HARDWARE IMPLEMENTATION OF PAN & TOMPKINS QRS DETECTION ALGORITHM[1]

C. Pavlatos*, A. Dimopoulos*, G. Manis** and G. Papakonstantinou*

* National Technical University of Athens
Dept. of Electrical and Computer Engineering
Zografou 15773, Athens
Greece

** University of Ioannina
Dept. of Computer Science
P.O. Box 1186, Ioannina 45110
Greece

{pavlatos, alexdem, papakon}@cslab.ntua.gr, manis@cs.uoi.gr

Abstract : This paper presents a hardware implementation of the Pan and Tompkins QRS detection algorithm, described in Verilog HDL (Hardware Design Language). The generated source has been simulated for validation, synthesized and tested on a Xilinx FPGA (Field Programmable Gate Array) board using the European ST-T database. To the best of the authors' knowledge this is the first attempt for the hardware implementation of the Pan and Tompkins QRS detection algorithm, in reconfigurable FPGA boards. The generated hardware achieves a speed up of 250% compared to the software implementation. Given that and the vital importance of a fast and accurate QRS detection, the hardware implementation seems a promising approach.

Introduction

The QRS detection algorithm introduced by Pan and Tompkins [1] is the most widely used and often cited algorithm for the extraction of QRS complexes from electrocardiograms. The methodology followed is that the ECG is passed through a low-pass and a high-pass filter in order to remove noise from the signal. Then the filtered signal is passed through derivative, squaring and window integration phases. Finally, a thresholding technique is applied and the R-peaks are detected.

This work presents the hardware implementation of the Pan-Tompkins algorithm. The Verilog [2] hardware description language has been used. The building blocks and the actual implementation were designed, tested and evaluated using the ISE tool available from Xilinx [3].

The comparison of our hardware implementation with an equivalent software approach showed that the necessary clock cycles are significantly less for the hardware implementation.

[1] This work is co - funded by the European Social Fund and particularly the Program "Pened 2003".

The rest of the paper is structured as follows. The following section outlines some interesting QRS detection algorithms and describes in more detail the one proposed by Pan and Tompkins. The next chapter presents the general architecture of the proposed system, while implementation details are given in the next two chapters. The final section summarizes this work.

QRS Detection Algorithms

Several QRS detection algorithms have been proposed in the literature [22], [23]. Algorithms [4], [5] and [6] are based on the amplitude and the first derivative. In [4] a point is classified as QRS candidate when three consecutive points of the first derivative exceed a positive threshold (ascending slope) followed within the next 100ms by two consecutive points which exceed a negative threshold (descending slope). Fraden and Neuman [5] developed a QRS detection scheme where a threshold is calculated as a fraction of the peak value of the ECG. Gustafson [6] suggested that a point is a QRS peak candidate when the first derivative and the three next derivative values exceeds a threshold and the next two sample points have positive slope amplitude products.

Algorithms [7] and [8] are based on the first derivative only. In [7] the first derivative is calculated by a given formula and the slope threshold is calculated as a fraction of the maximum slope for the first derivative. In [8] the derivative and the derivative of the next three points should exceed the threshold.

Algorithms [9] and [10] are based on the first and second derivatives. Balda [9] suggested searching values exceeding the threshold in a weighted summation of the first and second derivative. Ahlstrom and Tompkins in [10] proposed that the absolute values of the first derivative are smoothed and added with the absolute values of the second derivative. Two thresholds are used, a primary and secondary one. A point is candidate for QRS peak point when the primary

ISSN: 1727-1983

threshold is exceeded and the secondary threshold is exceeded for the next six consecutive points.

Algorithms [11] and [12] are based on digital filters. Apart from the above referenced papers, a more detailed description of those ([4]-[12]) algorithms can be found in [13].

QRS detection algorithms have been proposed by our research group based on the length transformation and on syntactic methods [14], [24], [25], [26], [27], [28]. These algorithms calculate the length and energy signals of the ECG and identify peaks using variations and thresholds in these signals. Relative work which proposes alternative approaches to QRS detection includes [15-20].

Pan & Tompkins QRS detection algorithm

One of the most popular QRS detection algorithms, included in virtually all biomedical signal processing textbooks, is that introduced by Pan and Tompkins in [1]. An overview of the algorithm follows. Figure 1 shows a graphical representation of the basic steps of the algorithm.

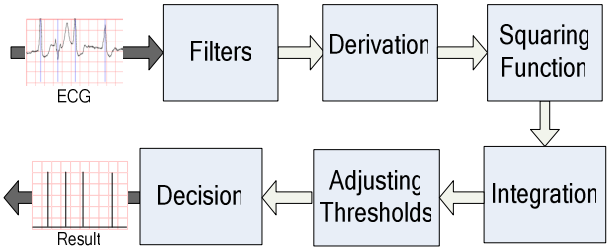

Figure 1: A graphical representation of the algorithm. The signal passes through filtering, derivation, squaring, and integration phases before thresholds are set and QRS complexes are detected

In the first step the algorithm passes the signal through a low pass and a high pass filter in order to reduce the influence of the muscle noise, the power line interference, the baseline wander and the T-wave interference.

The low-pass filter is described by the formula:

$$y(n) = 2y(n-1) - y(n-2) + x(n) - 2x(n-6) + x(n-12)$$

and the high-pass one is given by:

$$y(n) = y(n-1) - \frac{1}{32}x(n) + x(n-16) - x(n-17) + \frac{1}{32}x(n-32)$$

After filtering, the signal is differentiated to provide the QRS slope information using the following formula:

$$y(n) = \frac{1}{8}[2x(n) + x(n-1) - x(n-3) - 2x(n-4)]$$

Then the signal is squared point by point making all data point positive and emphasizing the higher frequencies.

$$y(n) = x^2(n)$$

After squaring, the algorithm performs sliding window integration in order to obtain waveform feature information.

$$y(n) = \frac{1}{N}[x(n-(N-1)) + x(n-(N-2)) + ... + x(n)]$$

where N is the size of the sliding window and depends on the sampling rate. For a sampling rate of 200 samples/sec the size of the window can be 30 samples.

A temporal location of the QRS is marked from the rising edge of the integrated waveform.

In the last step two thresholds are adjusted. The higher of the two thresholds identifies peaks of the signal. The lower threshold is used when no peak has been detected by the higher threshold in a certain time interval. In this case the algorithm has to search back in time for a lost peak. When a new peak is identified (as a local maximum – change of direction within a predefined time interval) then this peak is classified as a signal peak if it exceeds the high threshold (or the low threshold if we search back in time for a lost peak) or as a noise peak otherwise. In order to detect a QRS complex the integration waveform and the filtered signals are investigated and different values for the above thresholds are used.

To be identified as a QRS complex, a peak must be recognized as a QRS in both integration and filtered waveform.

The Architecture of the Implementation

The architecture of the implementation is shown in figure 2 and consists of seven modules and one memory unit. The module on the left is the control unit which is responsible for the generation of the control signals and coordinates all calculations. Each one of the six modules in the middle is responsible for a different stage of the algorithm as described in the previous section. Each one of those modules read values from the memory, perform the necessary computations and store the new values back in memory.

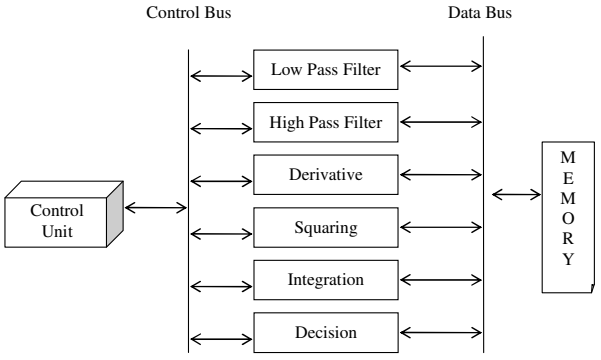

Figure 2: An overview of the architecture. The control unit coordinates the rest of the modules which perform calculations and interfere with the memory unit.

Even though the implementation of the algorithm in hardware seems a complicated task, a closer look at the algorithm reveals some similarities in the calculations performed by the first five stages since the filters, the differentiation and the integration units compute

ISSN: 1727-1983

difference equations. The general formula of these stages is:

$$y(n) = \sum_{i=1}^{m} a_i x(n-k_i) + b_i y(n-l_i)$$

Thus, those modules can be implemented based on a common model, which loads values from the memory, multiplies them with the appropriate factors, adds them, and stores the result back in memory.

Since the implementation philosophy for all units is the same, the common architecture is shown in figure 3.

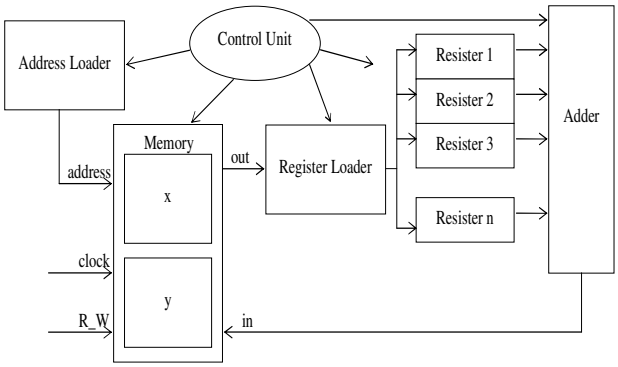

Figure 3: The architecture of the stages. Values are loaded from memory, multiplied with proper factors, added and stored back in memory

Without loss of generality we will describe the implementation of the low-pass filter. The low-pass filter is described by the formula:

y(n)=2y(n-1)-y(n-2)+x(n)-2x(n-6)+x(n-12)

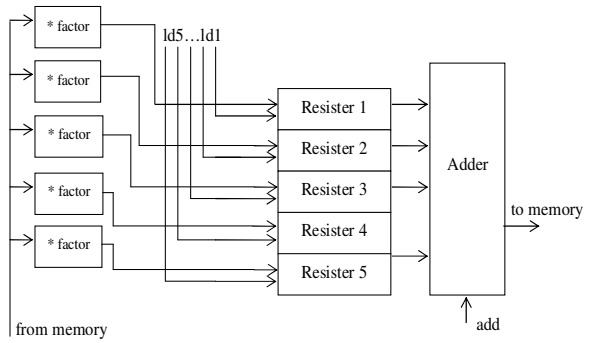

Figure 4: Register loader, registers and adder for the low-pass equation unit

A memory unit is used which stores, in the first half, the values of x. The second half is used for the produced values of y. The control unit is responsible for the creation of the necessary control signals which enable the basic units of the filter and synchronize the calculations. For example when the terms 2y(n-1), -y(n-2), x(n), -2x(n-6), and x(n-12) are available the control unit generates a control signal to enable the adder and produce y(n). The address loader is a complicated circuit which consists mainly from multiplexers and adders. It produces the addresses which store the values necessary to calculate y(n). For the low-pass filter these values are y(n-1), y(n-2), x(n), 2x(n-6), and x(n-12). In a similar way the register loader consists of multiplexers and arithmetic operations circuits. Under the instructions of the control unit, the loader multiplies with the appropriate factors the values read from the memory and loads them to the registers. Finally an adder adds the values of the registers and sends y(n) to the memory.

Table 1: Signals for the circuit of figure 4

time	value	ld5…ld1	add
t	2y(n-1)	00001	0
t+1	y(n-2)	00010	0
t+2	x(n)	00100	0
t+3	2x(n-6)	01000	0
t+4	x(n-12)	10000	0
t+5		00000	1

Let us select and present in more detail a subset of the whole circuit, the register loader, the registers and the adder. The architecture is shown in figure 4. The memory, under the supervision of the control unit, produces the required values which are the input for the circuit of figure 4. When one of these values is available, the control unit enables accordingly the ld signals. Suppose that the available value is y(n-1). Then the ld1 signal will be the only ld signal which will be set. Register 1 will load the value of y(n-1) multiplied by 2. When all registers have been loaded then the add signal will produce y(n). The whole synchronization mechanism is shown in table 1.

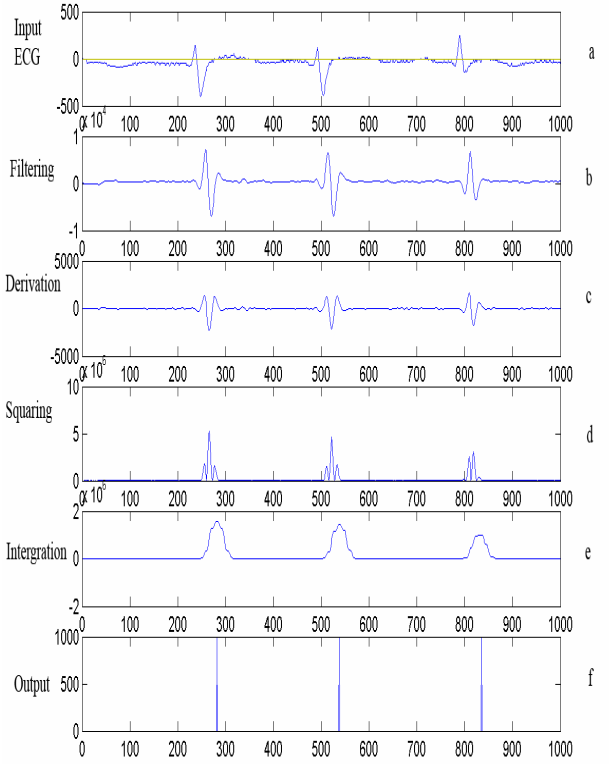

Figure 5: Output of each stage. (a) Original signal, (b) Output of bandpass (highpass and lowpass) filter, (c) Output of derivation, (d) Output of squaring, (e) Output of integration, (f) Output pulse.

ISSN: 1727-1983

After the execution of the first five stages the stage of decision takes place. During this step, peaks are detected in the data generated by the previous stages. In this phase the computed values of the thresholds are taken into account in order to identify the QRS peak in each RR interval. The thresholds and the duration of the expected RR interval are dynamically adjusting with the shape of the signal. Once a peak is detected the system generates a pulse.

The results of the implemented hardware system are presented in figure 5 where the output of each stage is shown for 1000 samples of the European ST-T database.

System Evaluation

The proposed hardware implementation achieves a speed up of 250% compared to a software implementation. The software implementation uses a conventional Risc microprocessor. Provided that the technology used for the hardware implementation is the same with the one used for the microprocessor we can safely claim that the clock frequency for both implementations may be the same. Consequently the performance in all implementations is measured in clock cycles. Measurements have been taken for various numbers of samples. In figure 6 the performance evaluation is shown for 1000, 5000, 10000, 20000 and 30000 samples.

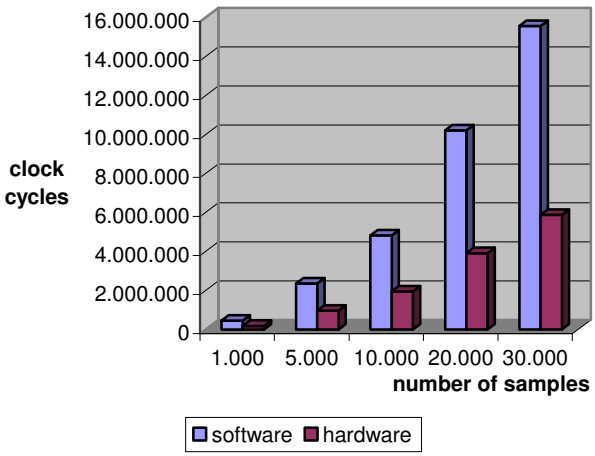

Figure 6: Performance evaluation of the implemented system for various numbers of samples.

In figure 6, is clearly shown that the hardware implementation is faster. The speed-up factor is close to 2.5 and gradually increases as the number of samples increases. This augmentation of the speed-up factor is shown in figure 7.

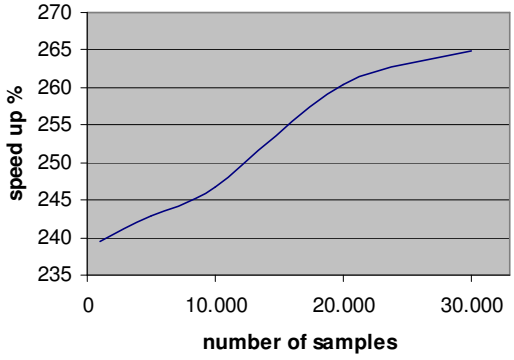

Figure 7: The fluctuation of the speed-up factor according to the number of samples

Conclusions and future work

In this paper we propose a hardware implementation of the Pan and Tompkins QRS detection algorithm based on a Xilinx FPGA board, accelerating the performance by a factor of approximately 2.5 compared to the software approach.

Our current research is focused on enhancing the performance of the presented architecture, by using pipelining techniques. Pipelining [21] is an implementation technique in which multiple tasks are overlapped in execution. The execution of each stage may not occur sequentially one after other but there may be an overlap in execution. In our current implementation every stage processes the entire sample and then can the next stage begin. However, the samples that have already been processed by a stage may be processed by the next without waiting all the samples to be processed by the first.

References

[1] PAN, J., and TOMPKINS, W. J. (1985): 'A Real-Time QRS Detection Algorithm", IEEE Trans. Biomed. Eng., 32, pp. 230-236

[2] PALNITKAR, S. : 'Verilog HDL, A guide to digital design and synthesis', PRENTICE HALL, Second Edition

[3] XILINX, Internet site address: http://www.xilinx.com

[4] MAHOUDEAUX, P. M. et al. (1981): 'Simple Microprocessor-based System for On-line ECG Analysis', Med. Biolog. Eng. Comput., 19, pp. 497-500

[5] FRADEN, J., and NEUMAN, M., R., (1980): 'QRS Wave Detection', Med. Biolog. Eng. Comput., 18, pp. 125-132

[6] GUSTAFSON, D. et al. (1977): 'Automated VCG Interpretation Studies Using Signal Analysis Techniques', R-104 Charles Stark Draper Lab., Cambridge, MA

ISSN: 1727-1983

[7] MENRAD, A. et al. (1981): 'Dual Microprocessor System for Cardiovascular Data Acquisition, Processing and Recording', Proc. of 1981 Int. Conf. Industrial Elect. Contr. Instrument., pp. 64-69

[8] HOLSINGER, W. P. et al. (1971): 'A QRS Pre-processor Based on Digital Differentiation', IEEE Trans. Biomed. Eng., 18, pp. 212-217

[9] BALDA, R., A., et al. (1977): 'The HP ECG analysis program' in VANBEMNEL J. H. and WILLEMS J. L. (Ed): 'Trends in Computer-Process Electro-cardiograms' , (North Holland), pp. 197-205

[10] AHLSTROM, M., L., and TOMPKINS W. J. (1983): 'Automated High Speed Analysis of Holter Tapes with Microcomputers', IEEE Trans. Biomed. Eng., 30, pp. 651-657

[11] ENGELSE, W., A., and ZEELENBERG C. (1979): 'A Single Scan Algorithm for QRS Detection and Feature Extraction', Proc. of IEEE Comput. Card., Long Beach, pp. 37-42

[12] OKADA, M., (1979): 'A Digital Filter for the QRS detection complex', IEEE Trans. Biomed. Eng., 26, pp. 700-703

[13] FRIESEN, G., M., JANNETT T. C., JADALLAH M. A., YATES S. L., QUINT S. R. and NAGLE H. T. (1990): 'A Comparison of the Noise Sensitivity of Nine QRS Detection Algorithms', IEEE Trans. Biomed. Eng., 37, pp. 85-98

[14] PAPAKONSTANTINOU G., SKORDALAKIS E. and GRITZALI, F. (1986) 'An attribute grammar for QRS detection', Pattern Recognition, 19 n.4, p.297-303,

[15] NYGARDS, M., and SORNMO, L. (1981): 'A QRS Delineation Algorithm with Low Sensitivity to Noise and Morphology Changes', Proc of Comput. Cardiol., pp. 346-350

[16] MEAD, C., N., et al. (1981): 'A Frequency Domain-Based QRS Classification Algorithm', Proc of Comput. Cardiol., pp. 351-354

[17] BREKELMANS, F., M. and DE VAAL C. D. R. (1981): 'A QRS Detection Scheme for Multichannel ECG Devices, Proc of Comput. Cardiol., pp. 437-440

[18] BORJESSON, P., O. et al. (1982): 'Adaptive QRS Detection Based on Maximum a Posteriori Estimation', IEEE Trans. Biomed. Eng., 29, pp. 341-351

[19] THAKOR, N., V., WEBSTER J. G. and TOMPKINS W. J. (1983): 'Optimal QRS Detection', Med. Biol. Eng. Comput., 21, pp. 343-350

[20] SORNMO, L., PAHLM, O. and NYGARDS M. (1985): 'Adaptive QRS Detection: A Study of Performance', IEEE Trans. Biomed. Eng., 32, pp. 392-401

[21] PATTERSON, D., HENESSEY J. (1998): 'Computer Organization and Design: The Hardware/Software Interface', MK PUBLISHERS

[22] RANGAYYAN, R.: 'Biomedical Signal Analysis, A Case Study approach', IEEE Press Series in Biomedical Engineering

[23] KOHLER, B., HENNING, C. and ORGLMEISTER R.,(2002):'The principles of software QRS detection', IEEE Engineering in Medicine and Biology, pp.42-57

[24] GRITZALI, F. 'Towards a generalized scheme for QRS detection in ECG waveforms', Signal Processing, 15,pp. 183-192, 1988

[25] GRITZALI, F., FRAGAKIS G. and PAPAKONSTANTINOU G., 'A comparison of the length and energy transformations for the QRS detection', Proc. 9th Annual conf. IEEE Engineering in Med. And Biology Society. Boston, 1987

[26] PAPAKONSTANTINOU, G. and GRITZALI F.(1981) :'Syntactic filtering of ECG waveforms', Compt. Biomed. Res., 14, pp.158-167

[27] PAPAKONSTANTINOU, G, SKOLDALAKIS E. and GRITZALI F.(1986):'An attribute grammar for QRS detection' , Pattern Recognit., 19, pp. 297-303

ISSN: 1727-1983