# FPGA-Based derivative module for bioimpedance signal

**Sofienne Mansouri1† , Halima Mahjoubi2† and Ridha Ben Salah3†,**

Biophysics Research unit (Faculty of Medicine, Sousse, TUNISIA), Tunisia

**Summary**
Input here the part of summary. The use of Field Programmable Gate Arrays (FPGA) instead of Digital Signal Processors (DSP) or microcontrollers increases because of its flexibility. In this paper, a novel approach is proposed to implement derivative unit of plethysmographic signal using FPGA components. The module design optimization is realized by following rigorously an efficient methodology. The validity of the proposed method is shown through simulation results.

***Key words:***
*Bioimpedance, Derivative module, FPGA, Matlab.*

## 1. Introduction

Field Programmable Gate Array (FPGA) presents these last years a lot of interest. It is a reconfigurable hardware platform useful for the implementation of high digital functions variety. Using fixed point [1]., parallel computational structures, FPGA provides computational speeds as much as 100 times greater than those possible with DSP [2]. Besides its programmability, it permits the use of different fixed point format for each module in the same system.

The designers' tendency is to expand reusable modules in order to help users to develop their systems quickly and efficiently. However, designing modules requires an efficient methodology ensuring the reduction of both execution time and hardware resources consummation.

In deed FPGA can also be considered as an appropriate solution in order to boost performances and consequently to reduce the gap between the analog and digital world. When associated to fast ADC, the extremely fast computation capability of FPGAs allows a few microseconds real-time computation of algorithms in spite of their complexities. Furthermore, as DSPs, FPGAs are very low cost components. Even recently, a company has also introduced an FPGA family that includes several analog functions, such as an ADC.

The aim of this paper is to present a realization of derivative unit of plethysmographic signal (Fig1) indispensible to determine cardiovascular parameters [3].The development of the derivative unit will be firstly exposed detailing the different methodology steps. Simulation results done by ModelSim software will be then presented to validate the well working of the module.

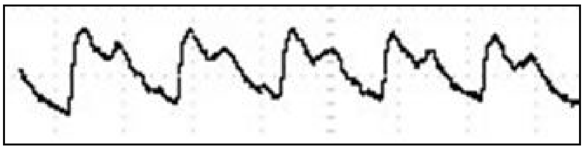

Fig . 1. Bioimpedance signal

## 2. Methodology

Field Programmable Gate Array (FPGA) presents In order to implement the algorithm on an FPGA device, design is developed following rigorously, from the initial specifications to the final hardware implementation, an appropriate design methodology [4] that offers considerable hardware design advantages [5], like reusability and optimization of FPGA-consumed resources. Coding of the optimized developed architecture is made with the VHDL Hardware Description Language (Very high speed integrated circuit Hardware Description Language) for the hardware implementation. The implemented architecture is then tested for the validation of the design. The different stages of this methodology are resumed in the diagram illustrated in (Fig.2).
Based on a modular architecture principle, the developed architecture is divided into two main parts, namely sequencer and data-path. The data-path is defined structurally as a network of registers, operators, multiplexers and buses. The sequencer is designed to control the communication inside the data-path. Its main task is to preserve the synchronization of data when crossing.

In the following, an application of each stages mentioned before is done for the derivative module.

Manuscript received December 5, 2009
Manuscript revised December 20, 2009

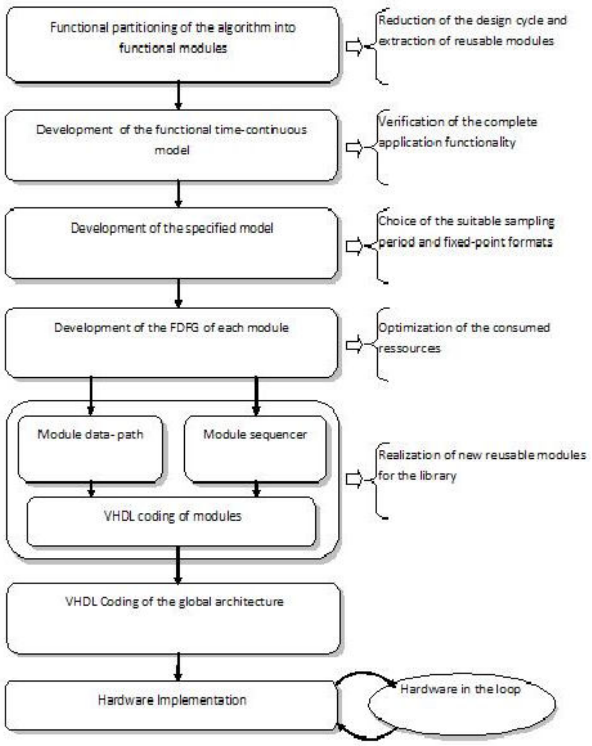
Functional partitioning of the algorithm into functional modules
Development of the functional time-continuous model
Development of the specified model
Development of the FDFG of each module
Module data-path
Module sequencer
VHDL coding of modules
VHDL Coding of the global architecture
Hardware Implementation
Hardware in the loop
Reduction of the design cycle and extraction of reusable modules
Verification of the complete application functionality
Choice of the suitable sampling period and fixed-point formats
Optimization of the consumed ressources
Realization of new reusable modules for the library

Fig .2. Development Methodology

## 2.1 Equations development:

The equation (see Eq. 1) presents the derivative (LAGRANGE):

$$y(x) = f'(x0) = \frac{1}{12*Te} * (f[x-2] - 8*f[x-1] + 8*f[x+1] - f[x+2]) \quad (1)$$

$$y(x) = \frac{1}{12*Te} * (f[x-4] - 8*f[x-3] + 8*f[x-1] - f[x]) \quad (2)$$

The discreet-time model of these equations is given below:

$$Y[k] = \frac{1}{12*Te} * (f[k-4] - 8*f[k-3] + 8*f[k-1] - f[k]) \quad (3)$$

Te: is the simple period chosen.

Coding the module inputs/outputs and constants must be in per unit “pu”; so, bases values must be fixed. The obtained equations are expressed in (4):

$$Yu[k] = Cu * (fu[k-4] - 8*fu[k-3] + 8*fu[k-1] - fu[k]) \quad (4)$$

$$Cu = \frac{Fb}{12 * Te * yb}$$

The indices “b” indicates bases values.

## 2.2 Simulation and Algorithm Refinement Procedures:

The simulation procedure is performed under Matlab-Simulink software environment. It is aimed to:

- Verify the functionality of the complete application.
- Find the suitable sampling period and fixed-point format refinement for each variables according to the needed performance constraints.

The functionality verification can be achieved by the development of a functional model using Simulink time-continuous blocks (Fig.3). Then, the parametrization of the per unit digital algorithm is performed by studying the influence of the sampling period and the effects of the chosen fixed-point format. Note that, the choice of the fixed-point format can be derived from adapted methods [6],[7]. At this level, the simulation is realized by the development of a digital fixed-point specification model using Matlab toolbox fixed point.

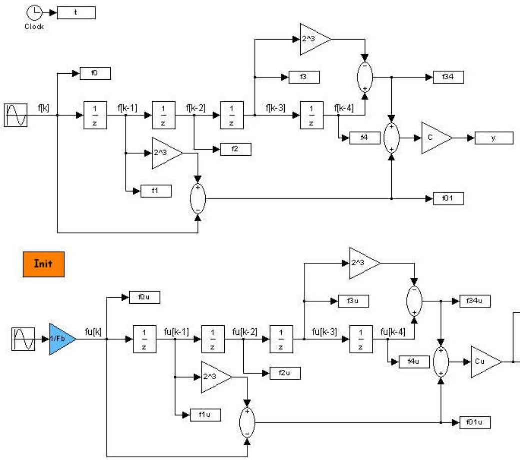

Fig.3. Functional model (in discreet-time & in per unit)

After a simulation in a continuous state, the model is designed and tested in MATLAB with the toolbox MATLAB. This toolbox provides the user with a library of blocks representing functions implantable in FPGA target as shown in (Fig.4).

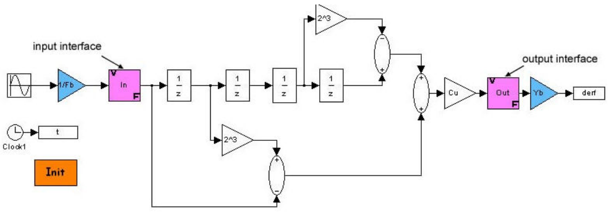

Fig.4. Specification model

Simulating using Toolbox fixed point permits to fix the bit number and the fixed point format required to code the derivate module inputs/outputs and constants.
The bit number permitting to have continuous state results and simulation results is '19' for inputs and outputs.

The results are overviewed in (Fig.5).

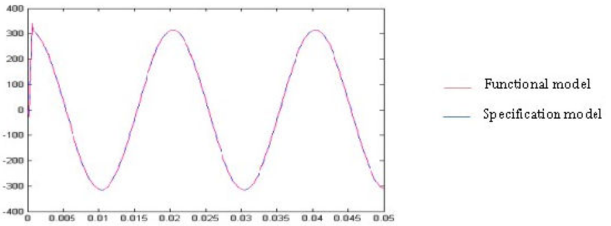

Fig.5. Simulation results (using 'Sinus' as input)

2.3 *Consumed resources optimization:*

In this stage and according to the expression (4), a Data Flow Graph (DFG) is determined. DFG is a graphical representation of the algorithm, which is aimed to bridge the gap between a high level algorithm representation and its hardware implementation [8].It includes no timing specifications regarding its expected implementation. It is composed of nodes and edges. Each node represents a simple arithmetic operation or a simple mathematical or logical function and each edge corresponds to a data transfer. After development of DFG, an Algorithm Adequation Architecture (A3) methodology [9] must be applied to find out an optimized hardware architecture for a given application algorithm, while satisfying size and timing constraints. In DFG, some operations are used several times.

If an operator is repeated n times, the A3 factorization process applied to this operator consists in keeping only m realizations of this operator with m < n.

Most of the time m is equal to one. Operator has to be understood here as the hardware support of a given operation. The A3 methodology is generally applied to the greediest operators in terms of hardware consumed resources like multipliers. The result of the DFG factorization is the *Factorized Data Flow Graph* (FDFG).

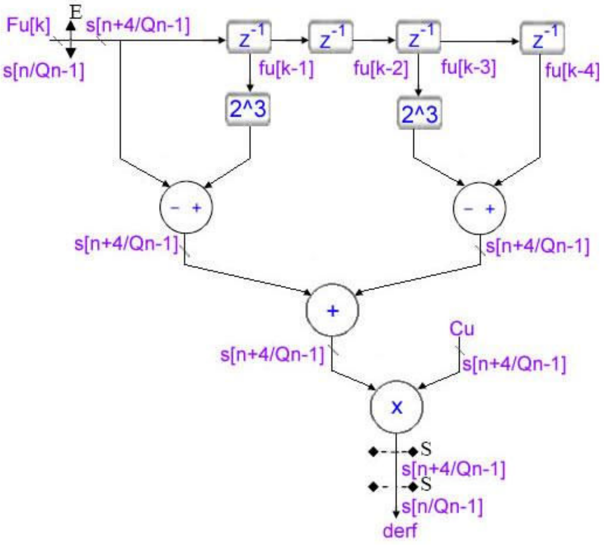

Fig.6. Data Flow Graph (DFG)

Doesn't incorporate repetitive operations that occupy a vast space that can be factorized, the data path is directly drawn from the DFG without going through the factorization phase.

2.4 *VHDL coding:*

The use of registers in relevant in the module developed in VHDL. A modular principle is followed to split the architecture into two main parts: a data-path and a control unit. The data-path of the architecture is always obtained by replacing each node of the final graph with its corresponding operator and every edge with a data bus between operators. Data bus transfers are managed by a control unit, which is a simple Finite State Machine (FSM), which is synchronized with the clock signal (Clk). The control unit (FSM) of a module is always activated via a Start pulse signal.

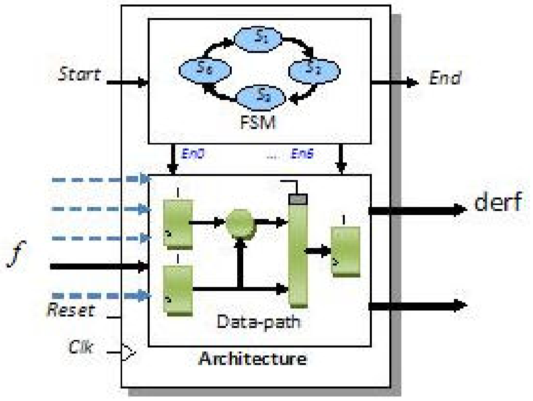

Fig .7. Architecture of the algorithm

$f$ : input (plethysmographic signal), derf: output.

In the following each of data-path and controller of the derivate unit are detailed on (fig.8). Both of the two will be gathered in a same module.

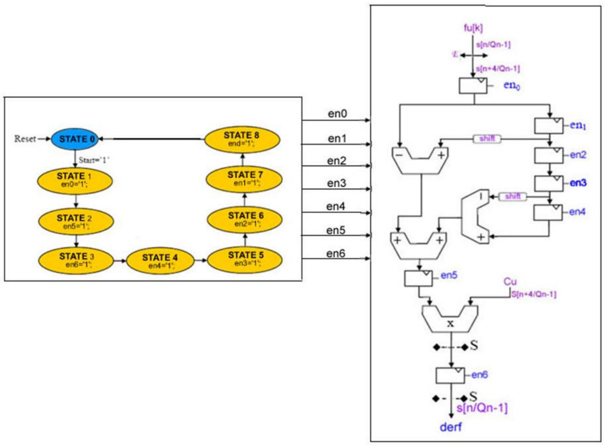

Fig.8. (a): Data-path and (b) : controller of the derivate algorithm

Since the constant "Cu" fixed point format must be n+4/Q(n-1), an extension of the input must be done (n+1/Qn-1)→n+4/Q(n-1). The multiplication output bit number is 2n .A slice operation is, consequently, applied to this output to get it on n bits. When the computation time process is over, an End pulse element signal indicates to the global unit that the data outputs of the module are ready to be used.

The developed module can be realized using only a single FPGA, for example SPARTAN III XC3s400 chip from Xilinx.

The consumed resources ratio is equal to 9% (352 logic ports). This consists one of the many advantages of the FPGA target use since more than one module can be programmed in a same chip.

## 3. Results

The aim here is to check the well working of the derivate module coded in VHDL by carrying on a simulation with software named ModelSim. So, the input signals of the module are programmed in VHDL and the ModelSim software generates the output (Fig.9). Comparison between this output and the one obtained by simulation on MATLAB-Simulink environment for the same inputs is done in order to validate the module.

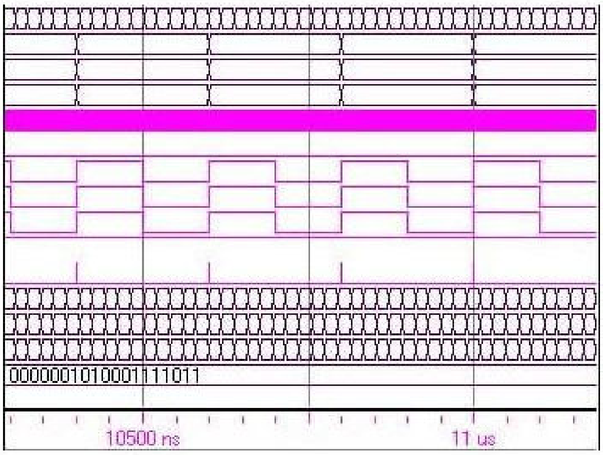

Fig.9. Simulation results obtained with Modelsim

The superposition of the outputs is shown in (Fig.10), validates the well working of the derivate module carried in VHDL.

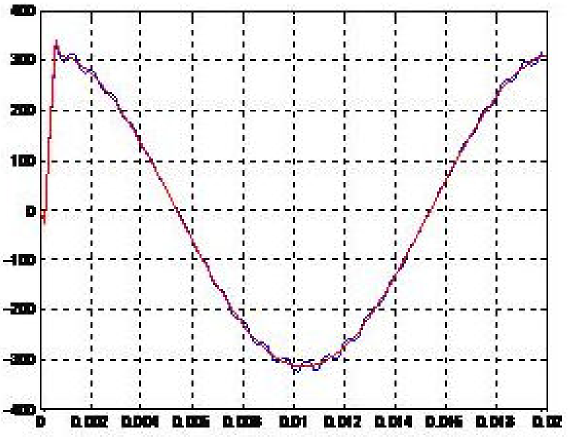

Fig.10. Simulation results by MATLAB-Simulink (Red) and by ModelSim (Blue)

## 4. Conclusion

In this paper, a pertinent use of the FPGA is presented for medical devices working in real time. A derivate module is implemented respecting an efficient methodology. The first step consists on the development of the relative equation expressing the output in continuous state, discreet time and in per unit. Second, the hardware resources used are optimized by applying A3 methodology to DFG to obtain FDFG graph. After, a simulation of the module in continuous state under MATLAB-Simulink environment is done followed by another simulation using the toolbox fixed point of MATLAB to fix the adequate fixed point format and the bit number. Then, the architecture (data-path and controller) is synthesized in a structural VHDL code. Finally, the developed architecture is tested with a simulation under ModelSim software to validate the well working of the derivate module.

## References

[1] Hyeokho Choi – "Fixed Point Arithmetic" –Version 2.1: 2003/02/24 13:29:34.8572 US/Central.

[2] Zhengwei Fang, Joan E. Carletta, Robert J. Veillette –"A Methodology for FPGA-Based Control Implementation" – IEEE transactions on control systems technology, vol. 13, no. 6, November 2005.

[3] Patterson RP. 1989. "Fundamentals of Impedance Cardiography", IEEE Engineering in Medicine and Biology Magazine 8:35–38.

[4] M-W.Naouar, L.Charaabi, E.Monmasson, I.Slama-Belkhodja –"Realization of a Library of FPGA ReconfigurableIP-Core Functions for the Control of Electrical Systems"…..

[5] L. Charaabi, E. Monmasson and I. Slama-Belkhodja "Presentation of an efficient design methodology for FPGA Implementation of Control Systems. Application to the Design of an antiwindup" PI controller IECON 2002_The 28th Annual Conference of the IEEE Industrial Electronics Society, 5-8 November 5-8, 2002; Sevilla – Spain.

[6] D. Menard and O. Sentieys. "Automatic Evaluation of the Accuracy of Fixed-point Algorithms," in Proc. IEEE/ACM Conf. Proc. on Design, Automation and Test in Europe, 2002, CD-ROM.

[7] F. Zhengwei, J. E. Carletta, and R. J. Veillette, "A methodology for FPGA-based control implementation," IEEE Trans. Control Systems Technology, vol. 13, no. 6, pp. 977-987, Nov. 2005.

[8] M-W.Naouar, L.Charaabi, E.Monmasson, I.Slama-Belkhodja – "Realization of a Library of FPGA Reconfigurable IP-Core Functions for the Control of Electrical Systems" – EPE 2004.

[9] T. Grandpierre, C. Lavarenne and Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessor," in Proc. CODES'99, Rome, Italy, May 1999, CD-ROM.

**Mansouri sofienne** is an electronic engineer, he received his master in biomedical engineering from the university of technologies of compiègne (UTC-France). He is now preparing his doctoral thesis at the Faculty of sciences of Tunis and teaching at Higher Institute of Medical Technology of Tunis.