International Journal of Innovation and Applied Studies
ISSN 2028-9324 Vol. 31 No. 3 Dec. 2020, pp. 587-592
© 2020 Innovative Space of Scientific Research Journals
http://www.ijias.issr-journals.org/

FPGA Based on airflow Noise Cancelling of mechanical ventilation

Khalifa Elmansouri[1], Mohamed Abouzahir[2], and Aziz Et-Tahir[2]

[1]UM6SS, ESGB, Casablanca, Morocco

[2]High School of Technology Mohamed V University, Rabat, Morocco

Copyright © 2020 ISSR Journals. This is an open access article distributed under the ***Creative Commons Attribution License***, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

ABSTRACT: The traditional de-noising method used by classical hardware equipments can't achieve successful de-noising effect and the software-only method never meets a high real time capability. Based on the Undecimated Wavelet Transform (UWT) which is an effective technique for de-noising signals corrupted by non-stationary noises, we propose implementing the UWT method on the field programmable gate array (FPGA) to realize a digital electronic circuit to de-noise the airflow of mechanical ventilation. The experiment results obtained was done regarding signal to noise ratio (SNR) and the requirement of real-time signal processing.

KEYWORDS: FPGA, Wavelet decomposition, LabVIEW, Computer hardware.

1 INTRODUCTION

Mechanical ventilation of the respiratory system is a lifesaving practice in patients, which are supporting many respiratory failures. However, the majority of critically ill patients in intensive therapy units (ITU) spend some time with their ventilated lungs by a mechanical ventilator, which uses traditional removing noise techniques; the embedded circuit and available technology provide the possibility to design a flexible system with an improved filtering algorithm.

Tiny embedded systems have not been an ideal outfit for high performance computing due to their constrained resources. Limitations in processing power, battery life, communication bandwidth and memory constrain the applicability of existing complex medical analysis algorithms such as the undecimated wavelet transform which is redundant, linear, shift invariant, more robust and less sensitive to noise.

Several works have been reported in the literature to reduce the processing power and memory requirement for computing Discrete Wavelet Transform (DWT) on limited resource platforms [1], [2]. For example, scheduling methods have been proposed by using a moving block to compute DWT for an entire image [2]. An overlap between two consecutive windows is demanded in many of these block processing methods. The amount of computations increases depending on the amount of the overlap. Also, there are several successful implementation of ASIC such as in [3], [4]. For example, wavelet codec based on FPGA which designed by Elghamery, wavelet transform coprocessor based on FPGA which given by Nibouche and so on. Most of these projects demonstrated above are successful in some conditions and it is pertinence for some products but it lack of application.

In this paper, we have proposed an improved airflow de-noising application based on UWT implemented on FPGA. The paper is organized as follows: section II deals with theoretical overview regarding the UWT algorithm, mechanical ventilation and the processing module used for implementation; section III details the FPGA implementation; section IV presents the implementation results and section V contains the conclusions of the paper.

2 Theoretical Overview

2.1 Undecimated Wavelet Transform

Continuous and discrete are the two main types of wavelet transform. Computer programs use the DWT because of its discrete nature. The main drawback of DWT is not translation invariant. Translation of the original signal leads to different wavelet coefficients. The UWT is used to overcome this and to get more comprehensive feature of the analyzed signal. The idea behind UWT is that it does not decimate the signal. Thus, it produces more accurate information for the frequency localization.

The UWT is redundant, linear, shift invariant, more robust and less sensitive to noise, because this method involves finding zero-crossings in the multi-scale UWT coefficients. This method first finds zero-crossings among the coefficients with coarse resolution and then finds zero-crossings among the coefficients with finer resolution. Finding zero-crosses among the coefficients with coarse resolution enables to remove noise from a signal efficiently. Finding zero-crossings among the coefficients with finer resolution improves the precision with which you can find peak locations. Hence, it gives a better approximation than DWT. These properties provide the UWT to realize using a recursive algorithm. Fig. 1 shows the computation of the UWT where d_n and a_n are called the detail ant the approximation coefficients of the UWT, respectively. The filter H and G are the standard low-pass and hight-pass wavelet filters, respectively.

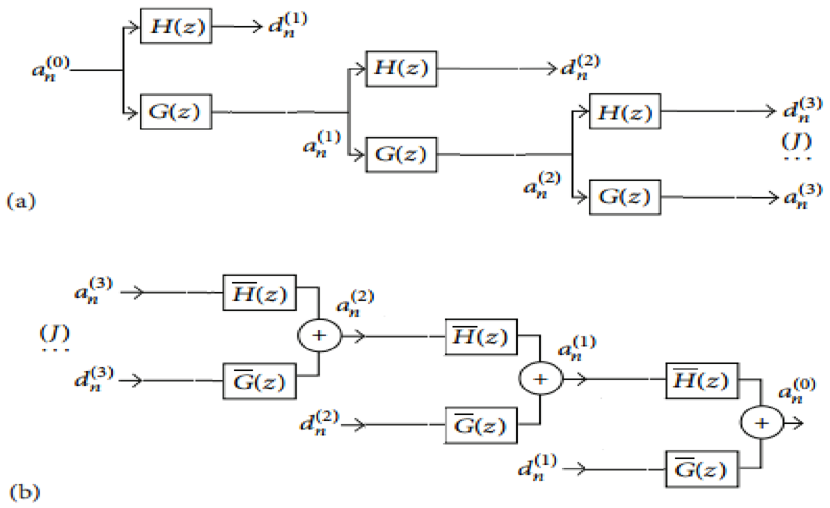

Fig. 1. ***Undecimated wavelet transform, (a) analysis filter bank and (b) reconstruction filter bank***

To further remove the noise from the airflow, UWT is employed to decompose the signal at various levels. For an N level UWT, we have N detail sub-bands (W). All the W sub-bands are used for de-noising purpose. An adaptive threshold selection using principle of Stein's Unbiased Risk Estimator [5] is used to calculate the threshold value. Then, soft thresholding [6] is used to remove the noise. Finally, the noise-free airflow is obtained by taking the inverse UWT.

2.2 The Pressure Controlled Ventilation Model

The pressure controlled ventilation is characterized by signals shown in and 3, the curve of pressure explains the setting variables in PCV device such as:

- Inspiratory pressure (IP),
- Inspiratory time (T_{in}),
- Expiratory time (T_{ex}).
- Constant time of ventilator (τ).

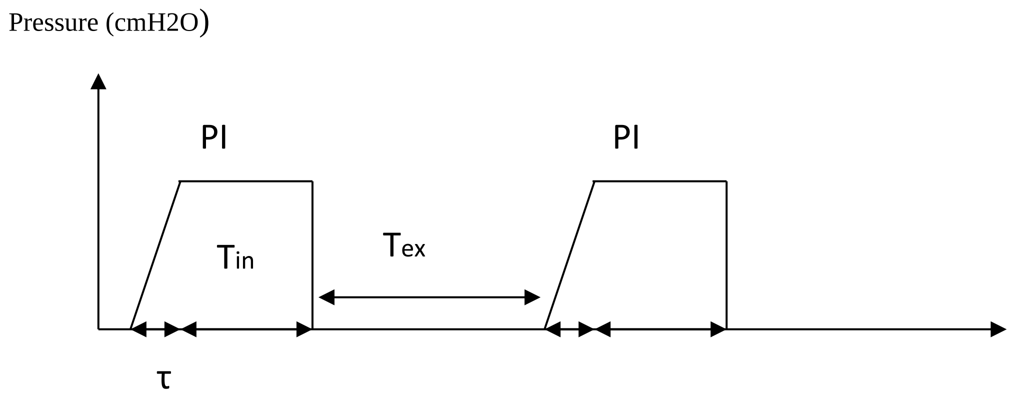

Fig. 2. Typical waveform of pressure support for PCV

The parameters values which are setting in this work are:

- IP = 25 cmH_2O.
- τ = 0.2s.
- T_{in} =1s, T_{ex}=2s.

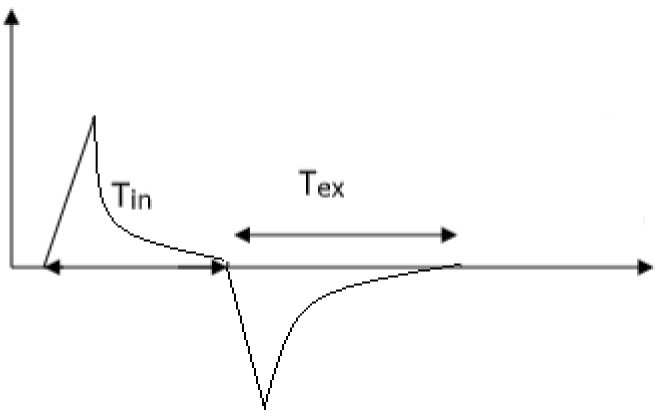

Fig. 3. Typical waveform of air flow measurement

2.3 The Used Hardware

The cRIO-9104 reconfigurable embedded chassis [7] is used to implement the UWT and to design custom hardware. The cRIO-9104 includes an embedded controller which offers powerful stand-alone embedded execution for deterministic LabVIEW Real-Time applications. The embedded chassis is at the center of the CompactRIO system because it contains the reconfigurable I/O FPGA core; also it has the following characteristics:

- -40 to 70 °C operating range
- Automatically synthesize custom control and signal processing circuitry using LabVIEW
- 3M gate reconfigurable I/O (RIO) FPGA core for ultimate processing power
- DIN-rail mounting options
- 8-slot reconfigurable embedded chassis accepts any CompactRIO I/O module

3 FPGA IMPLEMENTATION

3.1 GRAPHICAL PROGRAMMING

Our implementation was performed using the LabVIEW FPGA module which is a toolkit allowing FPGA implementation to be done graphically without requiring any knowledge of hardware description languages such as VHDL or Verilog. The LabVIEW environment incorporates FPGA parallelism inherent to FPGAs. Additional features of the LabVIEW FPGA module are discussed in reference [8]. Our LabVIEW FPGA implementation consisted of two components: a host side and a target side. These components are described below.

3.1.1 HOST SIDE

Initially all the required parameters for the computation of the wavelet transform are set on the user interactive Front Panel of the host VI (LabVIEW programs are called VIs or Virtual Instruments). These parameters can be changed on-the-fly while the program is running. Parameters include wavelet filter type, number of decomposition levels and number of samples. An input signal gets converted to fixed-point as FPGAs only support fixed-point operations. Wavelet coefficients are computed in fixed point. Fig. 2 shows a portion of the Block Diagram (graphical code) section of the host VI.

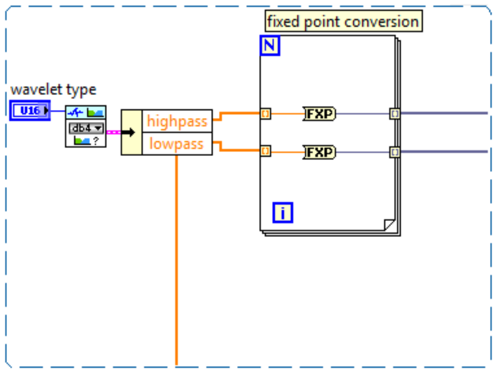

Fig. 4. Block Diagram of a section of the host VI

3.1.2 TARGET SIDE

DMA (Direct Memory Access) FIFOs were used to transfer data between the host and the target. The details of the transfer process can be found in reference [8]. Interrupts were used to synchronize the data transfer between the host and the target. All the operations on the target side were done in fixed-point. The implementation was optimized by the efficient use of memory elements and by parallel processing pipelining as explained in the previous section.

3.2 FLOATING-POINT TO FIXED-POINT CONVERSION

The algorithm was initially implemented using floating-point data type to check for the correctness of the outcome before implementing it on an actual FPGA processor using fixed-point data type. Depending upon the precision of the fixed-point implementation, the mean squared error of the reconstructed signal varied. In addition, scaling was done at every decomposition level to control any possible overflow. FPGAs allow variables to have flexible word sizes without the need for them to be a multiple of bytes. Consequently, this led to a more optimized fixed-point implementation. It is worth mentioning that the relationship between word size and processing time is not linearly proportional on an FPGA processor, hence increasing the word size did not increase the processing time significantly in our implementation.

3.3 MEMORY MANAGEMENT

Since accessing memory on FPGA is computationally costly, when a large amount of memory is utilized, the real-time operation suffers. Thus, in our implementation, memory reading and writing operations were reduced by the optimal use of memory items as follows: whenever an element was read from memory, our design was optimized such that all the operations on that element were completed before writing it back to memory. Also, for small data storage, flip-flop FIFOs were used instead of memory items in order to achieve faster access to data.

3.4 Stage Pipelining

To further make use of the FPGA parallelism, each stage of the UWT computation was pipelined. Fig. 3 illustrates how the stage pipelining was done by computing the coefficients in stages. Memory buffers were dedicated to each stage.

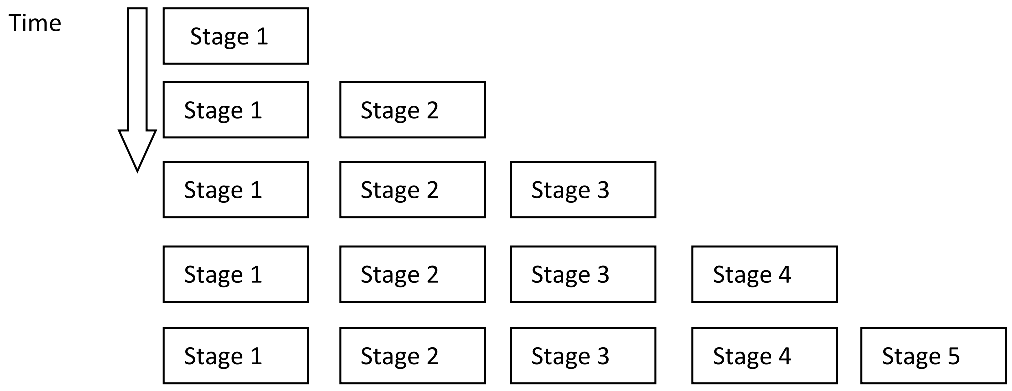

Fig. 5. Illustration of stage pipelining for 5 stages

3.5 Filter Pipelining

Computation of wavelet coefficients involves filtering. Filtering consists of multiply and sum operations. Instead of doing these operations sequentially, registers were used to pipeline these operations. Fig. 4 illustrates this procedure. As can be seen from this figure, loop execution time is decreased by reducing the length of the critical path in each loop iteration.

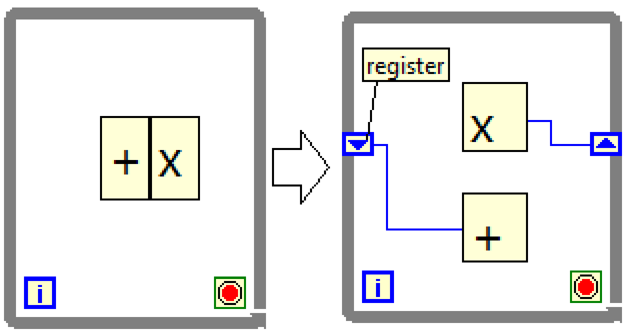

Fig. 6. Illustration of filter pipelining

4 Real-Time Implementation Results

We write the configuration parameters, which include frame data length, filter coefficients, wavelet transform layers and the threshold value. The data frame length is 128. Length is 32. Wavelet transform layer is 6. The first layer decomposition of wavelet coefficients processing threshold is estimated by Stein's Unbiased Risk Estimator. The decomposition filter was considered to be of 4th order here.

The input signals were compromised by a white noise with different SNR. After reading they were sent to a FIFO memory of the FPGA chassis where they were processed by the hardware architecture. The flip-flop FIFO memories have the length set to 1023 samples. The processing and reading of the signals is done sample-by sample from the FIFO's.

The FPGA hardware usage depends on the number of bits describing the process accuracy. Thus, if the hardware usage is the priority then a low pass filter and high pass filter length is recommended or the signals accuracy must be decreased.

The quality of the process was measured in terms of the de-noising signal's SNR and the convergence measured in seconds. The SNR for the output signal is evaluated as:

$$SNR = \frac{RMS(x)}{RMS(y-x)}$$

Where RMS (x) and RMS (y-x) are the root mean squares of the pure input signal throughout the de-noising process and the noise at the output. The output noise is estimated as a difference between the de-noised and the pure input signals (y-x).

Fig. 5 and table 1, show the tracking ability of the process.

Table 1. Quality estimation of the process

SNR [db]		**Convergence (s)**
Input	**Output**	
-5	1.9	0.035
-10	-3.6	0.779
10	30	0.0061

5 CONCLUSION

The contribution in this paper involves the real-time FPGA implementation of the UWT approach without using any hardware description language. This has been achieved by using the FPGA graphical programming feature of LabVIEW.

The performance of the UWT algorithm implemented by hardware is comprehensively analyzed in terms of convergence performance and filtered signal's SNR.

REFERENCES

[1] A. Shahbahrami, "Improving the performance of 2D discrete wavelet transform using data-level parallelism," Proc. of IEEE Int. Conf. on High Performance Computing and Simulation, pp. 362-368, 2011.

[2] S. Salehi and R. Amirfattahi, "A block-based 2D discrete wavelet transform structure with new scan method for overlapped sections," Proc. of IEEE Middle East Conf. on Biomedical Engineering, pp. 126-129, 2011.

[3] Nevin A Elghamery, S.E.-D.Habib (2000) An efficient FPGA implementation of a wavelet coder/decoder. Microelectronics, ICM, Proceedings of the 12th International Conference 12: 269-272.

[4] Tian Xin, Tan Yihua, Tian Jinwen, "New design for 9/7-tap wavelet filter and its VLSI implementation. Jisuanji Xuebao/Chinese Journal of Computers," Vol.31, no.3, pp.411-418. Mar. 2008.

[5] CM Stein, "Estimation of the Mean of a Multivariate Normal Distribution," The Annals of Statistics, Vol. 9, pp. 1135-51, 1981.

[6] DL Donoho, "De-Noising by Soft-Thresholding," IEEE Transactions on Information Theory, Vol. 41, pp. 613-27, 1995.

[7] www.ni.com.

[8] N. Kehtarnavaz and S. Mahotra, Digital Signal Processing Laboratory: LabVIEW-Based FPGA Implementation. Brown Walker Press, 2010.