COMPRESSION SYSTEM FOR THE PHONOCARDIOGRAPHIC SIGNAL

F. J. Toledo-Moreo, A. Legaz-Cano, J. J. Martínez-Álvarez J. Martínez-Alajarín, R. Ruiz-Merino

Dpto. Electrónica y Tecnología de Computadoras, Universidad Politécnica de Cartagena
Cuartel Antiguones, Pl. Hospital, 1, 30202 Cartagena Spain
email: javier.toledo@upct.es

ABSTRACT

An FPGA-based approach is proposed for implementing a compression system developed specifically for the signal of phonocardiogram. The compression method offers better rate and distorsion than standard audio compression techniques. Both the algorithm and the details on the solutions adopted for its implementation are presented in this paper.

1. INTRODUCTION

In last few years, the auscultation of the heart has regained the interest lost in past decades in favour of more accurate and expensive techniques. The automated analysis of the phonocardiogram (PCG) has been rediscovered as a useful, fast and low cost technique to diagnose valvular diseases. In this context, we are working on the development of a set of tools for the follow-up of patients suffering from cardiopathies. Like in any telemedicine application, an efficient transmission is required for ubiquitous monitoring. With this aim, we have developed a specific algorithm to compress the PCG signal. No PCG compression works have been found in the literature to the authors' knowledge. Since PCG is the graphical representation of heart sounds and murmurs, standard audio compression techniques could be used. However, they are mainly focused on music or speech, and their performance in PCG compression is low.

In this paper, an FPGA-based embedded system for the efficient compression and transmission of the PCG signal is proposed. It is a mixed hardware/software approach, based on the MicroBlaze processor. The system digitizes the PCG signal from a stethoscope, compresses it and sends it to a remote PC through a wireless serial communication. In the PC, the PCG is decompressed and analyzed by experts.

Next, the compression algorithm is presented in Section 2. In Sections 3 and 4, the two main processing blocks are described: the Discrete Wavelet Transform module and the MicroBlaze-based system. Details about the prototype and results are reported in Section 5. Finally, conclusions are drawn in Section 6.

This work has been supported by Ministerio de Ciencia y Tecnología of Spain, under grant TIN2006-15460-C04-04.

2. COMPRESSION ALGORITHM

The proposed algorithm is a lossy compression method applied to non-overlapping blocks of N samples of the PCG signal. Each block is compressed independently from the others. It is adapted from the algorithm proposed in [1], and consists of the following four steps:

- Decomposition of the PCG signal using the WT
- Dynamic thresholding of the wavelet coefficients
- Compression of the wavelet coefficient vector using zero removal
- Compression of the significance map using Run-Length and Huffman enconding.

First, the Wavelet Transform (WT) decomposes a signal block in $J+1$ subbands, where J is the number of decomposition levels or octaves of the WT. The WT represents the information of the signal block as a vector WC of wavelet coefficients extracted from each subband, defined as:

$$\mathrm{WC} = (\mathrm{CA}_J, \mathrm{CD}_J, \mathrm{CD}_{J-1}, \ldots, \mathrm{CD}_1) \qquad (1)$$

where CA_J is the approximation coefficient vector in the level J, and CD_j are the detail coefficient vectors in the levels j, with $j = 1, 2, \ldots, J$. The length of each vector is given by $N/2^{j-1}$, and the total length of WC is N, the same as the original block signal.

In the second step, the WC coefficient vector is thresholded using an iterative algorithm which looks for the best compression rate, while ensuring that a desired error percentage, fixed a priori, is achieved. A modified version of the Percent Root-mean-square Difference (PRD_m) has been used as measure of the error:

$$\mathrm{PRD}_m = \sqrt{\frac{\sum_{i=1}^{N}(wc_i - \widehat{wc}_i)^2}{\sum_{i=1}^{N} wc_i^2}} \times 100 \qquad (2)$$

where wc_i are the N elements of the WC vector given by the wavelet transform and $\widehat{wc}_i$ are the elements after thresholding.

The algorithm modifies the threshold value to reach the target value specified for the PRD_m. For the first iteration, the threshold is fixed to a percentage of the maximum absolute value of WC. This is its highest value in the algorithm,

1-4244-1060-6/07/$25.00 ©2007 IEEE.

and yields to the highest PRD_m error and the highest compression rate. In the next iterations, the threshold value is modified until the desired PRD_m is reached.

This step gives two outputs: the thresholded coefficient vector (TC), which is the same as WC but with the coefficients with absolute value less or equal to the threshold made zero; and the significance map (SM), which is a binary vector that indicates with '1' the position of nonzeroes in WC. The third step of the compression algorithm simply consists of removing zero coefficients from TC. Finally, with the aim of further reducing the size, the SM vector is compressed using the Run-Length and the Huffman encoding techniques.

The tests performed during the development of the algorithm revealed that the optimum value for the size N of a block signal is 4096 samples, and that the most suitable wavelet transform is a 4-octave Daubechies order 10. So, this configuration has been the implemented one. More details on the compression method like, for example, the relationship between the compression rate and the percentage of the signal energy retained after compression, or a comparison with premier standard audio compression techniques, can be found in [1].

3. IMPLEMENTATION OF THE DWT

The hardware implementation of the Discrete Wavelet Transform (DWT) is based on Mallat's work [2], who demonstrated that the wavelet representation of a signal can be computed by convoluting the signal with a pyramidal structure of quadrature mirror filters. This algorithm leads to a direct implementation with J two-channel filter banks in cascade performing low-pass and high-pass filtering operations. However, the direct implementation derived from Mallat algorithm is not efficient, since the computational load is not well shared out among the functional units due to the decimation between stages. To optimize the hardware implementation, two different approaches have been evaluated: folding and digit-serial.

The folding approach [3] allows implementing any J-octave wavelet in just one time-multiplexed filter bank. Since it is not necessary to compute the data that will be discarded later in the decimation process, the pyramidal algorithm can be reformulated [4], and, with the right multiplexing of inputs and outputs, it is possible to compute the DWT with one filter bank and some registers where to store the data required by the filter bank.

For this folded approach, different structures and design parameters have been analyzed in this work, pursuing the most efficient implementation in the PCG signal processing application. Thus, for the two-channel filter bank three configurations have been considered: two parallel FIR filters, two MAC filters and just one MAC filter (multiplexed to compute the high-pass and the low-pass filtering). In addition, multiplication operations have been implemented using embedded multipliers and general purpose logic resources, and, in this last case, with parallel and fully-serial structure.

Digit-serial computation is based on the division of n-bit data into k m-bit digits, with $m \in (1, n)$ and $k = n/m$. Computations are carried out on one digit at a time, and therefore k cycles are required to operate on the data. Regarding the DWT, this technique allows implementing it with so many stages as octaves, but each one with a different digit size $m = n/2^{j-1}$, with $j = 1, 2, \ldots J$. This considerably reduces the required resources within a direct implementation, and makes it feasible to achieve a 100% hardware utilization. Again, different alternatives have been considered: the filters have been designed with MAC structure and based on distributed arithmetic; polyphase decomposition has also been applied to the filter banks.

For both approaches and their design alternatives their resources and timing features have been evaluated. Figure 1 shows the influence of these alternatives on the resources required and on the maximum bandwidth of the input signal, when implementing a 3-octave Daubechies order 4 DWT with 16-bit input data resolution. The impact of the number of octaves and the data size have been also evaluated. Detailed information and conclusions can be found in [5].

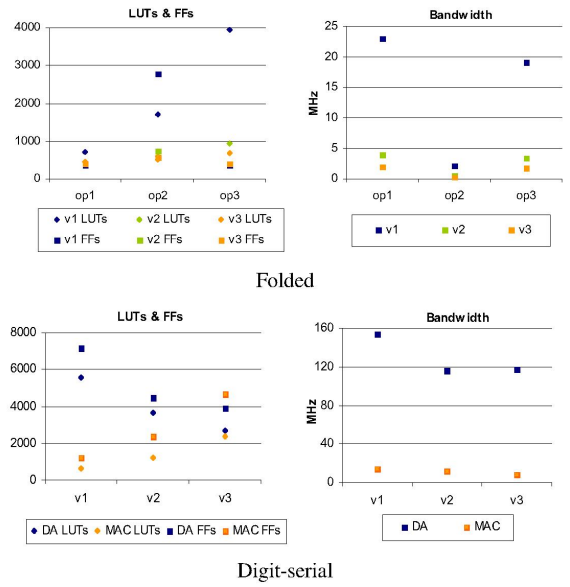
Looking at Fig. 1, the a priori more interesting options for our application are the folded with one MAC filter and the digit-serial with distributed arithmetic and polyphase decomposition. Both of them have been implemented to design the 4-octave Daubechies order 10 DWT used in the compression method. According to the results shown in the Table 1, and bearing in mind the low bandwidth of the PCG signal, the folded architecture with just one MAC FIR filter has been considered the most suitable choice, since it achieves real time processing of the PCG signal reducing to the maximum the required resources. The FIR filter has been implemented with an embedded multiplier, which is successively multiplexed to perform the high-pass and the low-pass filtering of each filter bank. A 16-bit data resolution has been adopted for inputs and outputs; internally, the DWT module works with up to 32-bit data.

4. MICROBLAZE SUBSYSTEM

Once the wavelet coefficients have been generated, they are thresholded using a dynamic algorithm. Then, zero coefficients are removed and RLE and Huffman encoding techniques are applied. In the proposed embedded system, all of

Table 1. Resources used in the implementation of the DWT in the compression algorithm on a Xilinx Virtex4 LX25.

	4-input LUTs	FFs	Slices	DSP48s
Folded 1 MAC with embedded mult.	2312	1368	1183	1
Digit-serialDA with polyphase decomp.	5170	7601	4163	0



Folded

v1:	2 parallel filters	op1:	embedded multipliers
v2:	1 parallel filter	op2:	serial multipliers
v3:	1 MAC filter	op3:	parallel multipliers

Digit-serial

v1: digit size equals to data size
v2: digit size function on the octave
v3: digit size function on the octave with polyphase decomposition

Fig. 1. Evaluation of the different approaches and alternatives considered for the implementation of a 3-octave Daubechies db4 DWT on Xilinx Virtex4 FPGAs.

these steps are performed in a software application running in the MicroBlaze soft processor. Therefore, a MicroBlaze-based system has been implemented. The overall architecture is shown in Fig. 2.

As can be seen in Fig. 2, the communication between MicroBlaze and the DWT module has been designed with a data buffer and an interrupt signal. In accordance with the block size and the data resolution, the buffer is a 4K×16 memory, implemented with BlockRAM. It is connected to MicroBlaze through an FSL channel. The DWT module controls the writing of each wavelet coefficient in its suitable buffer address. When a 4096-sample PCG signal block has been transformed into wavelet coefficients and stored in the buffer, the DWT module activates the interrupt to MicroBlaze, and begins the acquisition and transformation of a new 4096-sample block. In the software application, the interrupt handler manages the execution of the steps of the compression algorithm and the transmission of the output data. In the first place, data are read from the buffer and written in an internal memory. This avoids data overwriting with the wavelet coefficients corresponding to a new block. Like the buffer, this internal memory is a 4K×16 memory implemented with BlockRAM. As the data are read from

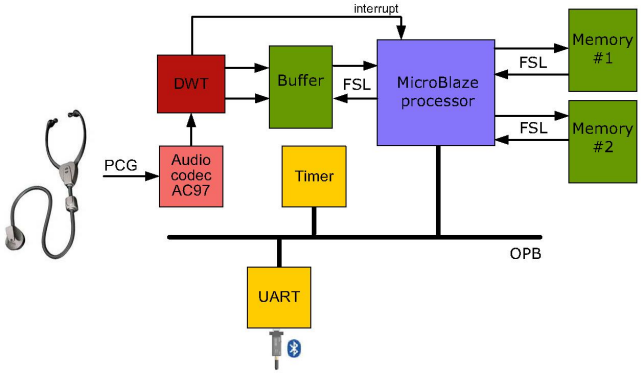


Fig. 2. Overall architecture of the proposed system.

the buffer to be written in the internal memory, the maximum absolute value is found. It is required to fix the initial threshold. Then, the thresholding process starts: in each iteration the internal memory is read, the coefficients with absolute value less or equal to the threshold are zeroed and the PRD_m is calculated. The process iterates until the target PRD_m is reached or a timer warns about the time available to complete the compression. To speed up the reading from the memory, the FSL bus has been the adopted communication channel.

Once the data have been thresholded, the zero values are removed and the non-zero values are saved. Next, the significance map, derived from the zero removing, is compressed using Run-Length and Huffman encoding. Both of these techniques have been also implemented in the software application running in MicroBlaze. Due to the size and amount of data involved and generated in these steps, a second internal memory, connected to MicroBlaze through an FSL channel, has been included. This option has been preferred to the definition of large variables in the software application.

Finally, the compression resulting data are sent to a remote PC via a bluetooth connection. The communication has been implemented using a UART peripheral connected to MicroBlaze and a Promi-SD module, from Initium, as RS232/bluetooth interface. The UART has been configured to operate at 230400 bauds, the maximum data rate achieved by the Promi-SD module. A small header is included in the data packet in order to implement a simple communication protocol. It contains information that allows the software in the remote PC to manage correctly the received data and to rebuild the PCG from the successive compressed block.

All of these steps are executed while new PCG data are being acquired, transformed by the DWT and stored in the buffer. Therefore, an N-sample block of the PCG signal must be compressed and transmitted before the following is ready to be compressed. Since the signal is sampled at 8 KHz and each block is made of 4096 samples, the total available time (the time between consecutive interrupts from the DWT module) is 512 ms. To avoid exceeding that time,

Table 2. Summary of resources used.

	DWT & logic	MicroBlaze	Total %
Slices	1637	2432	38%
Flip flops	2163	1848	19%
4 input LUTs	2681	3812	30%
DSP48	1	7	17%
BlockRAMs	0	28	39%

and bearing in mind that the compression is independent for each block and hence it can take different time in each one, the timer included in the block diagram of Fig. 2 is used.

Finally, it must be remarked that the implementation of the DWT as a specific coprocessor, instead of in the software application, frees the processor from computing the wavelet coefficients. It allows the execution of the dynamic thresholding algorithm, and, briefly, makes it feasible the real-time compression of non-overlapping blocks without losing any data between blocks.

5. DETAILS OF THE PROTOTYPE AND RESULTS

A first prototype has been built using a Xilinx Virtex-4 FPGA and Avnet boards. The FPGA is the Virtex-4 LX25, populated in the Virtex-4 Evaluation kit from Avnet. This board includes the RS232 connection where the Promi-SD module is plugged in. To acquire the signal from the stethoscope, the Philips UCB1400 stereo 20-bit audio codec has been used. It is available in the Audio/Video module, also from Avnet. This audio codec is initialized and managed with a controller implemented on the same FPGA. The FPGA resources occupied by the design are summarized in Table 2. The percentages refer to the resources available in the mentioned FPGA (21504 4-inputs LUTs, 21504 flip-flops, 10752 slices, 72 BRAMs and 48 DSP48s).

MicroBlaze works at 100MHz, its memory size is 32 KB and its architecture includes a Floating Point Unit. The mentioned clock frequency let the functions that make up the algorithm take the times indicated in Table 3, in function of the compression rate. These results show that a lower compression rate requires more iterations in the thresholding, and that the Huffman encoding is more complicated and longer. Besides, as the compression rate decreases, fewer wavelet coefficients are thresholded and the Huffman code is larger, which implies that the transmision takes more time. On the other hand, the zero removing and the RLE encoding hardly suffer modifications.

The algorithm was originally developed in Matlab®, using real PCG records in WAV format (16 bits, 8KHz). The Table 3 indicates the times required to compress a signal block in Matlab using a Pentium4 at 2.8 GHz and in the proposed system. Due to the different internal data resolutions and to small modifications in the original algorithm motivated for increasing the efficiency in the software running in MicroBlaze, the output vectors resulting in both approaches are not always identical. Nevertheless, the compression rates do behave exactly equally.

Table 3. Time required for the compression algorithm (ms).

	Compression Rate			
	10	15	20	25
Steps				
Thresholding	105.4	96.4	78.9	71.5
Zero removing	2.54	2.53	2.50	2.50
Run-length encoding	0.95	0.94	0.93	0.93
Huffman encoding	4.84	4.03	2.94	2.48
Transmission	33.2	23.9	17.4	14.3
Complete algorithm				
FPGA-based	146.9	127.8	102.7	91.7
Matlab	205.2	212.1	195.7	192.9

6. CONCLUSION

The implementation of a novel compression method designed for the PCG signal is presented. An FPGA device has been used as hardware platform to design a MicroBlaze-based solution which can compress and transmit in real time the PCG signal. For the implementation of the DWT, which has been included as specific coprocessor to MicroBlaze, different architectures, structures and design parameters have been evaluated, looking for the best solution in the described PCG signal processing application.

The use of an FPGA-based approach has made it possible the design of a low cost and small size embedded system which can compress and transmit the PCG signal in real time, and faster than the PC solution initially developed.

Future works are focused on developing an "smart stethoscope". It will be an FPGA-based embedded system where the PCG signal is not only compressed and transmited but also processed, with the purpose of being helpful for medical diagnosis of people suffering from cardiopathies.

7. REFERENCES

[1] J. Martínez-Alajarín and R. Ruiz-Merino, "Wavelet and wavelet packet compression of phonocardiograms," *Electronics Letters*, vol. 40, no. 17, pp. 1040–1041, Aug. 2004.

[2] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[3] K. Parhi, *VLSI digital signal processing systems: design and implementation.* Wiley, 1999.

[4] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," *IEEE Trans. on Signal Processing*, vol. 42, no. 3, pp. 673–676, 1994.

[5] F. J. Toledo, M. B. García, J. J. Martínez, and J. M. Ferrández, "Evaluación de arquitecturas para implementación eficiente sobre FPGA de la DWT," in *Proc. 6th Jornadas de Computación Reconfigurable y Aplicaciones*, Sept. 2006, pp. 173–178.