A VHDL Based Controller Design for Non-contact Temperature and Breathing Sensors Suitable for Crib

Dat Tran, Kiet Duong, and Ujjal K. Bhowmik

Abstract— **Continuous monitoring of vital signs has the potential of improving infant health care and reduce SIDS (sudden infant death syndrome). With the emergence of micro-sensors and wireless technology can enable change in the conventional health care systems, replacing it with wearable and non-contact based wireless health care system. In this paper, an infra-red non-contact temperature sensor, and a microphone based breathing sensor are used to detect vitals sign of an infant. Interfacing different sensors and acquiring and processing data in real time is a challenging task and requires a dedicated hardware. Field programmable gate array (FPGA) based technology has become the most widely used technique for real-time application. The HDL (hardware description language) based FPGA implementation of necessary drivers and interfacing hardware are proposed in this paper. With the help of necessary simulations and experimental results, the functionality and accuracy of the proposed system are justified.**

Key words-Vital sign, FPGA, HDL, non-contact sensors, SIDS, Infra-red temperature sensor, respiratory sensor.

I. Introduction

Vital signs, such as temperature and respiratory rate, have long been used by the physicians to detect any immediate health concerns. Vital signs are important marker of human health condition. For example, the body' temperature represents the balance between the heat produced and heat lost, also known as thermal regulation. In the clinical environment, body temperature may be affected by factors such as underlying pathophysiology (e.g. sepsis), skin exposure (e.g. in the operating theatre) or age [1]. Respiratory rate measurement serves a number of purposes, such as being an early marker of acidosis. It is also one of the most sensitive indicators of critical illness [2]. An increase from the patient's normal respiration rate is an early and important sign of respiratory distress and potential hypoxaemia [3].

Infant needs constant care. Continuous monitoring of temperature and respiratory rate can provide an extra level of protection in infant health care. Several companies are working on vital sign monitoring systems. Some of the popular products are Mimo Baby [4], and Healthdyne System [5]. Most of these products are developed based on wearable technology in which sensors require physical contact. The main disadvantage of the contact based system is not only it is inconvenient for the infants but it also poses threat to infant health. The heat and the radiation generated by the electronic circuits could have an adverse effect on infant health. A body of researchers has been working on non-contact based microwave breathing sensors [6-9]. However these breathing sensors are still in the preliminary stages of research. There is not enough data available to justify the validity of using microwave sensors on infants. To overcome the limitation of existing approaches, we proposed a non-contact based vital sign detection system which uses an infra-red temperature sensor and microphone to detect temperature and respiratory rate, respectively. Interfacing different sensors and acquiring and processing data in real time is a challenging task. The current trend in hardware design is towards implementing a complete system in a single chip. Field Programmable Gate Array (FPGA) technology has become the most successful technology for developing system which requires a real time application. Our proposed system is designed using hardware description language. The system is implemented and verified on an Altera DE2-115 FPGA board. During the development phases of our system, a simulation software, ModelSim 10.3 PE, is used for testing and verifying the functionality of different modules of our design.

The organization of the rest of the paper is as follows. Section II, firstly describes the operation and communication protocol of the temperature sensor and then with the help of flow chart, simulations and experimental outputs it discusses in detail the temperature controller. In section III, the design and implementation of an FPGA controller of a microphone based breathing sensor is discussed. In section IV, a brief discussion on challenges and future works is given. Finally, the paper concludes with a short conclusion.

II. Interfacing Non-Contact Temperature Sensors

A. Infra-red temperature sensor

A medical grade infra-red temperature sensor, MLX90615 (Melexis Microelectronics Integrated Systems), is used in our research for non-contact temperature measurement. The MLX90615 is a 4-pin device (as shown in Fig 1), in which both IR sensitive thermopile detector chip and the signal conditioning chip (which has low noise amplifier, 16 bit ADC, a powerful DSP unit) are integrated in the same package [10]. This smart sensor is chosen so that the temperature measurement can be carried out with a minimal impact on the test subject, which in our case is infant in a crib. The non-contact methods measure the heat radiated from the object by measuring the frequency of the heat.

The Authors are with the Electrical Engineering and Computer Science Department, Catholic University of America, Washington, DC 20064, USA, (e-mail: bhowmik@cua.edu).

978-1-4799-7502-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BIBE.2014.11

Pin Name	Function
SDA/PWM	Digital I/O. In SBM mode serial Data I/O
VDD	External supply voltage
SCL	Serial clock input. (for 2-wire communication protocol)
VSS	Ground. The metal is also connected to this pin.

Figure 1. MLX90615 and its Pin functions

MLX90615 has two modes of communication, one is PWM and another is SMBus (default mode). In our research, we use an FPGA controller as a master device (CON1, as shown in Fig 2.) which communicates with MLX90615 via SMBus protocol. The SMBus interface is a 2-wire protocol, allowing communication between the Master Device (MD) and one or more Slave Devices (SD). In the system only one master can be present at any given time. The MLX90615 can only be used as a slave device. The MLX90615 supports only Read Word and Write Word commands of SMBus interface. The bus protocol for SMBus temperature read is shown in Fig 3. First 16 bits, which are sent by master, are for setting up the SD in read mode. Next, the master send 7 bit slave address with RD=1 to receive 16 bits temperature reading and 8 bits PEC from slave. After every 8 bits received by the SD an ACK/NACK is followed. Only the SD with recognized SA gives ACK, the rest will remain silent. If the SD does not send ACK the MD should stop the communication and repeat the message. A NACK could be sent to SD after the PEC. This means that there is an error in the received message and the SD should try sending the message again. The PEC calculation includes all bits except the START, REPEATED START, STOP, ACK, and NACK bits. The PEC is a CRC-8 with polynomial X8+X2+X1+1. The Most Significant Bit of every byte is transmitted first [10]. An example read word format is also shown in Fig 4.

B. MLX90615 Controller

The block diagram of MLX90615 controller for temperature reading is shown in Fig 5. MLX90615 interface module generates clock signal, SCL of 17.5 KHz, for MLX90615. This module also generates and control the direction of SDA according to read word format timing diagram [11]. The main function of this module is to send the read command continuously to acquire new temperature reading from MLX90615. At each time when it finishes receiving MSB of objects temperature, it asserts "ld" signal high for one clock cycle and continues to receive CRC packet (8-bit PEC) and sends it to CRC controller. This module also outputs the 16-bit sensor data to the CRC controller.

CRC controller is a finite state machine, which has three state (**IDLE, SHIFT and COMPARE**) and two external inputs (ld and done). Assume that CRC controller is initially in **IDLE** state (as shown in Fig 6). It moves to the **SHIFT** state when "ld" changes to "1". Before changing to **SHIFT**

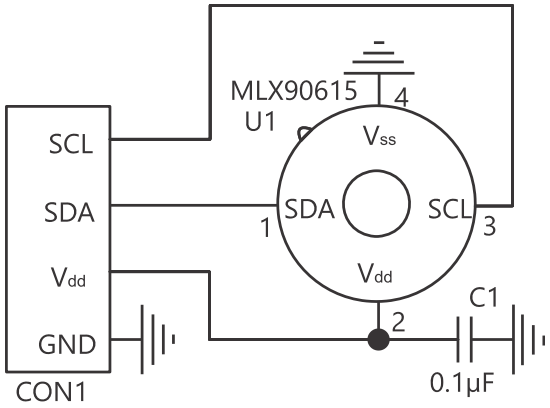
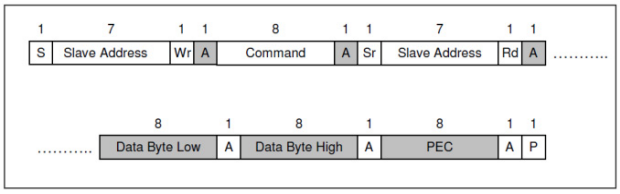


Figure 2. Typical MLX90615 connection to SMBus



S	Start Condition
Sr	Repeated Start Condition
Rd	Read (bit value of 1)
Wr	Write (bit value of 0)
A	Acknowledge (this bit can be 0 for ACK and 1 for NACK)
P	Stop Condition
PEC	Packet Error Code
	Master-to-Slave
	Slave-to-Master

Figure 3. Read word format

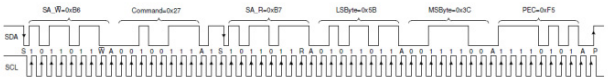


Figure 4. Typical Read word example

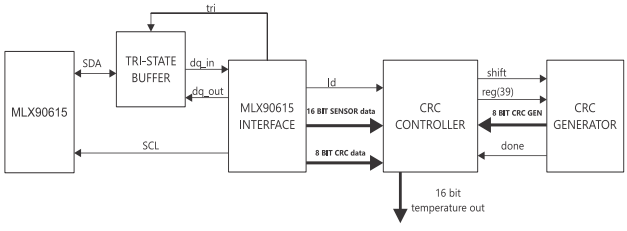


Figure 5. MLX90615 Controller

state, a value by concatenating "B627B7" (hex value, first 3 bytes of read word format) and sensor data is stored in to a

register. When it is in **SHIFT** state, the "shift" signal is asserted high and the register is shifted left in each cycle. If "done" signal is "1", the state machine will move to the **COMPARE** state. If CRC_data, which is received from MLX90615, is equal to CRC_GEN which is generated by CRC Generator, then the received sensor data is considered as the correct temperature reading and the CRC controller assigns sensor data as temperature data and it moves back to idle state after one clock pulse.

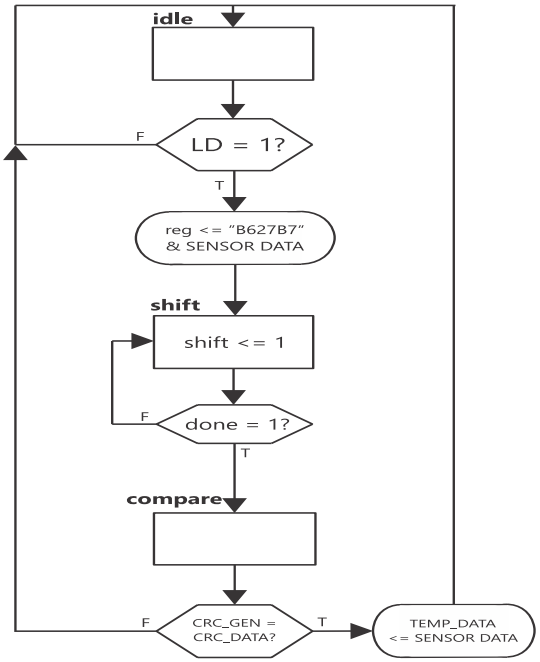


Figure 6. CRC Controller ASM chart

The most significant bit of the 40-bit register in CRC_CONTROLLER is connected to input of CRC_GEN module. CRC_GEN module contains an 8-bit CRC shift register (as shown in Fig 7) [12]. The input bit is shifted into the very left XOR gate. The MSB (leftmost bit) of each byte is shifted in first. CRC_GEN module just only shift the input bit if the "shift" signal is high. When it finishes shifting all 40 bits of CRC_CONTROLLER register, it feedbacks "done" signal with CRC_GEN data.

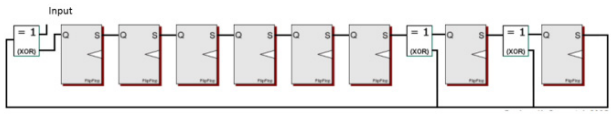


Figure 7. CRC Shift register

C. *Simulation and Experimental Results*

The simulation of our VHDL code is shown in Figure 8. ModelSim PE 10.3a is used for our simulation. From the Simulation results it is clearly observed that our MLX90615 Controller produces desired results. This VHDL is controller is implemented on an Altera DE2-115 FPGA board. After receiving the temperature reading the data is processed and displayed on a VGA monitor (Fig. 9). Experimental results verify that our controller produces desired results. However, according to the data sheet of MLX90615 the temperature measurement is dependent on the field of view (FOV) and the distance of the object from the sensor. Thus necessary calibration needs to be done for getting accurate temperature reading. In our future endeavor, our system will be attached on a crib and necessary calibration will be made to get accurate results.

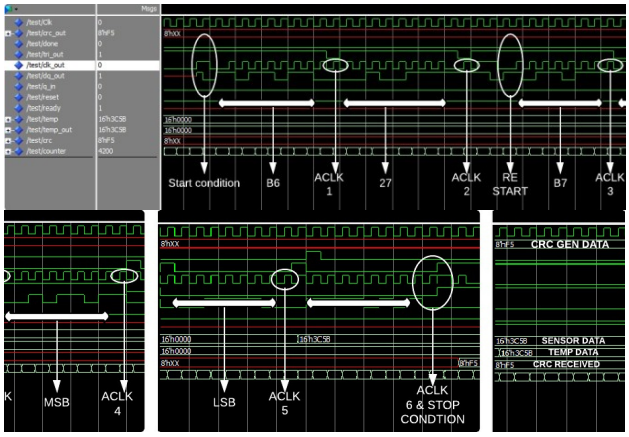


Figure 8. MLX90615 Controller simulation

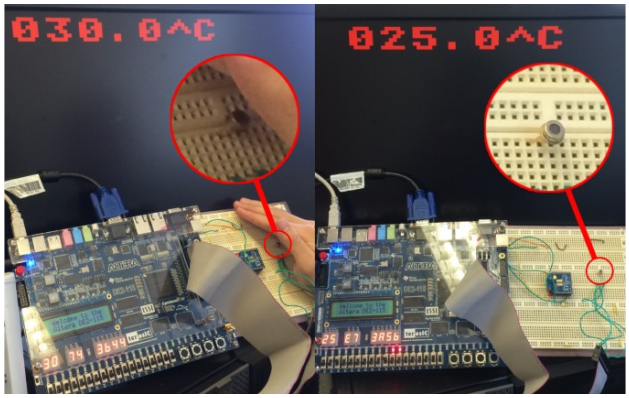


Figure 9. Experimental result of temperature sensor

III. Breathing Detection System

With the desire to build a reliable, low cost, and real time breathing detection system, we have implemented a microphone based breathing detector using hardware description system. The block diagram of our proposed system is shown in Figure 10. The core component of our system is Wolfson WM8731 audio CODEC. This chip supports microphone-in, line-in, and line-out ports, with a sample rate adjustable from 8 kHz to 96 kHz. The WM8731 is controlled via serial I2C BUS INTERFACE. Other necessary modules are DATA FETCHER, FILTER, and PEAK DETECTOR. Brief description of each of the module is given below.

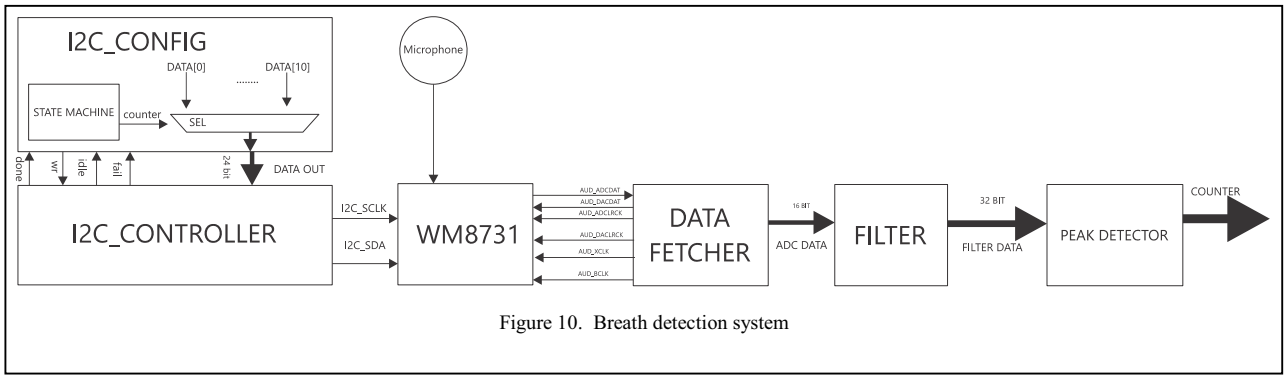


Figure 10. Breath detection system

A. *I2C Interface and I2C Controller*

The I2C protocol is a widely used low-speed serial bus for efficient communication between devices [13]. It's standard mode supports a data rate up to 100 K bits per second. The basic timing diagram of a typical data transfer with read write sequence is shown in Fig 11. The I2C bus consists of two bidirectional lines, sda (for "serial data") and scl (for "serial clock"), for data and clock, respectively. The two lines are connected to the sdin and sclk pins of WM8731.

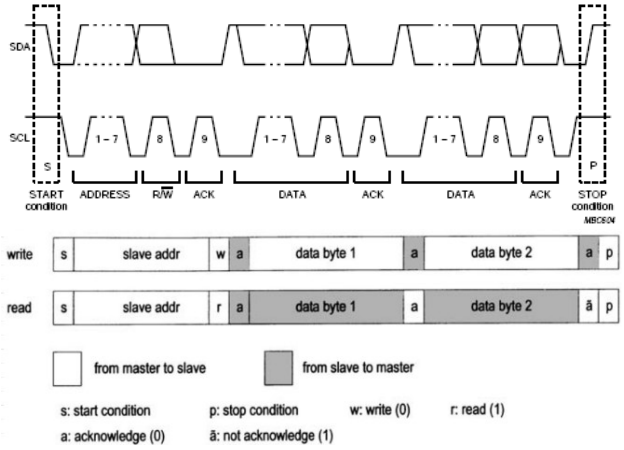


Figure 11. Read Write sequence and Timing diagram

For our purpose, we need to just perform write operation to configure the 11 internal registers of WM8731. Custom I2C controller has been designed to serve our purpose. The flow chart of the I2C controller operation is shown in Fig 12. Assume that I2C_CONTROLLER is initially in **IDLE** state. It moves to **START** state if "wr" signal is high. At **START** state the I2C controller generates the start condition signals. At **DATA** state, it sends a byte serially to I2C_SDA pin. Since we need to transmit 24 bits for configuring the internal register of WM8731, we divided the packet into 3 bytes. By tracking number of bit has been sent, we can know when it finishes sending a byte or not. After sending a bytes, it moves to **ACK** state and checks if it finishes transmitting 3 bytes or not. If not, it moves back to **DATA** state and transmits remaining bytes. When **ACK** is 1, it assert fail flag high and moves to **stop** state. At **STOP** state, it just simply transmits stop condition and after that asserts "done" signal high and moves to **RESUME** state because we must need some additional time before start sending another packet again. Our I2C Controller is designed using VHDL. We have used ModelSim 10.3 to verify the functionality of our I2C controller. The simulation results as shown in Figure 13 justify that the I2C Controller produces expected results.

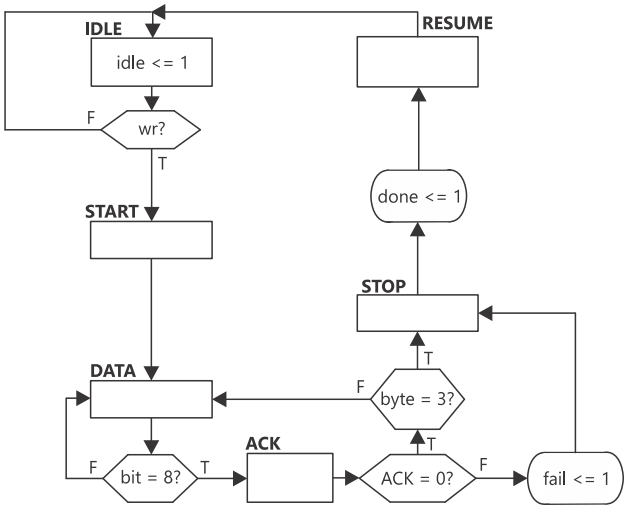


Figure 12. I2C Controller ASM

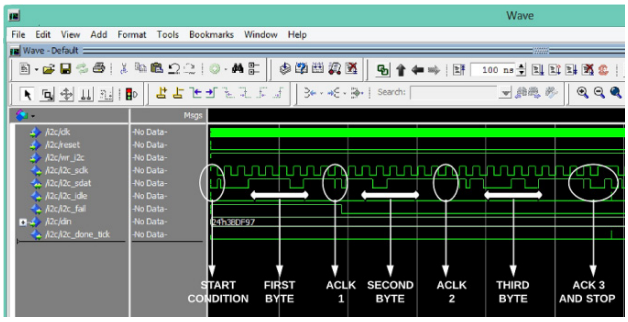


Figure 13. I2C Controller simulation

B. *I2C_CONFIG Module*

I2C_CONFIG module is used for sending data to I2C_CONTROLLER for configuring 11 internal registers of

WM8731. The flow chart of I2C_CONFIG is shown in Figure 14. Assume that I2C CONFIG is iniatially in **SEND** state. It moves to **WAIT DONE** state when "idle" signal from I2C_CONTROLLER is high. Before moving to **WAIT DONE**, it asserts "wr" signal high. If "done" signal is high, it checks "fail" signal . If "fail" signal is "1", it moves back to **SEND** state for sending that packet again. When it succeeds in transmitting a packet, it increases the counter by 1. If the counter equals 11, it moves to **FINISH** state and stays at this state forever. If counter is not equal to 11, it moves back to **SEND** state for sending remaining packets. I2C_Config module is implemented using VHDL. The simulation results shown in Fig verifies the functionality of our design.

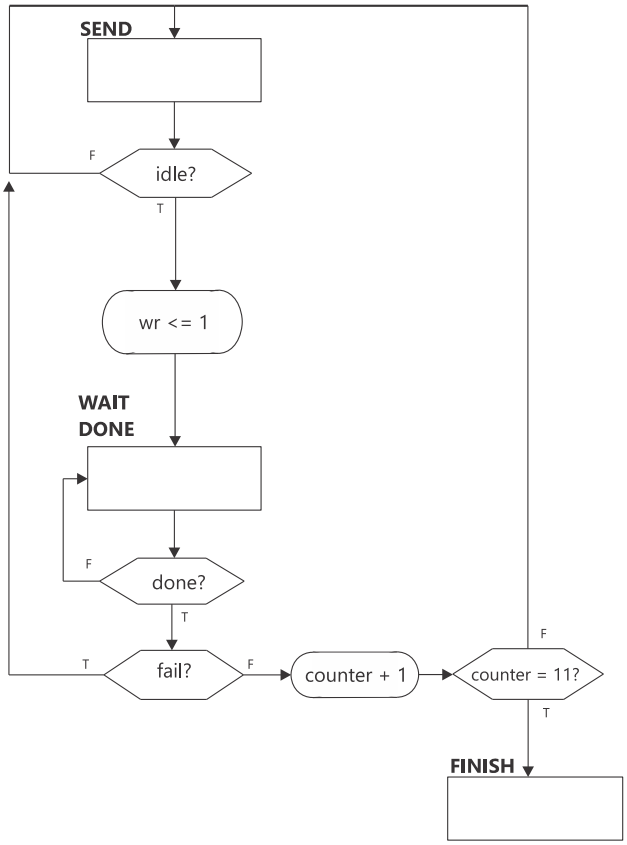


Figure 14. I2C Config ASM Chart

C. *Data Fetcher Module*

In our project, WM8731 is configured to operate in left justified mode and 16 bit resolution at 48 KHZ sampling rate. DATA_FETCHER module is used for generating AUD_ADCLRCK, AUD_DACLRCK, AUD_XCLK, AUD_BCLK and conversing from serial to parrarel of the incoming ADC data or parrarel to serial of outgoing DAC data. The AUD_BCLK signal functions as a bit clock and a new data bit is transmitted out in the AUD_ADCDAT line at every high-to-low edge. The AUD_ADCLRCK signal is an alignment clock that indicates whether the left- or right-channel data is presented in the adcdat line. The MSB is transmitted first. For a sampling rate of fs, there are fs samples per second and each sampled data must be transferred in the interval of 1/fs second [13]. The interval becomes the clock period of AUD_ADCLRCK, as shown in Figure 16. The simulation of this module is shown in Fig 17.

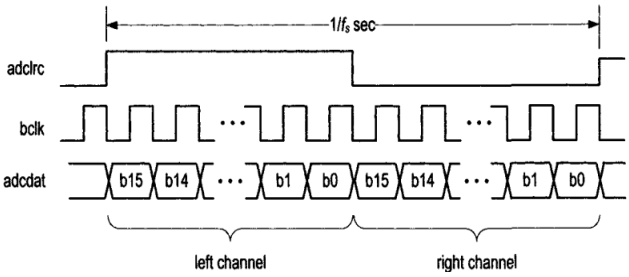


Figure 16. Data Fetcher timing diagram

D. *VHDL Filter*

The VHDL filter is a band pass filter of range 300Hz to 800 Hz to reduce the influence of external noise. It also provides the envelope of the signal for simple peak detection to be done later. Because of the complexity, this filter was first generated by Matlab and then converted to VHDL by Matlabs. Its input is 16 bit 2' complemented data and its output is 32 bit unsigned data.The filter needs to run at the same frequency of the sample rate of codec. Therefore, we use ADC_ADCLRCK as its clock input. The input data is supplied from the righ channel of audio codec (as shown in figure 16) [14].

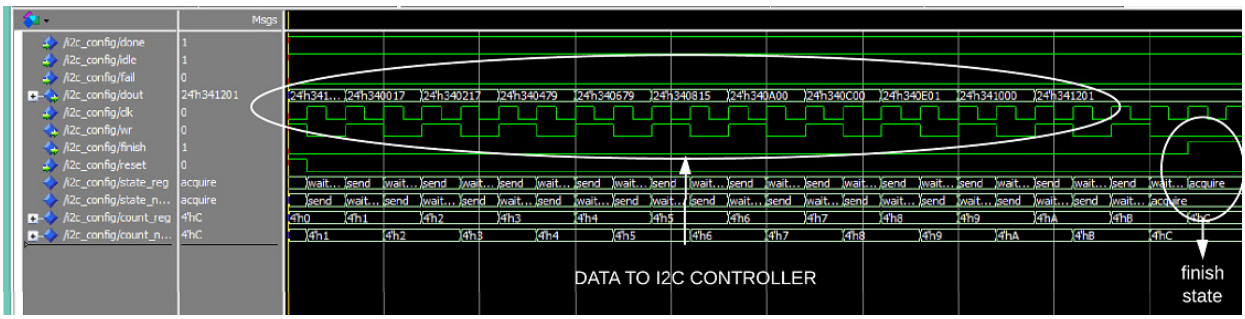


Figure 15. I2C Config simulation

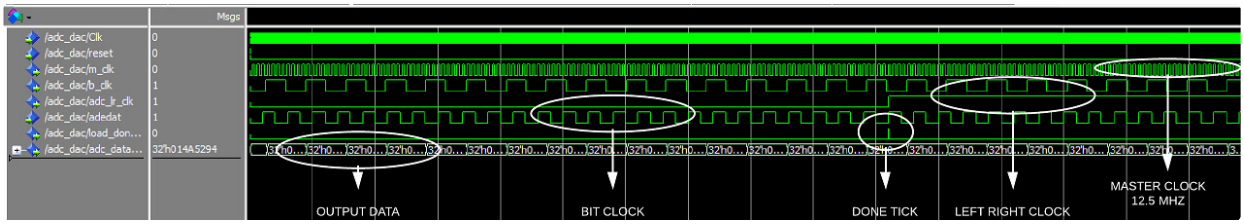


Figure 17. Data Fetcher simulation

E. *Peak Detector*

This module is used for detecting the peak of the VHDL filter output (filtered breathing data collected by microphone). Peak detection algorithm is shown in Fig 18. At first , we have to find the maximum value in an interval of time. After that, we compare the maximum values of three consecutive intervals. If maximum value of the middle interval is the greatest of 3 maximum values and greater than a choosen threshold, we detect a peak and a breating is counted. We do that for all the intervals. We chose the interval small enough, so that we minimize the probability of missing any breath even if people breath very fast. By comparing the peak value with threshold, we can reduce the false detection from background noise. Based on observation, a threshold value of 2^30 is choosen for reducing false peak from external noise. We can decrese the threshold for increasing the detection range at the cost of detecting more false peaks.

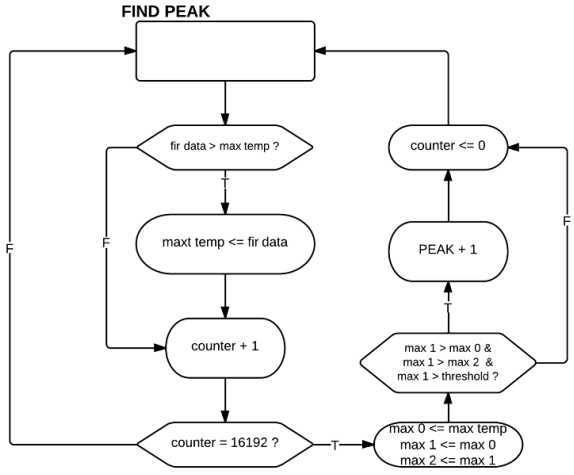


Figure 18. Peak Detector Algorithm

To check the functionality of the peak detector algorithim, we transferd the VHDL filter output to the laptop using USB port. To do that, necessary steps are shown in Fig 19. We applied the matlab version of the peak detection algorithim on VHDL filter data set. We checked it for 4 peaks and 5 peaks cases and plotted the results in the matlab. From these plots, it can easily be seen that our peak detection algorithm provides desired results (Figures 20 and 21).

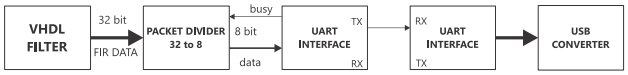


Figure 19. Data transferring system

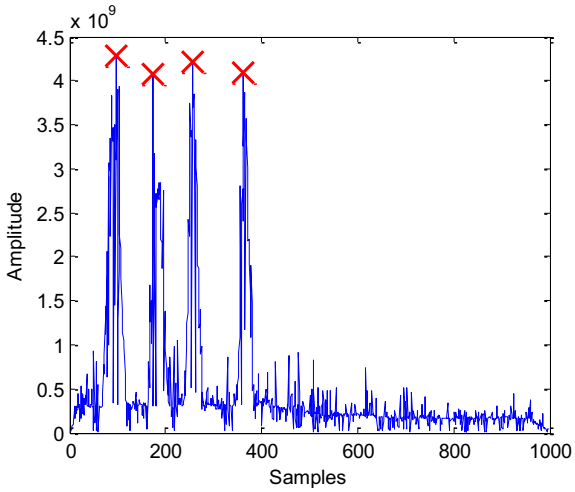


Figure 20. 4-Peak case

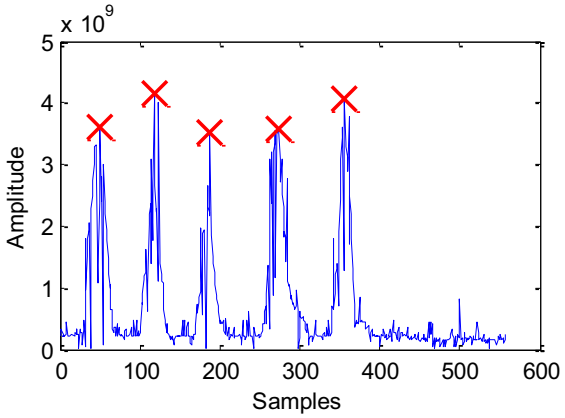


Figure 21. 5 peaks case

F. *Implementation on an FPGA board*

After successful simulation of each sections, we have implemented our Breathing Detection System on Atera DE2-115 board. The DE2-115 board provides high-quality 16 bit to 24-bit audio via the Wolfson WM8731 audio CODEC (Encoder/Decoder). This chip supports microphone-in, line-

in, and line-out ports, with a sample rate adjustable from 8 kHz to 96 kHz. The WM8731 is controlled via our serial I2C bus interface, which is connected to pins on the Cyclone IV E FPGA. A schematic diagram of the audio circuitry is shown in Figure 22.

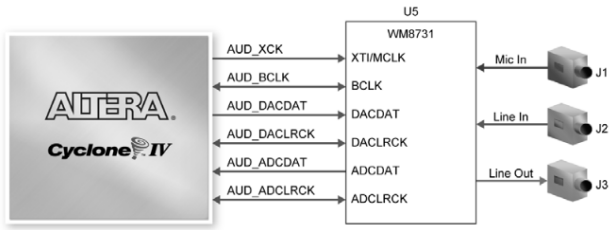


Figure 22. Connection between FPGA and Codec

The value of the peak detector counter is converted in BCD format and displayed on a seven segment display of the DE2-115 board as shown in Fig 23. We have used an ordinary microphone to verify the functionality of our system. The proposed breathing detection system produces expected results. Currently, we are in the process of testing our system using several commercially available small sized high sensitive microphones suitable for using with the crib. We are mainly focusing on the suitable interval size and threshold value to tackle the background noise.

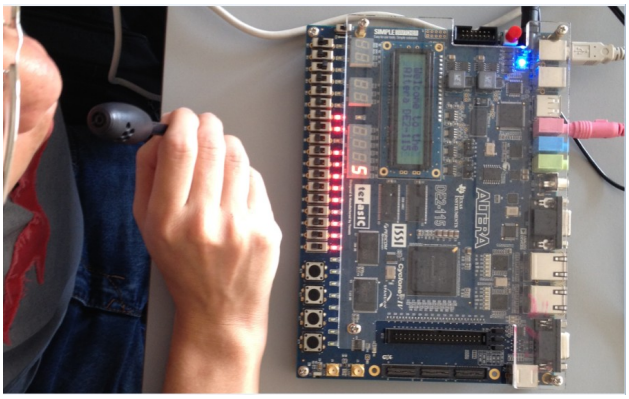


Figure 23 Breathing Detection Experiment

IV. Discussion

Our current research is focused on developing a low cost, non-contact vital signs detection system suitable for using in a crib. We have designed the necessary components for sensing temperature and respiration of an infant. However, to design a reliable system, we need to use several temperature sensors and microphones on the crib. The whole idea is, depending on the position of the infant in the crib, different sensors will produce different output and by continuously monitoring all the sensors, we can minimize the number of false alarms. For that purpose, we have designed and implemented a data transfer controller module (update module as shown in Fig 24) which will monitor all the sensors and whenever any updated sensor data is available, the module will transfer the data to a remote station through XBee wireless connection. The remote station can process all the sensor data and generate alarm signal to the caregiver or a mobile device of parents. A brief description of the update module is given in the Appendix section.

V. Conclusion

In this paper, we have designed and implemented a non-contact temperature and respiratory rate detection system. A medical grade infrared temperature sensor and an economical microphone are used in this research. The system is designed using Hardware Description Language and implemented on an FPGA board. With the help of simulation and experimental results, the functionality of the system is verified. In our future endeavor, efforts will be made to calibrate our system and make it appropriate for using in a crib.

References

[1] 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.

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

[3] D. Field, *Principles and practice of high dependency nursing field,* 2nd Ed. Oxford, United Kingdom: Bailliere Tindall, 2006.

[4] *Mimo Baby.* [Online]. Available: http://mimobaby.com/

[5] *Healthdyne 970 Smart Apnea Monitor.* [Online]. Available : http://www.dremed.com/irsrental/product_info.php/cPath/412_413/products_id/9336,

[6] J. C. Lin, "Noninvasive microwave measurement of respiration," Proc.IEEE, vol. 63, no. 10, pp. 1530–1530, Oct. 1975.

[7] [D. D. Mawhinney, Noninvasive Heart Rate Monitor No. RCAPRRL-83-CR-13. RCA LABS Princeton NJ, 1983.

[8] E. F.Greneker, "Radar sensing of heartbeat and respiration at a distance with applications of the technology," in Radar Systems Conf. (RADAR 97), Jan. 1997, pp. 150–154.

[9] C. Li, V. M. Lubecke, O. Boric-Lubecke, and J. Lin, "A review on recent advances in doppler radar sensors for noncontact healthcare monitoring," IEEE Trans. Microw. Theory Tech., vol. 61, no. 5, pp. 2046–2059, May 2013.

[10] MLX90614 I2C Infrared Thermometer. [Online]. Available : http://mbed.org/users/4180_1/notebook/mlx90614-i2c-infrared-thermometer

[11] System Management Bus (SMBus) Specification. [Online]. Available: http://smbus.org/specs/smbus20.pdf

[12] W. W. Peterson, "Cyclic Codes for Error Detection" *Proceding of the IRE,* vol .49, pp. 228-235. Jan, 1961

[13] P. P. Chu, "Audio codec controller," in *Embedded Sopc Design with Nios II Processor and VHDL Examples.* Hoboken, New Jersey : Wiley, 2011, pp 511-525.

[14] B. Berg. (2011, June.). Detection of Breathing and Infant Sleep Apnea.[Online].Available: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1041&context=cpesp

[APPENDIX]

UPDATE MODULE

A. *Update module's signal and data lines*

The module's input signals are the 15-bit data from 8 different sensors and "busy" signal from the Xbee driver. The output signal consist of "start" signal and 18-bit "cmd" signal whose first 3 bits indicate the address of the sensor and last 15 bits are the sensor's new data.

The incoming 15-bit data of the sensors will go to 8 different update_checker module and an 8-to-1 multiplexer with output chosen by s_sig signal generated by the controller. The 15-bit output of the multiplexer will become the last 15 bits of "cmd" output.

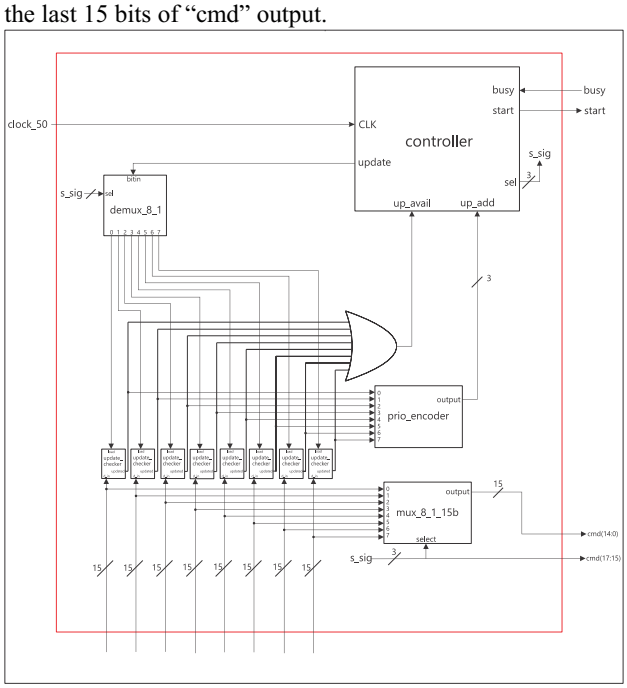


Figure 24. Update module

Each update_checker module contains a buffer (initialized to 0) used to store the previous data and a comparator to compare the new data with the old data. If they are different, the output "update" will be set to high. The module also has an input called "load". At positive edge of load signal, the buffer inside the module will be updated with the new data.

All the "update" output of update_checker will go through an OR gate and go to the up_avail port of the controller to tell it that there is updated information.

They will also go to a priority decoder to get the address of the update with highest priority and this address will go to the up_add port of the controller.

B. *Update module's controller*

The controller has 3 states called "idle", "communicate", and "waiting"

In idle state, if up_avail signal is 1 (new update available), the controller will check Xbee driver's "busy" signal to see if the Xbee is busy or not. If Xbee is busy, the controller will go to "waiting" state where it will wait until "busy" signal become 0 to go to "communicate" state, otherwise it will go to "communicate" state directly.

In "communicate" state, it will load up_add value to an internal buffer "sel" and output that value to s_sig signal. It will then send "update" signal and use s_sig signal to update the value of the buffer of the update with highest priority. The s_sig is also used as select bit for the 8-to-1 mux and as the 3-bit header of the "cmd" output. The controller will then generate a "start" pulse to tell the Xbee driver to get the data from cmd port. Finally, the controller will return to idle state.

C. *Simulation*

In this simulation, we will consider 3 different sensors. At the start, each of the sensor has value different from 0, since sensor0 has the highest priority, the first command sent to the Xbee is "000 000 0000 0000 0000 0001" with the first 3 bits "000" for sensor0. Ater the busy signal return to 0, it will start sending the next command, since sensor0 is now updated, the update with highest priority is now sensor1. The second command sent is "001 010 1010 1010 1010" with the first 3 bits "001" for sensor1. Because the value for sensor0 was changed while the second command was being sent, the next command sent was not for sensor2 since sensor0 has higher priority, the third command sent is "000 111 0000 1111 0000". After the third command is sent, the only new data is sensor2 so the next command sent is "010 101 0101 0101 0101". After all data are updated, the update module return to idle state to wait for new update.

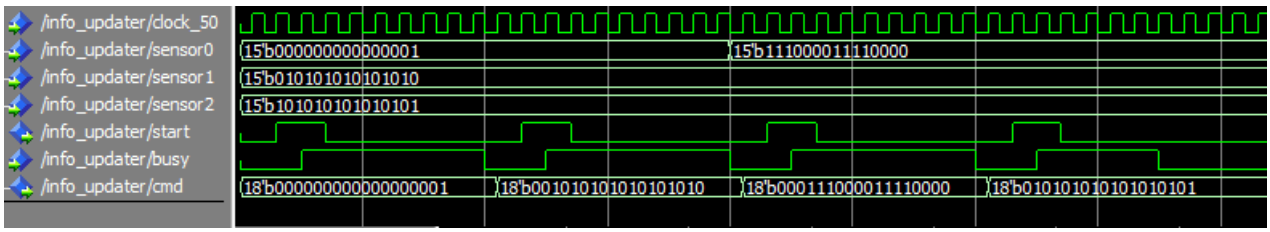


Figure 24. Simulation of Update module