A VHDL Based Controller Design for Photoplethysmography-Based Heart Rate Monitoring System

Kiet Duong, Dat Tran, and Ujjal Kumar Bhowmik
Electrical Engineering and Computer Science Dept.
Catholic University of America, Washington DC 20064, USA
Email: bhowmik@cua.edu

***Abstract*—Continuous monitoring of heat rate using wearable technology has the potential of improving healthcare and fitness and reducing the risk of cardiovascular diseases. A Photoplthysmography (PPG) based optical sensors are becoming popular to detect heart rate from human body. However, interfacing different sensors, acquiring and processing data in real time is a challenging task and requires dedicated hardware. Field Programmable Gate Array (FPGA) has become most widely used technology for real-time application. In this research, the necessary drivers and interfacing hardware for the PPG sensor are implemented on a FPGA platform using hardware description language (HDL). With the help of simulation and experimental results the accuracy and functionality of the proposed system are verified.**

Keywords—: Photoplthysmography (PPG), Heart-rate monitoring, FPGA, VHDL, Wearable healthcare.

I. INTRODUCTION

The emergence of micro-sensors and wireless technology has enabled changes in the conventional healthcare systems, replacing it with wearable technology. Wearable devices for measuring ambulatory heart rate becomes a new research topic in industry and academia. Several industries have recently introduced smart-watch and smart wrist-band for healthcare and fitness. Most of these smart devices use PPG based sensors to measure heart rate and other parameters from human body. Heart rate monitoring using PPG technology has many advantages compared to the conventional ECG, such as convenience in using, lower-cost, portability and continuous monitoring without any special assistance [1], [3], [4]. Recent advances in the optical technology, PPG based sensors are becoming more popular in clinical settings and fitness industries. However, integrating and interfacing different sensors in healthcare, fitness or other systems and acquiring and processing data in real time is a challenging task. The current trend in hardware design is implementing the complete design in a single chip. The FPGA based technology has become most successful and widely used technology for developing systems that require real-time signal processing [5]. In this research, the VHDL based hardware description language is used to design, implement the necessary interfacing hardware, controller, and signal processing units. The system is implemented and verified on an Altera DE2-115 FPGA board. During the development phases of our system, different signal processing modules are first designed using MATLAB. After successful implementation and verifications, the MATLAB code is converted to VHDL code using HDL-coder toolbox. A simulation software, ModelSim 10.3 PE, is also used for testing and verifying the functionality of different modules of our design.

The rest of the paper is organized as follow. Section II describes briefly the TI AFE4400 heart rate sensor used in this research, section III discussed elaborately the PPG data acquisition system, section IV discusses the MATLAB implementation of necessary filters, section V discusses the three-stage signal processing section, and finally a brief conclusion is given in section VI.

II. HEART RATE MONITOR AND PULSE OXIMETER SYSTEM

The TI AFE4400 is a fully-integrated analog front-end (AFE) specifically designed for pulse oximetry applications [2]. The PPG data is obtained by illuminating the skin and measuring the changes in light absorption caused by the pulses (Figure 1). The sensor consists of a low-noise receiver channel with an integrated analog-to-digital converter (ADC) and LED driver. The device communicates with the host processor using SPI communication [2] [6].

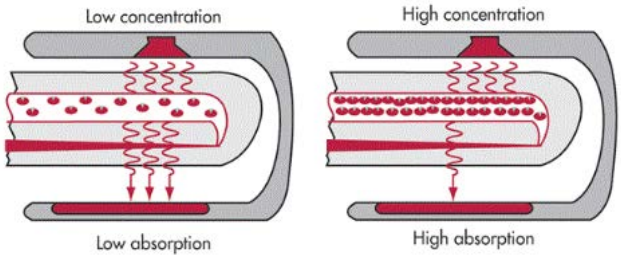

Figure 1. PPG signal acquisition

The sensor has two working mode: reflection mode and transmission mode as illustrated by Fig 2. In this project, reflection mode is chosen over transmission mode since it can obtain data from more parts of the body while not requiring exact placement like the transmission mode does. A typical PPG signal is shown in figure 3.

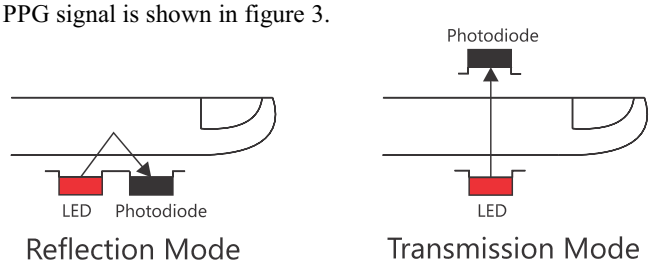

Figure 2. Two modes of PPG operation

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.156

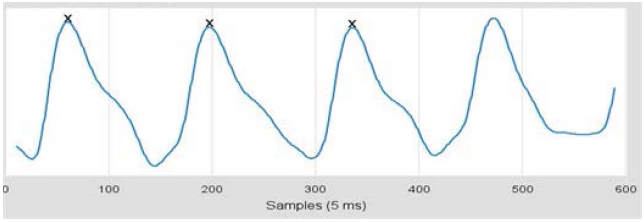

Figure 3. A typical PPG Signal

The required hardware for pulse oximeter system, as illustrated in Figure 4, consists of three main components: an AFE4400 chip, a photodiode (PD) and a dual LED (DLED) of wavelength 660nm (RED) and 905nm (IR) [2]. AFE4400 is in charge of driving the DLED. The sample data of the output current of the photodiode is stored inside the registers of AFE4400. Data can be read from AFE4400 via SPI interface using the FPGA driver developed in this research.

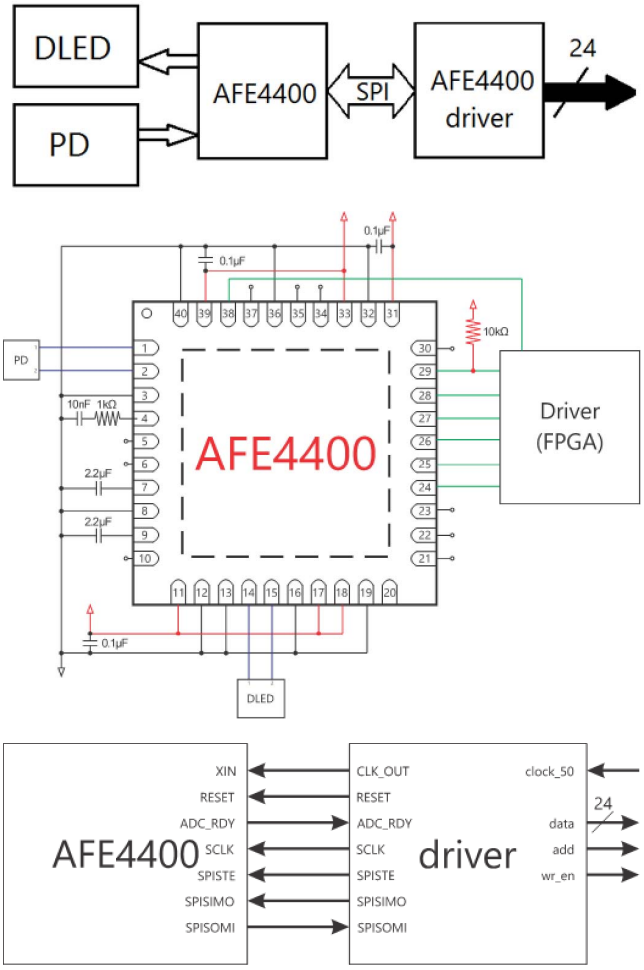

Figure 4. AFE4400 with necessary FPGA driver

III. PPG Data Acquisition System

After turning on the AFE4400, the FPGA driver will initialize the AFE4400 and configure it to desired operation mode by sending write commands to change the values of its internal registers. This step is required as the chip cannot work under default setting. When the initialization phase is done, the FPGA driver will send read commands to obtain the data whenever new data is ready. AFE4400 signals new data ready signal through ADC_RDY pin. All communications are done using SPI protocol [6]. SPI write and read protocol are shown respectively in figure 5.

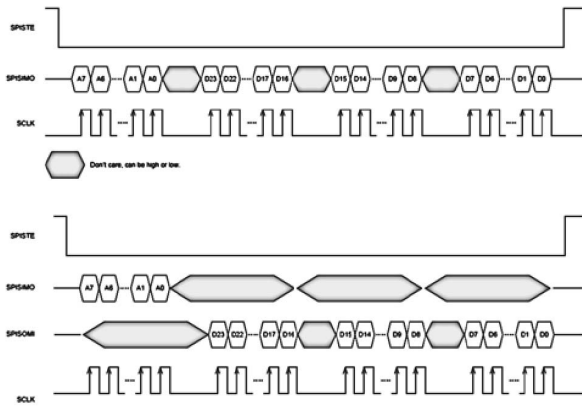

Figure 5. SPI write (top) and read (bottom) protocol

After being configured, the AFE4400's operation can be described by figure 6. For each pulse repetition period, it will first turn on the LED1 to get the sample data and then turn it off to get the ambiance data. It then does the same for the LED2. After obtaining each sample, it will convert them to 24-bit digital signal and store the result in its register while it takes the next sample. At the end of each period, it will overwrite the old data and drive the ADC_RDY signal high to inform the driver that new data is ready to be read. After that, it goes to the next repetition period.

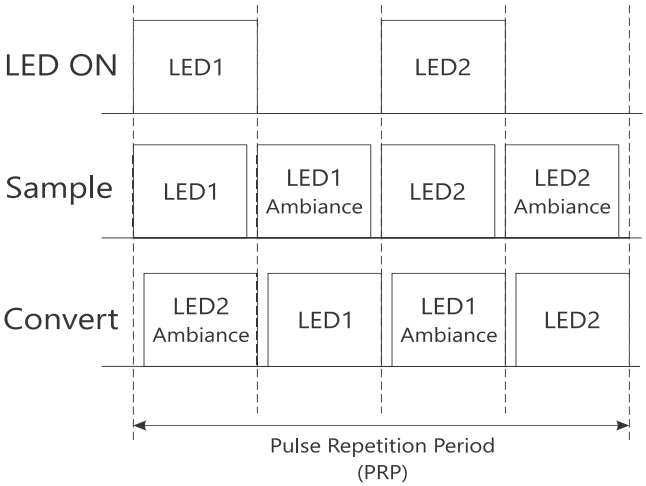

Figure 6. AFE4400 Operation

The frequency of the repetition period is equal to the sampling rate, which can be set between 62.5 Hz and 5 kHz. In this project, the sampling rate is chosen as 100 Hz and the duty cycle is reduced to 20% instead of 100%. The PPG signal obtained in low noise environment is shown in figure 7.

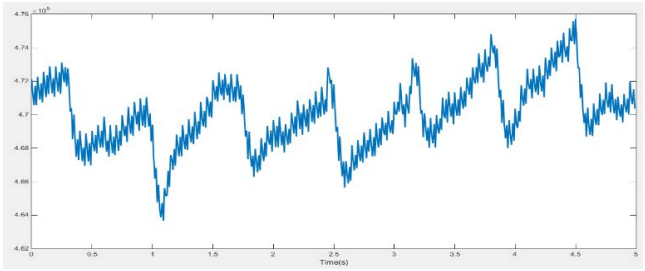

Figure7. PPG data obtained in low-noise environment

IV. Designing Filters Using MATLAB

After obtaining the PPG signal from the driver, the data will be processed to obtain the heart rate. The data from only one of the two LEDs is sufficient to calculate the heart rate. A sixth order Butterworth band pass filter with cut-off frequency of 0.5 Hz and 2.5 Hz, which correspond to heart rate of 30 BPM and 150 BPM respectively, is implemented in MATLAB to extract heart rate from sampled dataset. The Bode plot of the frequency response of the filter is shown in figure 8.

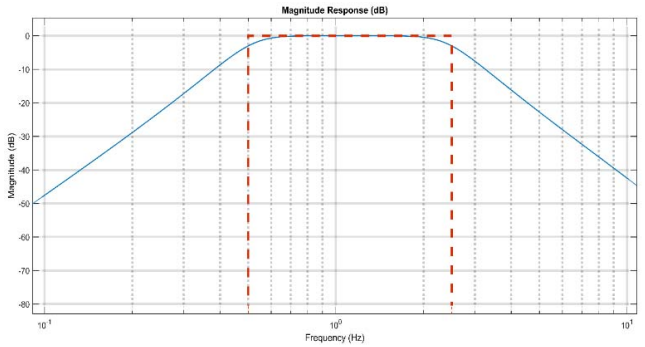

Figure 8. The Bode plot of the filter's frequency response

The recorded data from the sensor are imported to MATLAB to verify the functionality the algorithm. But since FPGA is better suited for fixed point arithmetic computation, a conversion of the floating-point arithmetic filter to fixed-point arithmetic filter has been implemented in MATLAB. It is also to be noted that in order to avoid overflow, the filter requires zero mean input. Therefore, one additional pre-filter step is required. The average of the data is computed and subtracted from the raw data in pre-filter step. From MATLAB simulation, the output of floating-point filter and the output of fixed-point filter are shown in figure 9. Note that the filters require some time to calibrate before giving the correct result. From the simulation, it is clear that the response of the fixed-point filter is consistent with the response of the floating-point filter. However, floating-point hardware implementation usually requires a lot of resource of the FPGA. For this reason, the fixed-point filter is developed in MATLAB and converted to VHDL code using HDL-coder toolbox in MATLAB R2014a.

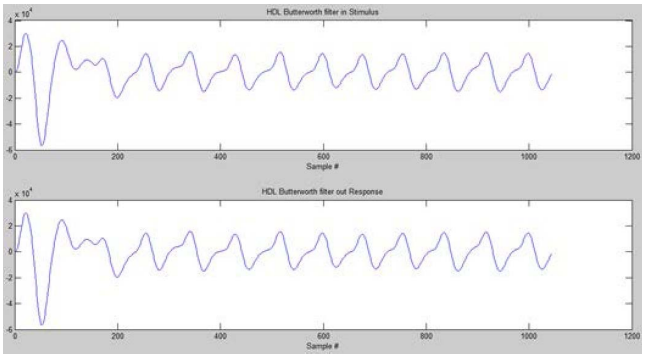

Figure 9. Floating point filter vs Fixed point filter

To verify the functionality of the filter and its performance in real-time, a stimulus list is created from the zero-mean data and stored in the FPGA's ROM and it is used as the excitation input for the VHDL filter. Using Signaltap II Logic Analyzer tool provided in Quartus II allowed us to examine the behavior of the output signal of the VHDL filter in real-time. The VHDL filter worked correctly as expected. In figure 10, the first signal is the input data and the second signal is the output data of the VHDL filter.

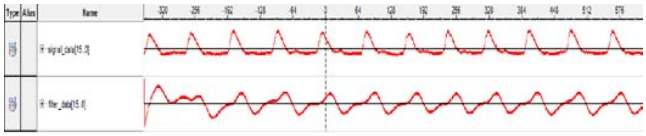

Figure 10. Logic Analyzer output waveform

V. Three-step Signal Processing For Detecting and Calculating Heart Rate

After verifying the functionality of the algorithm, the algorithm will be implemented on an FPGA platform. In this research, the PPG signal will be passed through three stages of digital signal processing (DSP), namely Pre-processing, Processing, and Post-Processing steps to extract the heart rate from sampled PPG dataset (figure 11).

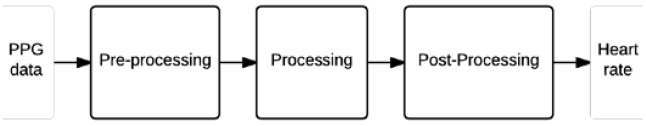

Figure 11. Three stages of DSP

In the pre-processing step, the signal will be passed through a pre-filter module to create the zero-mean signal required for the filter to work. To make the output a zero-mean signal, the pre-filter will subtract the mean value of a number of previous input data from the current input data. The output can be describe by the following function, in which $x[n] = 0\ \forall n < 0$

$$y[n] = x[n] - \frac{1}{N}\sum_{i=0}^{N-1} x[n-i]$$

The division operation usually requires a lot of resource. To reduce the amount of resource needed, N = 2^m, where m is a positive integer, is chosen in this research. Then the division operation can be closely approximated with a right shift of m-bit, which requires almost no resource in FPGA. Experimental results show that larger m gives better result. However, larger m also means more resources need to be used. To balance this trade-off, it is chosen as m = 5, for which the filter output is good enough for later processing while also keeping the amount of resource used reasonable. The output now becomes

$$y[n] = x[n] - \frac{1}{32}\sum_{i=0}^{31} x[n-i] = x[n] - x_M[n]$$

There are two approaches to realize this output, namely finite impulse response (FIR) and infinite impulse response (IIR). In the FIR approach, the mean value $x_M[n]$ is calculated with the formula

$$x_M[n] = \frac{1}{32}\sum_{i=0}^{31} x[n-i]$$

The block diagram to realize this function is shown below in figure 12. This approach uses a total of 31 registers and 31 full adders to calculate the mean value.

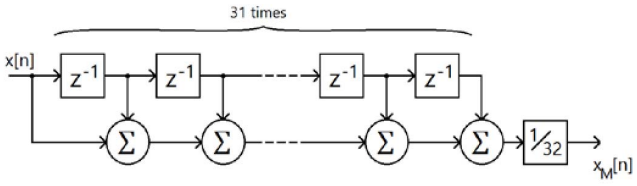

Figure 12. FIR Average Filter

In the IIR approach, the mean is calculated by the formula

$$\begin{aligned} x_M[n] &= \frac{1}{32}\sum_{i=0}^{31} x[n-i] \\ &= \frac{1}{32}\left[-x[n-32] + \sum_{i=1}^{32} x[n-i] + x[n]\right] \\ &= \frac{1}{32}\left[\sum_{i=0}^{31} x[n-1-i] + x[n] - x[n-32]\right] \\ &= \frac{1}{32}\left[x_M[n-1] + x[n] - x[n-32]\right] \end{aligned}$$

The block diagram to realize this function is shown in figure 13. IIR filter realization requires a total of 33 registers and 2 full adders. FIR approach gives slightly more stable results but uses significantly more resource, therefore, IIR approach is used in the final design. The final design for the pre-filter module is illustrated in figure 14. The input and output of the pre-filter module are shown in figures 15 and 16.

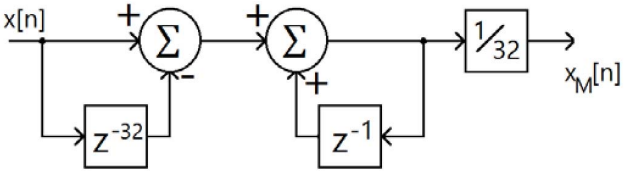

Figure 13. IIR Average Filter

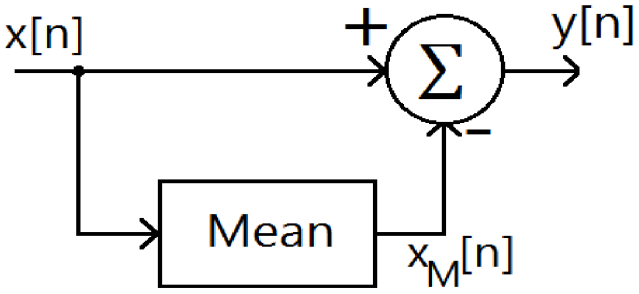

Figure 14. Pre-filter module

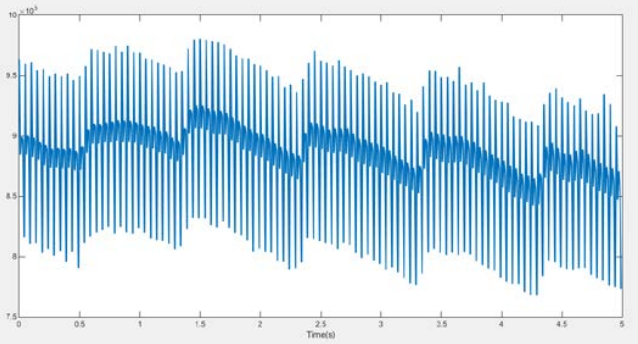

Figure 15. Noisy Input

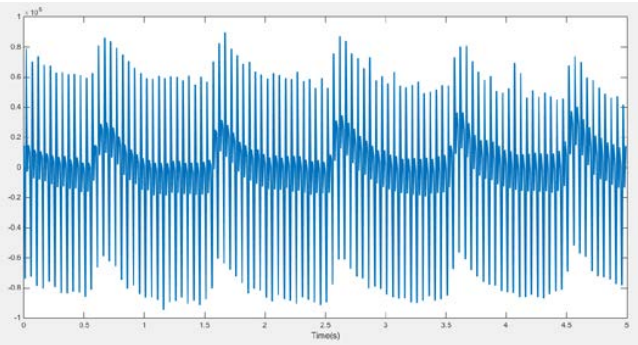

Figure 16. Pre-filter output of noisy input

In the processing stage, the pre-filter output will be fed to the band-pass filter generated by MATLAB. The filter output is shown in figure 17. It can be seen from the output that all of the noise have been filtered out.

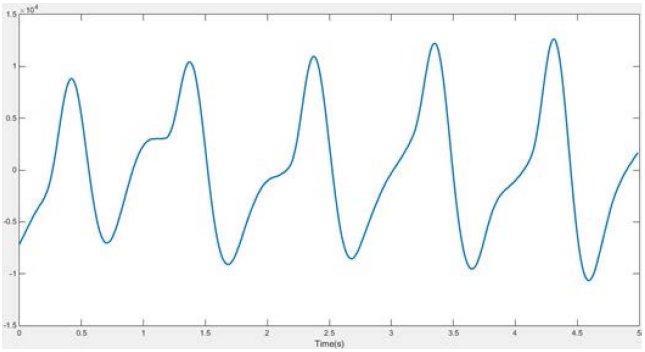

Figure17. VHDL filter output of the noisy input data

After filtering the noisy data, the filtered signal will be passed through a peak detector circuit to detect peaks in the signal. The block diagram of a typical peak detector is shown in figure 18.

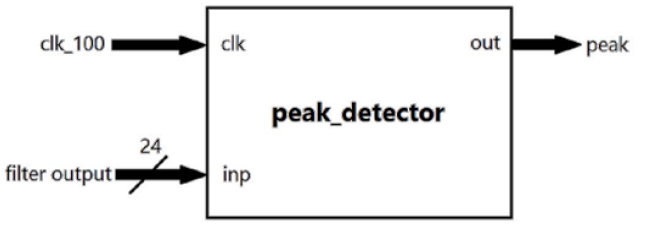

Figure 18. Schematic diagram of a Peak-detector circuit

The simplified description of the peak detector's operation is shown in figure 19. The peak detector will alternate between finding peak and finding trough in the signal to avoid false detection. For a value to be considered as a peak, it must be higher than a certain threshold and must stay as the maximum for at least 0.3 seconds. Currently, a static threshold value is used. However, to improve accuracy, a dynamic threshold calculated from past peaks should be implemented. Similarly, for a value to be considered as a trough, its value must be negative and stay as the minimum for at least 0.3 seconds. The "peak" output signal will be high for 30ms whenever a peak is detected. A typical output of the peak detector is shown in figure 20. The "peak" output of the peak detector will then be used as the input for the BPM_calc module to extract the heart rate. The schematic of the module is given in figure 21, in which the HR signal is an unsigned signal and the error signal means no peak has been detected for more than 2 seconds (i.e. the heart has stopped beating). The internal circuit and the detailed operation of the calculator are described in Figures 22 and 23. The pseudo codes for figures 19 and 23 are given in Table 1. To reduce the number of resources required to implement division operation in count2HR module, a look-up table generated by MATLAB is used in that module. After successful implementation of the proposed heart rate sensor, BPM_calc's output is compared against a manual observation of heart-rate on wrist. BPM_cal output is very much consistent with the manual observation. It is to be mentioned that the motion artifacts are not considered in this research. The primary goal of this research is to implement a working VHDL controller for AFE4400. In our future endeavor effort will be made to include an accelerometer in the system and verify the results in a medical setting against ECG output.

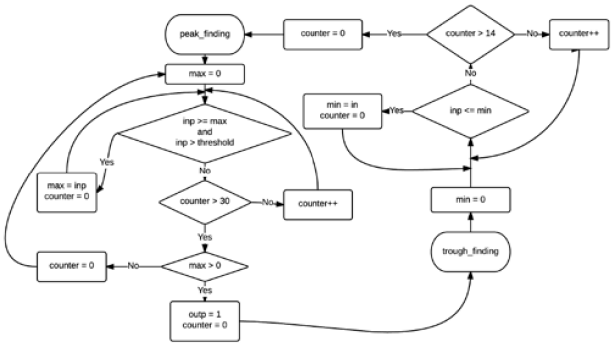

Figure 19. Peak detector's operation

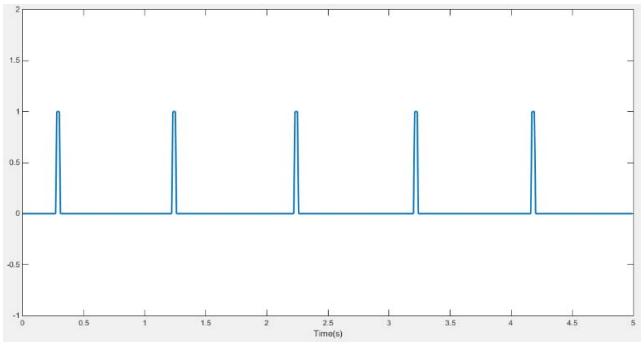

Figure 20. Typical peak detector output

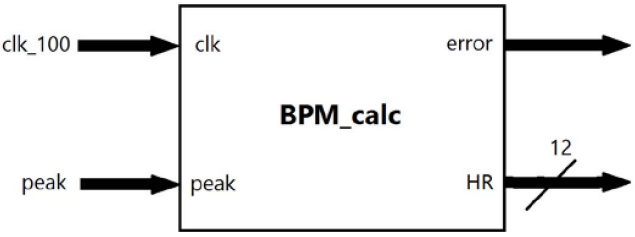

Figure 21. Schematic of a BPM_cal module

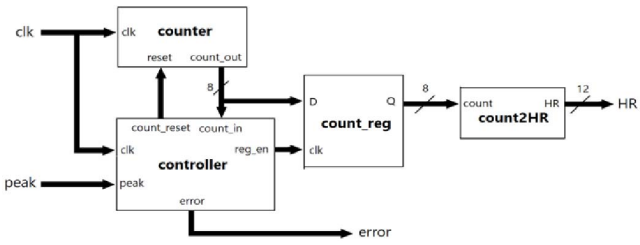

Figure 22. The calculator's internal circuit

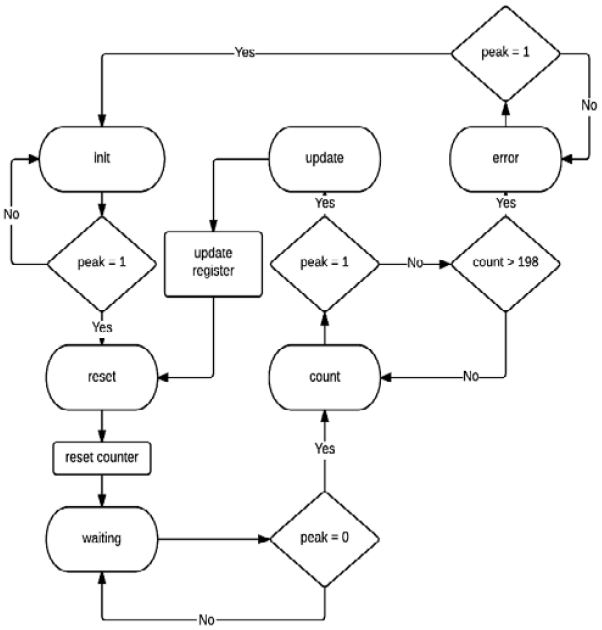

Figure 23. The Calculator's operation

Pseudo Code for Figure 19:	**Pseudo Code for Figure 23:**

WHILE TRUE
OUTPUT = 0
MAX = 0
WHILE COUNTER <= 30 OR
MAX = 0
IF INPUT >= MAX THEN
IF INPUT > THRESHOLD THEN
MAX = INPUT
END
COUNTER = 0
CONTINUE
END
IF COUNTER >= 28 AND MAX >
0 THEN
OUTPUT = 1
END
IF COUNTER = 31 THEN
COUNTER = 0
CONTINUE
END
COUNTER++
END
OUTPUT = 0
MIN = 0
WHILE COUNTER <= 14
IF INPUT <= MIN THEN
MIN = INPUT
COUNTER = 0
ELSE
COUNTER++
END
END
COUNTER = 0
END
	WHILE TRUE
	INIT: WHILE PEAK = 0
	END
	
	RESET: COUNTER = 0
	
	WAIT: WHILE PEAK = 1
	END
	
	COUNT: WHILE PEAK = 0
	IF COUNTER > 198 THEN
	GOTO ERROR
	END
	COUNTER++
	END
	
	UPDATE: REG_VALUE =
	COUNTER
	HR = COUNT2HR(REG_VALUE)
	GOTO RESET
	
	ERROR: ERROR_SIGNAL = 1
	WHILE PEAK = 0
	END
	ERROR_SIGNAL = 0
	GOTO INIT
	END

Table 1. Pseudo Codes for figures 19 and 23

VI. CONCLUSION

In this paper, we have designed and implemented a VHDL based controller suitable for Texas Instrumentation's AFE4400, a Photoplthysmography (PPG) based optical heart monitoring system. In the proposed system, we have implemented the SPI protocol to initialize and read data from AFE4400, we have designed a three-step signal processor to filter out heart rate signal from PPG dataset and finally we implemented a peak detector algorithm to detect heart rate for the processed data set. The design is tested against a set of manual observations of heartbeat taken from different persons. Experimental results are consistent with the heart-rate measured on wrist. In future, efforts will be made to test the system in clinical settings and make it suitable for ambulatory heart-rate measurement by incorporating accelerometer with it.

REFERENCES

[1] Toshiyo Tamura, Yuka Maeda, Masaki Sekine and Masaki Yoshida, "Wearable Photoplethysmographic Sensors—Past and Present," Electronics **2014**, 3, 282-302; doi:10.3390/electronics3020282

[2] AFE4400 Integrated Analog Front-End for Heart Rate Monitors and Low-Cost Pulse Oximeters [Online]. Available: http://www.ti.com/lit/ds/symlink/afe4400.pdf.

[3] M. Elliot and A. Coventry, "Critical Care: The Eight Vital Signs of Patient Monitoring" British Journal of Nursing, vol. 21, pp. 62-625, May. 2012.

[4] N. Cooper, K. Forest and P. Cramp, Acute Care, 2nd Ed. Malden, MA: BMJ Books, 2006.

[5] Dat Tran, Kiet Duong and Ujjal K. Bhowmik, "A VHDL Based Controller Design for Non-contact Temperature and BreathingSensors Suitable for Crib," *IEEE BIBE 2014,* Nov. 10-12, 2014, Boca Raton, Florida, USA.

[6] M.Jyothi, L.Ravi Chandra, M.Sahithi, S.Daya Sagar Chowdary, K.Rajasekhar, K.Purnima, "Implementation of SPI Communication Protocol for Multipurpose Applications with I2C Power and Area Reduction," International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 2, Mar-Apr 2012, pp. 875-883