FPGA-based Pulse Oximetry and Heart Rate Monitoring Device

Matthew Capuano, Nashwa Elaraby, Ph.D.

Abstract—The design and of an embedded system for pulse oximetry and heart rate monitoring using Field Programmable Gate Arrays (FPGAs) as the main controller is introduced. The main motivation for this design project is to create a pulse oximetry system that integrates into a complete brain-machine interface (BMI) device which monitors heart and brain measurements and processes them in real time. FPGA's can be used as an alternative to microprocessors in many embedded applications, biomedical included. Doing so usually increases the processing speed and capabilities of the embedded system due to the FPGA's ability to provide the parallel processing seen in hardware with the flexibility seen in software, all within a small chip. This project involved designing an FPGA that interfaced with a Texas Instruments AFE4490, a front-end pulse oximetry data acquisition device. The main aspects of the design involved data transfer, real-time DSP, and communication protocols. The system was designed and simulated using Intel Quartus Prime, and ModelSim was used for functional verification. The design was then implemented and tested using Intel Max10 and TI AFE4490 breakout boards. As both a baseline and comparison system for the FPGA design, an Arduino-based design was applied, analyzed, and compared to the FPGA system in order to conclude on the functionality and feasibility of an FPGA-based embedded design.

I. Literature Review

Pulse oximetry is a tool used in the measuring of blood oxygen levels of patients in clinical settings. It is both useful and important due to its ability of warning clinicians of potential hypoxia in patients, a condition that can quickly cause serious complications [1].

Pulse oximeters compute blood oxygen levels by measuring changes in the red and infrared light absorption characteristics of oxygenated and deoxygenated blood [2]. This is performed by using LEDs to illuminate a thin part of the body (i.e. finger) and using a photodetector to measure the amount of light absorbed on the other end [2]. The photodetector then generates an analog signal based on the amount of light it is receiving. It is expected that a digital embedded device will then receive the value of the analog signal, convert it to a digital signal, and use computational algorithms to convert the samples into heart rate and blood oxygen.

Microcontrollers are a standard for embedded system design due to being general purpose devices that are easy to program and understand [3]. However, FPGA's can offer truly parallel processing, while also being able to be designed in a way that offers maximum performance [3]. The flexibility involved with using an FPGA-based device also allows for addition peripheral devices, such as an EEG, to be added on to the system and whose data can be processed in parallel with the pulse oximeter.

II. Methods

The proposed system design for implementing an FPGA-based pulse oximetry device involved interfacing with the Texas Instruments AFE4490 mixed-signal front-end chip, processing the data on the FPGA, and sending the data out via an ESP2866 WIFI module to a cloud server.

The FPGA chosen for the design was the Intel Max 10 10MO4SAE1448G. This was due it having block ram, multiplier units, and logic gates that seemed appropriate for the design, as well as flash memory so that the system would retain the implemented design. Verilog was used as the hardware description language (HDL).

A. Texas Instruments AFE4490

The AFE4490 serves to drive the LEDs of the pulse oximetry probe, receive an analog signal from the photodetector of the probe, and convert that signal into a digital value. The AFE4490 samples the red and infrared LEDs independently, and also samples the ambient light value that exists after each LED is switched off. It then computes the difference between the LED and ambient light value as an effort to reduce noise coming from light sources other than the LEDs. These values are then stored in registers on the AFE4490, and the SPI communication protocol is required to extract the data from the device. The AFE4490 also has a large amount of configurability associated with it, and requires SPI communication to write to and configure the system. For this design, only the red and infrared LED values that the AFE4490 sampled were used for processing.

B. Main System Design and Interfacing with AFE4490

A large part of the FPGA design centered around interfacing with the AFE4490 in order to both configure it properly and extract information from it properly. The finite state machine(FSM) module was implemented in order to perform a

large sequence of operations, including initializing the AFE4490, running diagnostics on the device, writing configuration information to the device, and getting the system to stream data properly. The state machine executed these functions by sending control data to addressing, writing, reading, SPI, and DSP modules, which then determined the overall process and functions to be executed for each module during that specific clock cycle. The address module would then send addressing information to the read and write modules, which would then communicate with the SPI module to send both address and data associated with that address to the AFE4490.

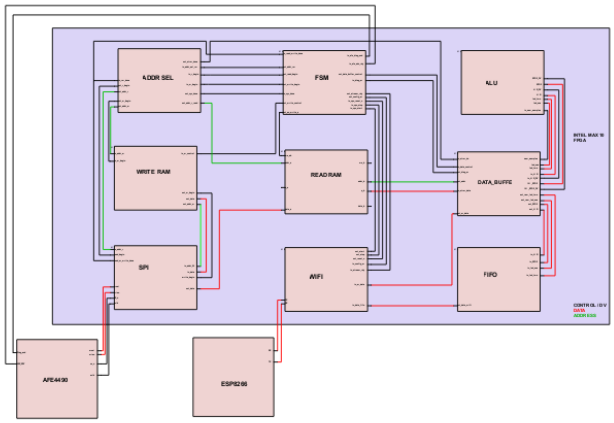

Figure 1: Schematic diagram displaying modules and I/O of the FPGA

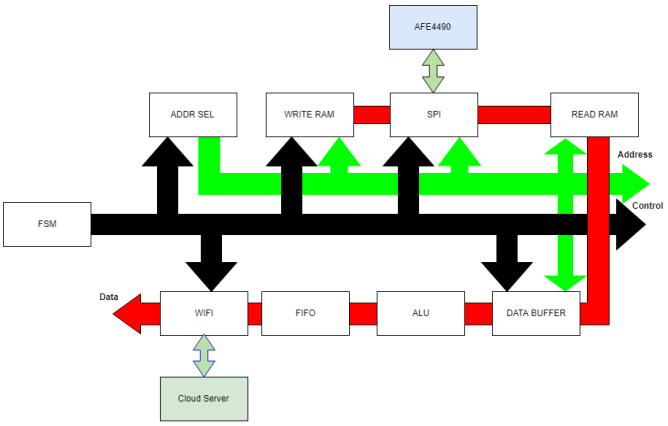

Figure 2: Simplified diagram displaying control, addressing, and dataflow of the designed system

An example of the control flow in the FPGA would be if the system wanted to write configuration information to the AFE4490. In this case, the FSM would send control data to the address and write modules, then send a data valid signal to the address module. The address module would then send addressing information based on its control information to the write module along with a data valid signal. The write module would then send information to the SPI module based on addressing and control data it is receiving, and the SPI module would format the information to be sent via SPI communication to the AFE4490.

C. *Real Time DSP*

After streaming samples are extracted from the AFE4490, samples get passed to a data processing module. The data processing module is a hardware adaptation of the Protocentral AFE4490 Oximeter open-source Arduino code. At a sampling rate of 500 samples per second, the top-level module functions to downsample, sending every 20th sample received into a dual port ram block until the system fills to 135 samples. This is done for both the red and infrared led values. A data valid signal is then sent to a submodule, which then performs processing on the data in the ram. The algorithm used is centered around moving averages and sorting algorithms. For heart rate, based on the data that is received, a threshold for peaks is determined. Then a sorting algorithm goes through and uses the threshold to determine where the peaks are in the signal, and how many there are. This information is then used with a formula involving the sampling rate to compute heart rate. For the blood oxygen calculation, a ratio of ratios of the AC and DC components of the red and IR LED values is calculated. The AC component of the signal was determined by calculating peak to peak values of the signals, while the DC component was determined by taking the average of the samples between those peak-to-peak values.

D. *WIFI Communication*

After computation of blood oxygen and heart rate had occurred, the information was to be sent via WIFI to a cloud server. As a proof of concept, a Rapsberry Pi was configured to function as a wireless access point, and an Apache web server was implemented on the Raspberry Pi in order to store information. On the server, a PHP file was written so that information sent to it would be taken and stored on a text file that was also on the server. Anyone connected to the server could then access and view the text file to see computed data.

In order to connect the FPGA to the Raspberry Pi and send data to the server on it, an ESP8266 WIFI modules was implemented. The ESP8266 has its own microcontroller on board that can be used to program the system, but it also comes pre-loaded with its own firmware. The preloaded firmware used AT commands in order send and receive data.

III. Results & Analysis

Each module of the FPGA was simulated in the ModelSim simulation tool inside Quartus before being integrated together and synthesized onto the FPGA. For prototyping purposes, the EarthPeople Technology MaxProLogic Development System was used as an FPGA testing platform, since it contains the same model of FPGA intended to be used for the final design. Once the Verilog modules were synthesized to the prototyping board, its GPIO pins were interfaced with the Protocentral AFE4490 breakout board, a breakout board containing the

AFE4490 and a pulse oximeter probe. Testing and analysis of data transfer from the AFE4490 to the FPGA was conducted.

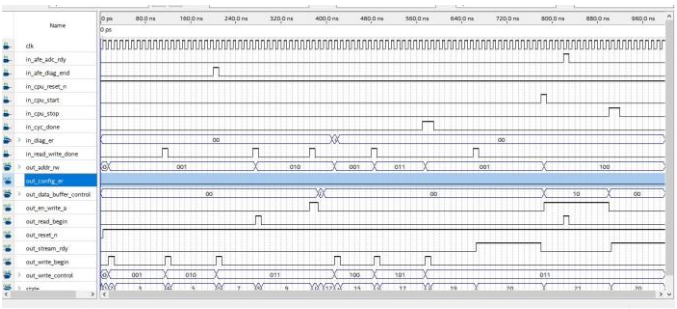

Figure 3: Simulated Timing diagram of the Finite State Machine

The WIFI modules were then designed, integrated with the rest of the code, and synthesized into the FPGA. These modules served to drive the functionality of ESP8266 WIFI device and send data being processed on the FPGA to it. For testing purposes, a breakout board of the NodeMCU ESP8266 12-E was implemented.

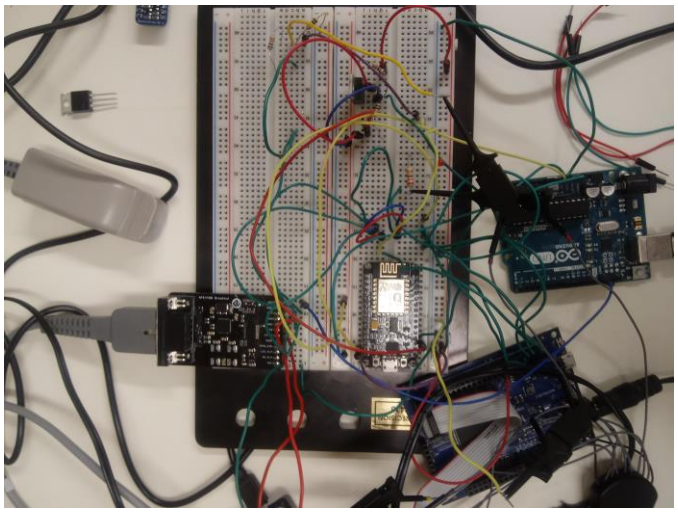

Figure 4: Picture displaying prototyping boards integrated together for testing. Left black board – AFE4490. Middle board – ESP8266. Bottom blue board – FPGA. Right blue board – Arduino Uno.

IV. CONCLUSION

Through both simulation and hardware testing, data transfer from the AFE4490 to the FPGA and then to the Raspberry Pi server was proven to occur. However, from viewing simulations, it was observed that the DSP algorithm had errors in its computation, and it was concluded that an adaptation of the DSP algorithm from microcontroller code may not be the most effective way of implementing a computation of heart rate and blood oxygen levels. Because the design is a for a pulse oximeter, accuracy of computation is valued over speed. Future developments will look into using alternative algorithms in order to maximize computational accuracy (i.e. Fourier Transform vs. sorting algorithm), implementing control algorithms to reduce noise from ambient light and control LED drive, and implementing remote control of the FPGA from a device connected to the server established.

REFERENCES

[1] Jubran, Amal. “Pulse oximetry.” *Critical care (London, England)* vol. 19,1 272. 16 Jul. 2015, doi:10.1186/s13054-015-0984-8

[2] N. Anju Latha1, B. Rama Murthy1, L. Suresh2. “Development of MSP based Pulse oximeter with LabVIEW” IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.

[3] Radovan Stojanovic and Dejan Karadaglic, “Design of an Oximeter Based on LED-LED Configuration and FPGA Technology” *Sensors* 2013, *13*(1), 574-586 https://doi.org/10.3390/s130100574.