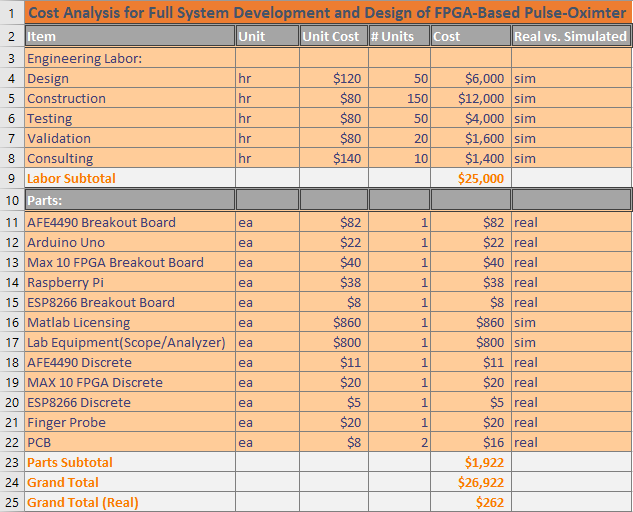
1. MATLAB simulation – week 1
2. FFT implementation in FPGA – week 1-2-3
3. Data processing algorithm – week 3-4
4. Quartus simulation – week 4-5
5. PCB design for hardware testing – week 5-6
6. Hardware testing/debugging – week 7
7. Research WIFI possibilities – week 8-9
8. Modify existing WIFI as needed – week 10-11
9. Perform overall system testing – week 12
10. PCB for human use – week 13-14
11. PCB for BMI system – week 15-16

Items 1, 2, 3, 4, and 5 were completed on time. The hardware testing and debugging phase (item 6) caused extensive problems and was extended for multiple weeks. Alternate implementations were considered and designed as well. The research for WIFI adjustments began regardless of the fact that the data processing system was not working, but actual changes were not made since there was no valid data to transfer. Charlie Wischner worked on item 10 throughout the entire course of the project. Ultimately, items 6, 8, 9, and 11 were never seen to full completion.

**PROJECT BUDGET**

Table 5 displays a chart of the cost analysis for the entire development cycle of the pulse oximeter device. Items 4 through 8 show the different types of labor needed for the design and the costs associated with them. Items 11 through 22 show the different types of parts needed for the actual design and prototyping of the product. In further analysis of the parts, items 11 through 17 are specifically used for testing and proof-of-concept purposes, while items 18 through 22 are used to design the finished product.

Table 5: Cost analysis chart used for determining budget



Staying within budget was not an issue for this project, and all parts that were needed to be ordered were ordered. Most units involved with the PCB design were not ordered due to the project coinciding with a global pandemic.

**SUMMARY AND CONCLUSION**

The primary goal of this project was to create a custom FPGA-based pulse oximetry device that researchers can use as a brain-machine-interface system for conducting research. This custom design was to serve to be a foundation for an optimized data-collection and processing system. By the end of the project, the final goal was to produce a final PCB that meets all the needs of the research group intending to use the device, as well as a PCB that can be used for noninvasive human testing.

The first 8 weeks of the project were on target. The data processing system was designed and debugging was taking place. Regular meetings with Charlie allowed for PCB design to be done in parallel to FPGA design. Regular meetings with Dr. Elaraby ensured that I was on the right track and the best possible solutions were being assessed. Problems arose when too much time had elapsed without successfully implementing the data processing system, and this problem compounded when enforced remote working took place, where motivation and routine became a complicating factor as well. An attempt to implement an alternative solution was made, but that had problems as well. This led to being unable to see the project to completion on time.

**PROJECT AFTER ACTION REPORT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **1** | **2** | **3** | **4** | **Score** |
| **Understanding of customer needs** | We didn’t understand the customer at all | We understood about half of the customer needs | We understood most of the customer needs | We fully understood all customer needs | 3 |
| **Product delivery based on customer needs** | We didn’t meet any of the customer’s needs | We met about half of the customer’s needs | We met most of the customer’s needs | We completely met all of the customer’s needs | 3 |
| **Product delivery based on engineering requirements** | We didn’t meet any of the engineering requirements | We met about half of the engineering requirements | We met most of the engineering requirements | We completely met all of the engineering reqts | 3 |
| **Use of skills from engineering courses** | We did not use anything learned in previous courses | We used minimal skills from courses – 1 course | We used a number of skills from courses – 2-3 courses | We used skills learned in 4 or more previous courses | 4 |
| **Importance of information literacy (research, etc.)** | We did no research at all to accomplish the project | We did minimal research; 1 or 2 sources at the most | We used a number of sources for the project | We were continuously researching (the entire project) | 4 |
| **Importance of human consulting/networking** | We didn’t need to consult/network at all | We did minimal human consulting/ networking | We regularly consulted or networked | We were consulting almost weekly for the entire project | 4 |
| **Engaging in lifelong learning as an individual** | I see absolutely no need for continuing to learn | There may be a few things I need to learn in the future | As jobs change, there will be many things to learn | My life as an engineer will involve continuous learning | 4 |
| **Lifelong learning as a team (If applicable)** | I see no value in the team for lifelong learning | There are a few things I can learn from team members | There are many things to learn from team members | The team is critical for life-long learning in the future | - |
| **Functioning as a team (if applicable)** | Our team was completely dysfunctional | We managed as a team (barely) – team roles were confusing | The team functioned well most of the time | This was the best team experience I’ve ever had | - |

**FUTURE WORK**

The project provided a massive amount of educational benefit and will serve as my first official project in embedded system design. A large limiting factor was that, since it was my first time with embedded systems and FPGAs, much of the system design and individual Verilog code is overcomplicated, messy, and confusing. This caused many issues when trying to build onto the design and debug errors. Moving forward, much attention will be given towards creating designs that are easy to come back to and update on timescales months to years into the future.

While the project is not yet perfect, the findings from it can still benefit the research group, since many members are exploring the use of FPGA architecture and looking into its functionality and its pros and cons.

**LIST OF REFERENCES**:

[1] P. Tilakaratna, “How Pulse Oximeters Work Explained Simply,” 2012. <https://www.howequipmentworks.com/pulse_oximeter/>

[2] A. Van Meter, “Beat to Beat: A Measured Look at the History of Pulse Oximetry,” 2017.

<https://www.sciencedirect.com/science/article/pii/S235245291530030X>

[3] N. Gazivoda, Ž. Beljić, P. Sovilj, D. Pejić and B. Vujičić, "Measurement firmware for pulse oximetry sensor," 2016 Zooming Innovation in Consumer Electronics International Conference (ZINC), Novi Sad, 2016, pp. 42-45.

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7513651&isnumber=7513633

[4] Texas Instruments, “AFE4490 Integrated Front-End for Pulse Oximeters,” datasheet, 2014.

<http://www.ti.com/lit/ds/symlink/afe4490.pdf>

[5] Maxim Integrated, “Pulse-Oximeters: Design Considerations,” 2019. <https://www.maximintegrated.com/en/design/partners-and-technology/solutions/healthcare/pulse-oximeters.html/tb_tab1>

[6] W. S. Johnston and Y. Mendelson, "Investigation of Signal Processing Algorithms for an Embedded Microcontroller-Based Wearable Pulse Oximeter," 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, New York, NY, 2006, pp. 5888-5891.

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4463147&isnumber=4461641

[7] Greenlight Guru, “Medical Device Life Cycle,” 2019.

<https://www.greenlight.guru/glossary/medical-device-life-cycle>