

UNIVERSIDADE ESTADUAL DE CAMPINAS
SISTEMA DE BIBLIOTECAS DA UNICAMP
REPOSITÓRIO DA PRODUÇÃO CIENTÍFICA E INTELECTUAL DA UNICAMP

Versão do arquivo anexado / Version of attached file:

Versão do Editor / Published Version

Mais informações no site da editora / Further information on publisher's website:

<https://www.scielo.br/j/rbef/a/74bjqGXFmZJm7pkR5rL3Mgt/>

DOI: 10.1590/1806-9126-RBEF-2019-0208

Direitos autorais / Publisher's copyright statement:

©2020 by Sociedade Brasileira de Física. All rights reserved.

DIRETORIA DE TRATAMENTO DA INFORMAÇÃO

Cidade Universitária Zeferino Vaz Barão Geraldo

CEP 13083-970 – Campinas SP

Fone: (19) 3521-6493

<http://www.repositorio.unicamp.br>

Introducing programmable logic devices in physics laboratories: a practical guide for the implementation of experiments

Cristhof Roosen Runge¹, Matheus Cerqueira¹, Francisco José Arnold^{*1} 

¹Universidade Estadual de Campinas, Faculdade de Tecnologia, Limeira, SP, Brasil

Received on August 15, 2019. Revised on October 29, 2019. Accepted on November 11, 2019.

Modern electronic instrumentation techniques are being introduced in the programs of the Physics and Physical Engineering courses and, in this context, the programmable logic devices play an important role. We present the fundamental steps for the design, development, and validation of systems based on programmable logic devices. Two digital filters have been implemented for illustrating the guide procedures. The results show that the function–response complies with the common requirements of generic filter performance. This work is targeted at students and lectures and aims stimulate and challenge them to develop electronic instrumentation using programmable logic devices in teaching activities.

Keywords: Electronic, Instrumentation, Filter.

1. Introduction

The electronic instrumentation resources are increasingly present in the laboratories of Physics and Physical Engineering, so much is that contents of this subject are immersed in the curricula of these courses. Because of the increasing technology advancements, upgrades in knowledge are becoming more and more pressing so that the training of students in these areas are maintained at high-level. Therefore, the contents to be taught in Physics and related courses may be updated. Often, these updates transcend consolidated foundations for many years in the academic world. Particularly, the technological evolution in the electronics has made great progress with the introduction of programmable logic devices in the late twentieth century. This work aims to present a working tool that, among many other possibilities, allows the design of digital filters that are widely used in current technological experiments and developments.

In the last decades, the dissemination of the technology has allowed the introduction of new low-cost resources in teaching laboratories [1]. The immersion of the technology in the teaching field is well-timed and very interesting under the points of view of the dynamism in the development of laboratorial practices, and updating and renewal of contents. Further, these new technologies may be used as alternatives for the replacement of old procedures.

The analog electronic instrumentation techniques are being transformed by a new paradigm based on digital techniques, mainly due to the evolution of the programmable devices. The integration of hardware and

software allows the development of virtual and real experiments and its insertion upon the student training. Further, the design techniques may be easily learned by users that have some practice with electronics.

Experiments with signals filtering are very common in the daily routine of the Physics and Physics Engineering teaching and researching laboratories. Analog filters are very common in physics laboratories because they can be used in many experiments, therefore they are a recurrent issue presented in the physics books [2-4]. Filters are circuits used to attenuate frequency components of an electric signal [5]. For instance, a low-pass filter can be used to attenuate high frequencies noise from the signals provided by a sensor. The filters are classified as: a) the response-function, the function related to the cutoff frequencies (low-pass, high-pass, band-pass, band-stop); b) the fundamental technology (passive or active, and analog or digital). The analog filters are mounted on a network consisting of resistive and reactive components on passive configuration. On the other hand, an active filter is implemented with passive components and an electronic amplifier device.

1.1. Related work

Several papers have been published in the journals specialized in education including electronic resources and computational simulation. Wireless Sensors Network is an interesting possibility for the implementation in electronic monitoring of sensors [6]. Microcontrolled systems for the control of experiments have been frequently reported in the literature [7,8]. Nowadays, the Arduino is a platform much used due to the easy of the programming

*Correspondence email address: arnold@ft.unicamp.br.

and low-cost [9-13]. LabView (National Instruments) has friendly interface and excellent resources for the emulation and automation of systems [14], whereas it is more expensive.

Arduino [15] is a device very popular because the low-cost, ease of programming and functionality. The basis of Arduino is an Atmel microcontroller with some more hardware including inputs/outputs for the connection with external circuits, such as sensors and actuators. Further, Arduino can be connected to a computer by a USB cable and programmed in C/C++ language. These characteristics make it very attractive for the people with little knowledge in hardware and/or software and allows to the users build simple settings with several goals, among them experiments for the physics laboratories.

On the other hand, the Field-Programmable Gated Arrays (FPGAs) [16] are not common in the practical activities of the laboratories for physics teaching and a few papers were found in our search [17,18]. The main reason may be the challenge of introducing a new component in the experimental activities of these laboratories. However, these components must be considered due to the large possibilities to implement new solutions for the experiments. The reader not familiar with the use of FPGAs may find fundamentals for VHDL and FPGA programming in [19,20].

This paper is intended to provide the fundamental steps for the electronic instrumentation developers, on graduate level, aiming to stimulate the creation of new and efficient solutions for the experimental practice both for the research and teaching.

1.2. Goals of the work

Aiming to expand the possibilities of the developers of electronic instrumentation in physics laboratories, we have focused on digital system design using a FPGA to build a Finite-Impulse-Response (FIR) filter. We have used a filter design which could be employed in several experiments in physics teaching laboratories to illustrate the technical implementation. The step-by-step demonstration may stimulate the development of other prototypes to be used in specific physics experiments.

This paper shows how to design, develop, and validate a generic FIR filter based on a programmable device. This filter may be used in experiments on demand for teaching in physics laboratories, aiming stimulate students and instructors to incorporate concepts and develop skills in digital electronic.

1.3. Organization of the paper

This paper is organized in five sections. In Section II, the fundamental concepts of FPGA and digital filters are presented. In this section, the reader comes into contact with some concepts not always presented in the physics courses. In Section III, the procedures used in this work are shown, including the emulation, the FPGA's

programming and the circuit design. The results and discussions from the implementation of two filters FIR are in Section IV. Finally, in Section V, the conclusion is presented.

2. Digital system design using FPGA and digital filters

In order to design a digital system to perform digital signal processing, different approaches and technologies can be used. Perhaps the three main approaches when dealing with digital circuits are: microprocessor-based design (e.g. Arduino, Raspberry), the Digital Signal Processor (DSP) based design (e.g. Texas TMS320 DSPs, Analog Devices Blackfin, Shark DSPs) and FPGA-based design (e.g. Intel, Xilinx). The choice of one of these approaches may be dependent on many issues.

Briefly speaking, microprocessor based design has the advantage of usually being performed in high-level programming languages. Nevertheless, the design is restricted to the microprocessor architecture and, as microprocessor code execution is serial, may limit the speed as well as the execution of parallel tasks.

DSP based design has the advantage of having a dedicated hardware architecture to digital signal processing, although the designs are strongly dependent on architecture.

FPGA based system has the advantage of giving more freedom to design that is not possible in other approaches. For example, it is possible to design either a single logic gate such as a NAND, or implement an entire processor or DSP or both in a single piece of silicon. The FPGAs are commonly used to validate circuits such as microprocessors in prototyping stage before the design of the final definitive silicon mask for manufacture.

A FPGA is a silicon device consisting in an array of logical gates that can be dynamically connected via programming. This fact allows the FPGA to be reconfigurable devices very convenient for prototyping digital circuits. Another advantage of using FPGA is writing codes in VHDL (Very High Speed Integrated Hardware Description Language), or Verilog language for circuit design [21,22]. This fact guarantees portability of the design for the different FPGA vendors.

Finally, in terms of educational aspects, a digital circuit developed in VHDL is a straightforward solution, that is, as it is not vendor dependent, such as microprocessors or DSPs, the students can identify the real circuit design based in pure logical gates. For example, for a filter FIR (finite impulse response) design, one can easily identify the same delay tap structure, multipliers and summations found in the classical theory textbooks [23,24].

The full path for the signal processing in a digital filter system starts with an analog-digital converter (ADC) that produces discrete-time samples from an analog input signal. A digital circuit (FPGA, in this case) changes these samples by performing mathematical operations.

Then, the transformed samples are converted from the digital to analog signal in a digital-analog converter (DAC). Thus, the resulting signal is restored with changes in amplitude and phase compared to the original input signal. The digital filters are separated in two categories [25]: the infinite impulse response (IIR) and the finite impulse response (FIR) filters. In this work, we only addressed FIR filters.

2.1. The FIR filter

Mathematically, a FIR filter performs the discrete convolution of two sequences. One of them generally corresponds to the sampled signal input $x[n]$, provided by the analog to digital converter (ADC). The second corresponds to the finite impulse response of the filter $h[n]$, represented by its coefficients. The output of the filter is simply $y[n] = x[n] * h[n]$, that can be performed with the convolution sum as described in Equation 1

$$y[n] = \sum_{-\infty}^{\infty} h[k] x[n-k] \quad (1)$$

where $h[k] = b_k$ for $0 \leq k \leq M$ and 0 otherwise, and b_k is the k^{th} coefficient of the FIR filter. Therefore, Equation 1 can be rewritten as

$$y[n] = \sum_0^M h[k] x[n-k] \quad (2)$$

Expanding Equation 2

$$y[n] = b_0 x[n] + b_1 x[n-1] + \dots + b_M x[n-M] \quad (3)$$

We can easily see that the FIR corresponds to the summation of the delayed versions of the input signal multiplied by the FIR coefficients, and this is the beauty of the FPGA VHDL implementation. The structure of the filter will reflect exactly this operation as can be seen in Figure 1.

In Figure 1, $x[n]$ denotes the input signal. The time-delayed versions of the input signal ($x[n-1]$, $x[n-2]$, ... $x[n-M]$) are obtained from the tapped delay line. Each time-delayed version is pre-multiplied by the corresponding FIR coefficient b_i . Finally, all products ($b_i * x[n-i]$) are summed yielding the output signal $y[n]$. This is a classic structure of FIR filter found in the technical literature [23-25].

2.2. The windowing technique

In order to design a FIR filter, many techniques can be employed, such as: frequency sampling, optimum equiripple, windowing, to name some. In this work, we chose the windowing technique because it is one of the simplest and easiest to understand. The idea is to approximate an ideal filter, truncating its infinite response using a finite window response. Basically this operation is performed mathematically by multiplying the infinity impulse response of the ideal filter by a finite windowing function in the sequence domain, or conversely, convolving the responses in frequency, as shown in the Figure 2.

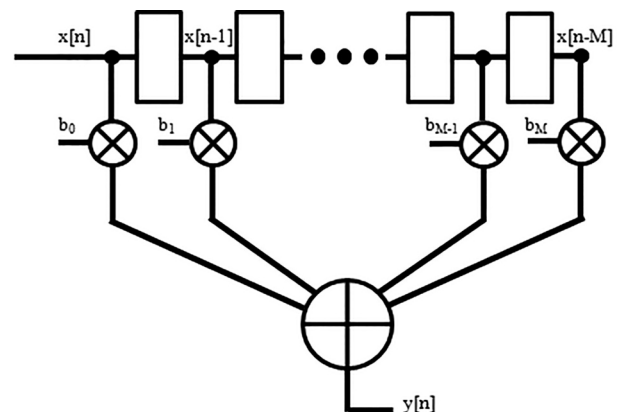


Figure 1: Structure of a FIR filter.

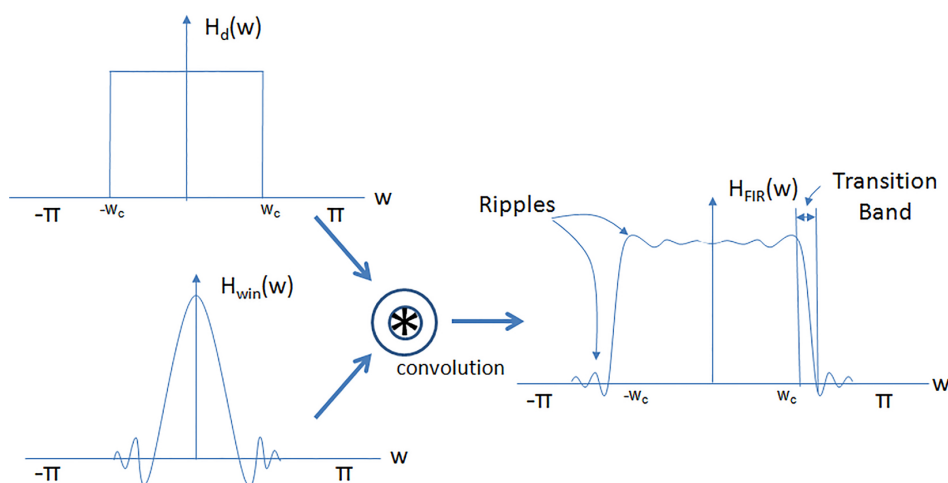


Figure 2: Finite windowing function convolution.

In Figure 2, $H_d(w)$ is the Fourier transform of the ideal filter impulse response, w is the frequency in radian and w_c is the cutoff frequency of the ideal filter, $H_{win}(w)$ is the Fourier transform of the windowing function, and $H_{FIR}(w)$ is the resulting Fourier transform of the FIR filter impulse response, that represents its frequency response. Basically, we may identify three regions of interest when projecting a filter: pass band, transition band and stop band.

The pass band region corresponds to the frequency interval where the filter will allow the incoming signal to pass to the output terminals. Sometimes, in this region the filter may introduce a gain of interest.

The transition band region corresponds to the frequency interval where the filter begins to attenuate the incoming signal, up to the frequency that reaches the maximum attenuation of the filter. Commonly, the transition region is considered to begin at the cutoff frequency, corresponding to the frequency at which a 3 dB attenuation is observed compared to the gain in the pass band region.

Finally, the stop band region corresponds to the frequency interval where the filter rejects the incoming signal, attenuating it. Typical values of attenuation in this region are 30 dB to 60 dB in relation to the gain in the pass band region.

As can be seen in Figure 2, the windowing operation introduces ripples and the transition bandwidth. These features are non-idealities in the FIR response compared to the ideal filter. Different windowing functions can be used to project the FIR filter. The choice of a specific windowing function is a tradeoff between complexity (e.g. number of coefficients, complexity of the window function) and the target parameters specification, such as: maximum allowed ripple, transition bandwidth, attenuation in the stop band. Tables with the characteristic of different windowing function can be easily found in the literature [23]. The most of the windowing function can be used in a straightforward manner as predefined functions (e.g. rectangular, triangular, Hamming, Hanning, Blackman, Kaiser). As will be seen in the next sections, in this work we will write a simple code to generate the coefficients of a FIR filter using the Kaiser windowing function. Although Kaiser windowing function is based on empirical approximations it leads to the minimum number of coefficients when compared to the other functions with the same specifications.

2.3. Butterworth filter

Among the configurations of active filters, the Butterworth filter (see some schematics in [26]) plays an important role in the design of electronic systems. This filter is composed of resistive and capacitive network and electronic amplifier. The number of resistive and capacitive devices in the network is proportional to the filter order. The order defines the rate of decreasing of the

filter response along the transition band. For instance, the response of a n th order decays $20n$ dB/decade; dependent upon the order of the filter. Butterworth's filter has flat response at the pass-band.

Although, other filters (Bessel, Chebyshev, elliptic) have sharper decay rate, the Butterworth's, when used with higher order, presents similar behavior. Further, Butterworth's filter does not present ripple at pass-band. These characteristics makes it attractive for many designs and suitable for the goals of this paper.

3. Materials and Methods

3.1. The integrated development environment steps

In this section, the steps followed to create an experiment from the initial specification of the FIR parameters to the final circuit implementation are described. This section describes the step by step procedure to simulate, validate and to synthesize the VHDL code of the FIR filter circuit. Firstly, in order to prepare the environment for simulation, different files are created and generated to produce the stimulus for the simulation, capture the outputs and validate the simulation results. For the simulation and validation of the results, the Octave (GNU Octave – J. W. Eaton, free software) and the free version of ModelSim (INTEL – FPGA Edition Software) are used. All code files written for this work are found in <https://github.com/RoosenRunge/PDSKitRef>. Further, in the same link, we have added a detailed step-by-step script to orient the developers to implement the device. Figure 3 shows a block diagram with the different stages, inputs and outputs of the integrated developed environment used for the simulation. Finally, after simulation and behavioral validation, the synthesis to implement the real circuit is performed using the free version of Quartus II (INTEL). In the following, the steps for simulation and synthesis are described.

3.1.1. Generating the FIR parameters

In order to create a full integrated educational experience, the first step in the filter design starts with the fundamental specifications such as cutoff frequency, transition bandwidth, attenuation in rejection band, and sampling frequency. Using the initial specification, a simple code was written in Octave language (genCoefs.m), to produce the coefficients values to parameterize the VHDL FIR code.

The windowing technique has been used to design the FIR filter. In this technique, only the coefficient values need to be modified to produce different filters. If a sharper or a smoother response is desired, in some situations the number of coefficients also needs to be modified. For educational purpose, the Octave code written also generates the theoretical graphical visualization of the FIR response based in the given specifications.

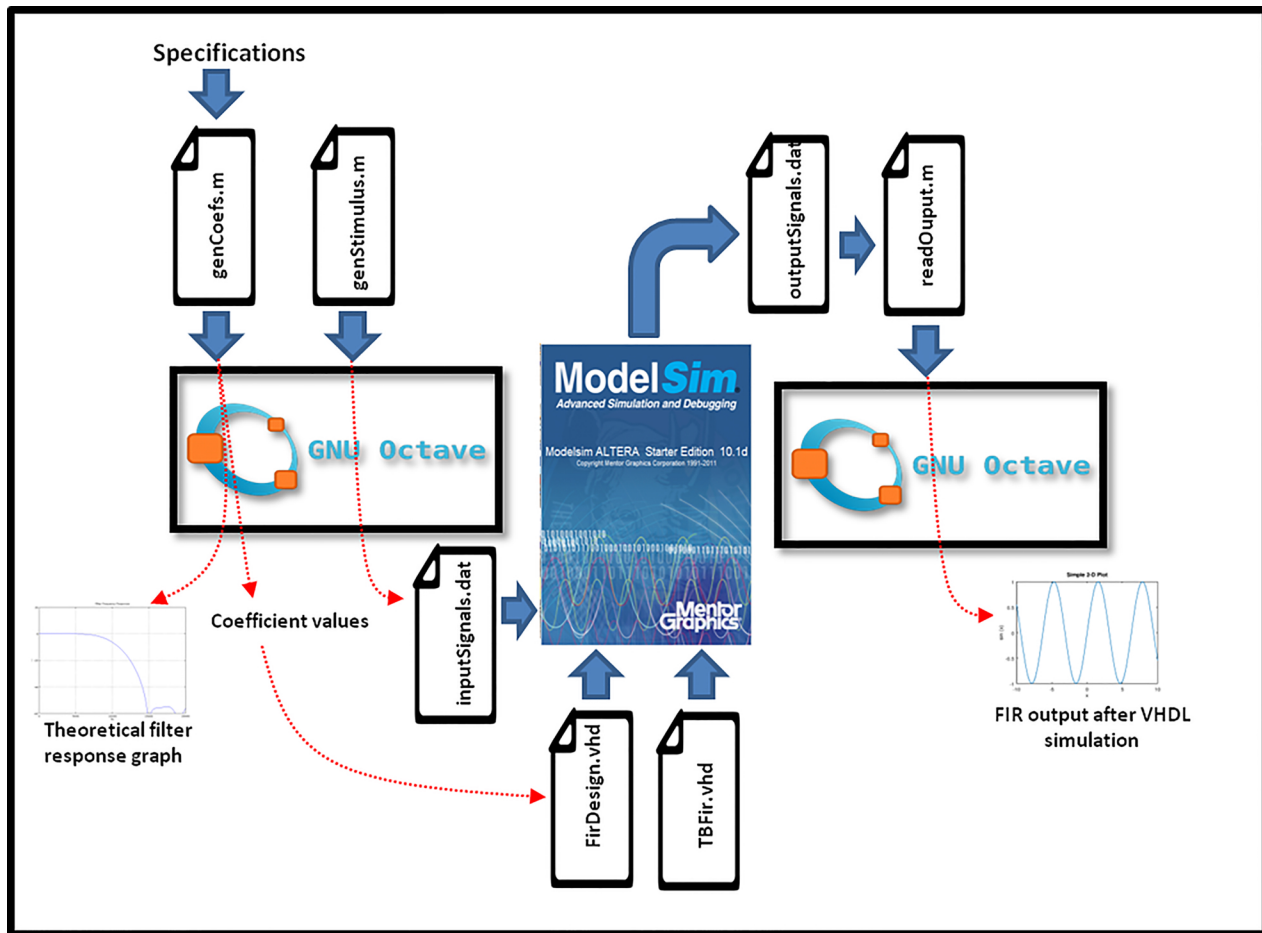


Figure 3: Block diagram for the system simulation.

3.1.2. Parameterization of the VHDL FIR code

The values of the FIR coefficients are manually modified in the FIR VHDL code (FirDesign.vhd) according to values generated by genCoefs.m in the previous stage.

3.2. Simulation and synthesis

The prediction of the system behavior was firstly performed by simulation. The whole flow has four fundamental steps: the generation of a stimulus vector file, the VHDL simulation, the reading of the results and the synthesis and programming file generation.

3.2.1. Generating the stimulus vector file

In order to validate the VHDL FIR response in a simulation environment, a second simple Octave code (genStimulus.m) is used to generate a test vector. This vector represents a sampled single tone sine wave. Each value of this vector is converted to an 8-bit word (corresponding to the signal input bus in the real FPGA device). The vector is written down in file (inputSignals.dat) that can be read by ModelSim during the simulation stage. ModelSim is a free version software provided by MentorGraphics to simulate VHDL circuits codes. By varying the sine fre-

quency in genStimulus.m file is possible to excite the FIR filter with different frequencies for simulation analysis purpose.

3.2.2. The VHDL simulation file (TestBench)

The test bench file (TBFir.vhd) is a purely behavioral VHDL code used only to test synthesizable circuits. It emulates all the stimulus supposed to be present in real FPGA device input pins, and capture the signals present in the output pins. The VHDL test bench code reads the file "inputSignals.dat" with the vector signal stimulus generated by Octave code, and write down an output file (outputSignals.dat), with the output provided by ModelSim corresponding to the signal received from the VHDL FIR (FirDesign.vhd) filter under test.

3.2.3. Reading the results graphically

In this stage a simple Octave program (readOutput.m) is used to read the output file "outputSignals.dat", and to exhibit the results graphically. It is important to note that, in spite of the fact that ModelSim provides a wave form output view, this is in digital logic level, and to

interpret the FIR behavior in terms of signal waveforms would be a difficult task.

3.2.4. Synthesis and Programing File generation

After VHDL simulation validation, the next step is to synthesize the VHDL code (FirDesign.vhd) in order to generate FPGA programming file to the evaluation board. For this purpose, we use the free version of Quartus II software. The Quartus II IDE provides many resources for complex FPGA implementation and analysis, but for educational purpose and small FPGA circuit the procedure to generate the FPGA programming file is quite simple. Using the Quartus II Wizard and the VHDL circuit file (FirDesign.vhd) validated in the simulation stage, we can create a project, specify the FPGA device, assign pins, and synthesize the circuit generating the final programming file for the board. After programming file generation, the board can be programmed and the circuit is embedded in the FPGA device.

3.3. The circuit implementation

We use a low cost (R\$180,00) Intel FPGA evaluation board for physical lab experiments. This board consists basically in a Cyclone IV FPGA (ep4ce6), a 50 MHz crystal oscillator, the logic for programming the FPGA, and four banks of IO pins. The FPGA board is shown in Figure 4.

In order to acquire the analog signal and convert to the digital input of the FPGA board and receive the FPGA digital outputs and convert back to analog signals after the digital filtering, we use an Analog-Digital Converter (ADC 0804) and a Digital-Analog Converter (DAC 0808) mounted in an experimental board using the devices datasheets basic standalone configurations. A simple second order analog Butterworth filter [26] with cutoff frequency of 20 kHz was connected to the DAC output in order to smooth the high frequencies caused by quantization process. The method for the calculation of the component values of the filter (passive network)

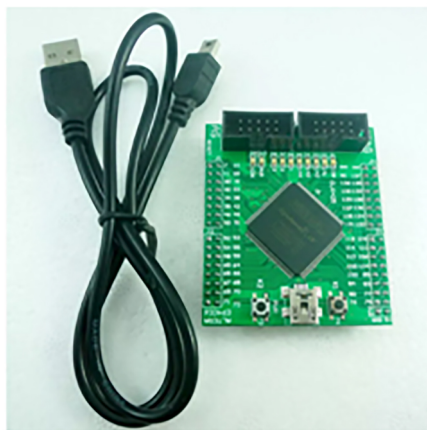


Figure 4: The FPGA board used in the project.

are also found in [26]. Figure 5 shows the experimental board.

Figure 6 shows a block diagram representing the setup of tests used to produce the experimental results. Figure 7 is the photograph of the setup.

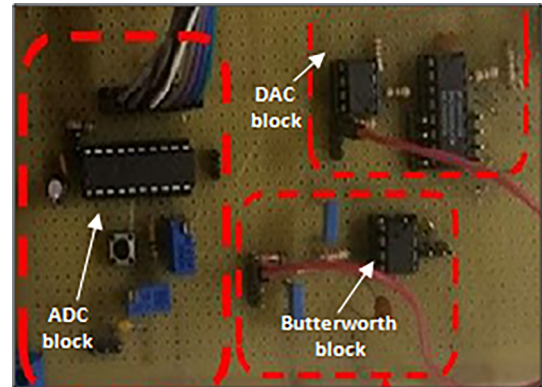


Figure 5: Experimental board employed in the tests.

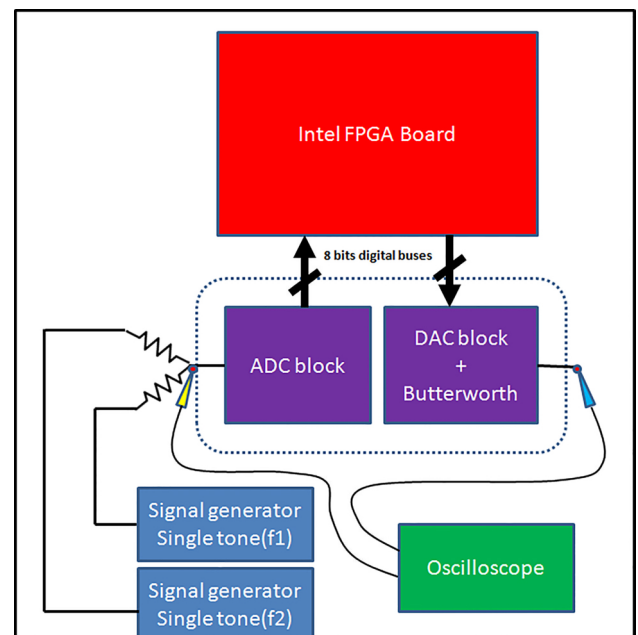


Figure 6: Setup Block Diagram of the developed system.

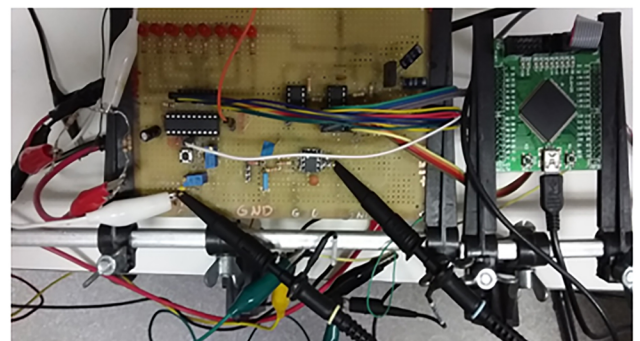


Figure 7: A photograph of the circuit under test.

3.4. Filters design

We have implemented two different FIR filters, a low-pass and a high-pass, which were used to illustrate the simulation and the implementation steps. The high-pass filter has the complementary response of the low-pass filter. The high-pass filter has simple mathematical operation. It consists of an all-pass filter minus the low-pass filter response. The low-pass filter specifications are:

Sampling Frequency (F_s): 23 kHz
 Window: Kaiser.
 pass band ($w_p=0.026\pi$) : 0 to $\cong 300$ Hz
 stop band ($w_s=0.0875\pi$) : $\cong 1000$ Hz
 transition band: $w_s - w_p$
 attenuation in stop band (linear value: $\delta = 0.015$)
 \Rightarrow (value in decibels: $-20\text{LOG}(\delta) \cong 36$ dB)

4. Results and Discussion

In this section are presented the results from the simulation and the real circuit experiment for both low-pass and high-pass filters.

4.1. Low-Pass Filter

Using the specifications above (Section III.4) and the file `genCoefs.m`, 65 coefficients FIR filter were generated. The theoretical frequency response and the temporal impulse response of the filter are shown in Figures 8a and 8b.

We use a combination of two tones with frequencies $f_1 = 100$ Hz (representing the low frequency input component), and $f_2 = 1000$ Hz (representing the high frequency input component) to test the simulation output of the low-pass filter using ModelSim. Figure 9 shows the input to ModelSim (using `genStimulus.m`) and the output provided by ModelSim (using `leSaida.m`) after the VHDL simulation. Figure 10 shows the result obtained with the real circuit after synthesizing the VHDL code.

Figure 10 shows that the goal of the filter has been achieved. The output of the low-pass filter (blue trace) is closely the low frequency component tone at 100 Hz.

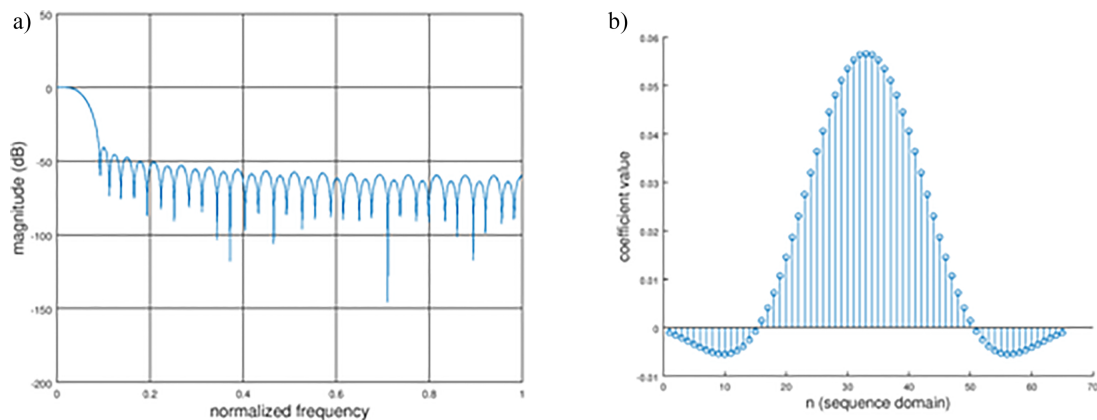


Figure 8: a) Frequency response of the low-pass filter. Magnitude (in dB) of the filter output as function of normalized frequency. b) Impulse response of the low-pass filter. Coefficient values as function of number of points.

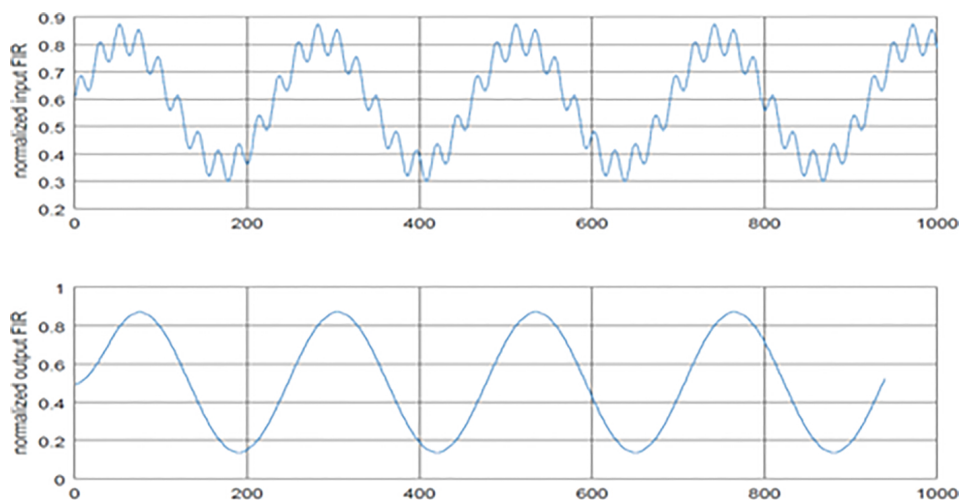


Figure 9: Input (above)/output (below) simulated wave forms of the low-pass filter.

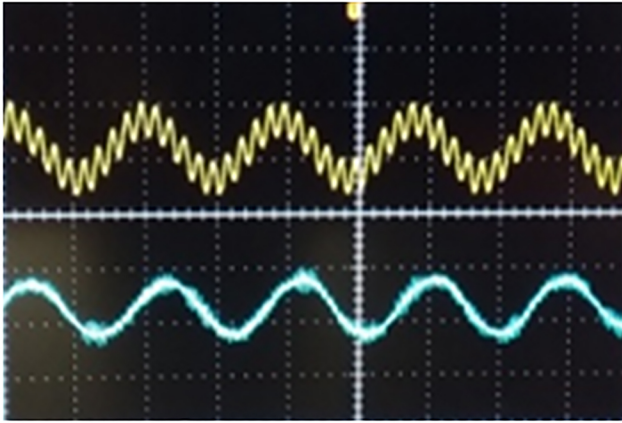


Figure 10: Input (yellow trace) /output (blue trace) obtained from low-pass filter real circuit captured from the oscilloscope screen. Each horizontal division is 5.0 ms.

4.2. High-Pass filter

Figures 11a and 11b show the result when generating the complementary high-pass filter with the all-pass minus the low-pass specification parameters.

We used the same combination of two tones with frequencies $f_1 = 100\text{Hz}$ and $f_2 = 1000\text{Hz}$ to test the simulation output of the high-pass filter using ModelSim. Figure 12 shows the input to ModelSim (using genStimulus.m) and the output provided by ModelSim (using leSaida.m) after the VHDL simulation was performed.

Figure 13 shows the results obtained with the real circuit after synthesizing the VHDL code. In this case, unlike Figure 10, the output of the high-pass filter (blue trace) is a tone at 1000 Hz, corresponding to the high frequency component of the input signal. Therefore, this result complies with the expected response of the filter.

We have repeated these experiments with other frequencies, and the obtained results are also compatible

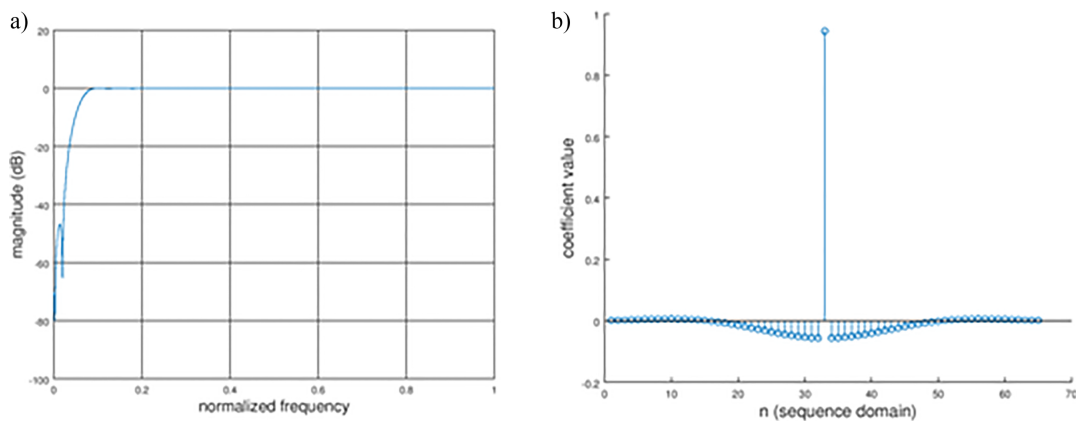


Figure 11: a) Frequency response of the high-pass filter. Magnitude (in dB) of the filter output as function of normalized frequency. b) Impulse response of the high-pass filter. Coefficient values as function of number of points.

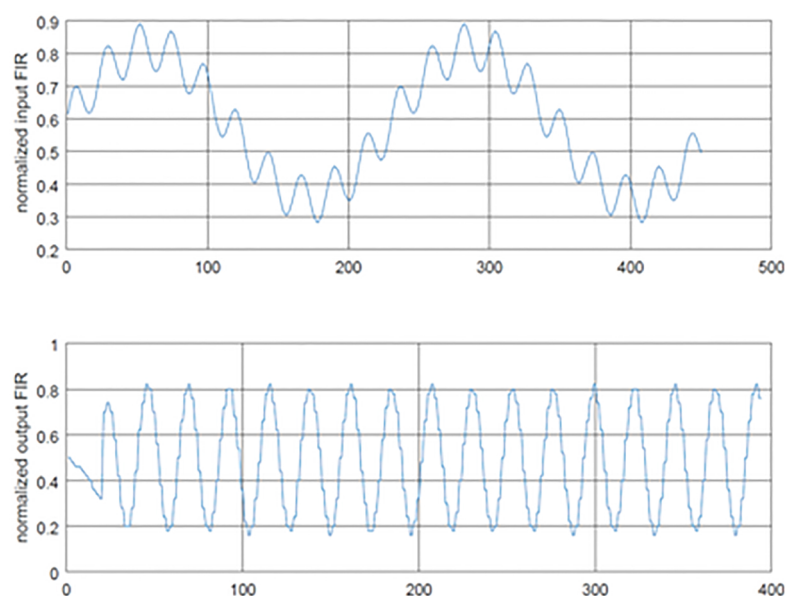


Figure 12: Input (above)/output (below) simulated wave forms of the high-pass filter.

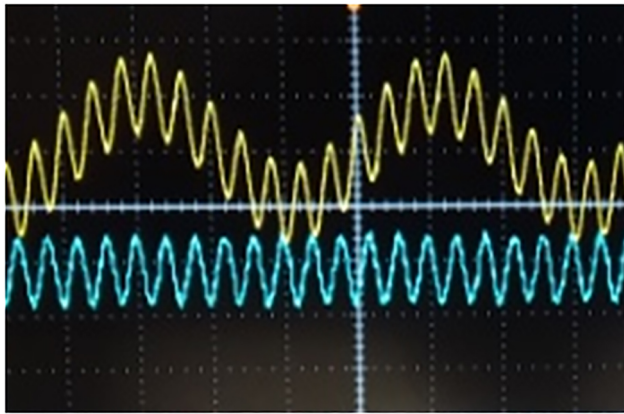


Figure 13: Input (yellow trace) /output (blue trace) obtained from high-pass filter real circuit captured from the oscilloscope screen. Each horizontal division is 2,5 ms.

with the expected filter response. The input signals will be restricted to maximum input frequency of 11.4 kHz due to the ADC hardware technology limitations employed (ADC 0804 – specification).

As shown in this section, the experimental results are consistent with the results of behavioral simulation. The simulation flow serves to anticipate experimental results before implementing the final circuit. In many situations, it is preferable to perform the simulation before implementation because it allows testing and validating the ideas and architectures prior to final implementation. The simulation gives confidence to test the actual circuit, showing what is supposed to be observed during actual operation for a given stimulus, without the risk of circuit failure or incorrect connections.

In addition, the experimental results well fitted with the simulated ones motivates the students, and enable them to relate the theory and practice, bringing the theoretical concepts, sometimes abstract, to the real world through measurements and observations in practical applications.

5. Conclusion

We have presented an approach for the implementation of electronic instrumentation for both didactic and research experiments. We have introduced the use of FPGA showing the steps for the simulation, validation and synthesis. As filters are one of the fundamental blocks for digital signal processing, we introduce and discuss the theory and practice to implement two FIR filters using the methodology proposed. Both simulation and implemented circuit present output signals response well fitted to the ideal filter function.

There are two main reasons to employ programmable logic devices: FPGA design is more flexible than other commonly used technologies; VHDL is a versatile and powerful language to describe the systems in different levels of abstraction. The design provides an easier con-

nection between theory and practice and facilitates the deepening in the understanding of the electronic instrumentation through the simulation and implementation. Thus, the implementations based on FPGA are interesting options for the development of electronic instrumentation in physics laboratory.

Moreover, this guide provides the basic steps to physics students interested in developing electronic instrumentation, encouraging them to learn and go deeper in this field of the technology.

References

- [1] R.V. Ribas, A.F. Souza and N. Santos, *Revista Brasileira de Ensino de Física* **20**, 293 (1998).
- [2] D. Halliday, R. Resnick and J. Walker, *Fundamentos de Física* (LTC, Rio de Janeiro, 2012), v. 4, 9^aed.
- [3] P.A. Tipler and G. Mosca, *Física para Cientistas e Engenheiros* (LTC, Rio de Janeiro, 2012), v. 2, 6^aed.
- [4] P.R. Kesten and D. L. Tauck, *Física na Universidade para as Ciências Físicas e da Vida* (LTC, Rio de Janeiro, 2015), v. 4, 1^aed.
- [5] R.L. Boylestad, *Introdução à Análise de Circuitos* (Pearson, São Paulo, 2012), 12^aed.
- [6] L.V. Piza, A.I.C. Arce, A.R.B. Tech and E.J.X. Costa, *Revista Brasileira de Ensino de Física* **35**, 1507 (2013).
- [7] J.C. Andrades, A. Schiappacassa and P.F. Santos, *Revista Brasileira de Ensino de Física* **35**, 2503 (2013).
- [8] G. Calderón, J. H. Muñoz and J.Y. Rivera, *Revista Brasileira de Ensino de Física* **40**, e2402 (2018).
- [9] L.R.M. Carvalho and H.S. Amorim, *Revista Brasileira de Ensino de Física* **36**, 3501 (2014).
- [10] M.A. Cavalcante, C.R.C. Tavoraro and E. Molisani, *Revista Brasileira de Ensino de Física* **33**, 4503 (2011).
- [11] H. Cordova and A.C. Tort, *Revista Brasileira de Ensino de Física* **38**, e2308 (2016).
- [12] L.A. Dworakowski, A.M. Hartmann, E.M. Kakuno and P.F.T. Dorneles, *Revista Brasileira de Ensino de Física* **38**, e3503 (2016).
- [13] A.M. Pereira, A.C.F. Santos and H.S. Amorim, *Revista Brasileira de Ensino de Física* **38**, e4501 (2016).
- [14] J.C. Viana and F. Arnold, *Revista Brasileira de Ensino de Física*, **39**, e2503 (2017).
- [15] A. Nayyar and E.V. Puri, in *3rd International Conference on Computing for Sustainable Global Development* (Institute of Electrical and Electronics Engineers, New Delhi, 2016), p. 1485.
- [16] E. Monmasson and M.N. Cirstea, *IEEE Trans. on Ind. Electr.* **54**, 1824 (2007).
- [17] B.D. Gamari, D. Zhang, R.E. Buckman and P. Milas, *Am. J. Phys.* **82**, 712 (2014).
- [18] J. Brody and C. Selton, *Am. J. Phys.* **86**, 412 (2018).
- [19] R. D'Amore, *VHDL: Descrição e Síntese de Circuitos Digitais* (Grupo Gen-LTC, São Paulo, 2000).
- [20] R.J. Tocci, N.S. Widmer and G.L. Moss, *Sistemas Digitais – Princípios e Aplicações* (Pearson Universidades, Rio de Janeiro, 2011), 11^aed.
- [21] Design Automation Standards Committee, *VHDL Language Reference Manual* (IEEE Standards, New York 2008), p. 1076.

- [22] Z. Navabi and S. Day, in *Proceedings Fourth Annual IEEE International ASIC Conference and Exhibit*. (Institute of Electrical and Electronics Engineers, Nova Jersey, 1991).
- [23] M.H. Hayes, *Schaum's Outline of Theory and Problems of Digital Signal Processing* (MacGraw-Hill, Nova York, 1999), 1^aed.
- [24] A.V. Oppenheim and R.W. Schaffer, *Discrete-Time Signal Processing* (Pearson, London, 2009), 3^aed.
- [25] D.J. Nowak and P.E. Schmid, IEEE Trans. on Electromag. Comp. **10**, 210 (1968).
- [26] A. Pertence, *Amplificadores Operacionais e Filtros Ativos* (Makron Books, São Paulo, 1996), 5^aed.