International Journal of Engineering Education Vol. 28, No. 3, pp. 651–662, 2012
Printed in Great Britain

0949-149X/91 $3.00+0.00

Open Core Hardware Description Practices for DSP Masters Degree Course*

JOSE DE JESUS RANGEL-MAGDALENO,[1] JESUS ROONEY RIVERA-GUILLEN,[2]
RENE DE JESUS ROMERO-TRONCOSO,[2] HAYDE PEREGRINA-BARRETO[2] and
JOSE PEDRO SANCHEZ SANTANA[1]
[1] Universidad Politecnica de Puebla, Juan C. Bonilla, Puebla, 72640, Mexico. E-mail: jjrangel@hspdigital.org
[2] HSPdigital—CA Mecatronica, Facultad de Ingenieria Campus San Juan del Rio, Universidad Autonoma de Queretaro, Qro., 76807, Mexico.

Nowadays, the industry is widely using embedded systems to match the requirements of low cost and high performance. Consequently, there is an increasing demand of engineers with high-level knowledge in software and hardware development. The contribution of this work is to propose a hardware signal processing course based on the pedagogical methodology known as design experiments, in which theory and practice are linked. The course covers the main topics of signal processing and includes the main hardware designs used in industry. The work focuses on the open core design used in those hardware descriptions through proposing several lab practices intended to get students trained in the hardware software co-design. MATLAB and field programmable gate arrays (FPGA) are used as the design tool and the synthesis platform, respectively. The work presents an easy-to-follow structure that can be easily understood by the students. The proposed course was validated from both pedagogical and industrial point of view through two surveys.

Keywords: Field Programmable Gate Arrays; education; Hardware Signal Processing; Hardware Description Language; MATLAB

1. Introduction

Nowadays, conventional general-purpose systems are not suitable for modern applications in controllers and data-processing. For this reason the industry is widely using embedded systems to match the requirements of low cost and high performance [1]. The embedded system design requires not only knowledge in software but also knowledge in hardware design. Taking this into account, universities around the world have made changes in their syllabus in order to attend this necessity. As an important part of any engineering studies, Hardware Signal Processing (HSP) courses should include hardware-software practice. Software design is covered using mathematical tools, which are necessary for learning digital signal processing (DSP) algorithms. The proposed mathematical tool for developing DSP algorithms in software is MATLAB because of its characteristics that allow clear, compact, and simple programming [2]. The hardware design is covered using very high-speed integrated circuit Hardware Description Language (VHDL) and field programmable gate arrays (FPGA). Hall and Anderson [3] performed a course using these tools; however, they did not present the hardware structure in their work. The Hardware Description Language (HDL) is an essential part in the development of the proposed course since it permits the description and simulation of circuits. There are plenty of software tools for hardware description. In this course the used software is GHDL (G Hardware Description Language) [4], due to the fact that the selected operation system is Linux. Nevertheless, other software tools for hardware description can be used. In the annex, simple instructions for downloading and installing GHDL under Linux are presented; GHDL can also be installed under Windows, it is free software for both platforms. The implementation of the hardware description can be done in FPGA. This technology satisfies some desired characteristics such as real-time processing, parallel processing, low cost implementation and open core design. Furthermore, it solves a principal query on the industrial applications of around 90% of senior students [5].

From the pedagogical point of view, the use of technology in education is a key factor for the student learning. Accordingly to [6], *design experiments* is a pedagogical methodology that perfectly links theory and practice; this methodology emphasizes on the development of theoretical ideas grounded in context of practice. In this perspective, the learning of signal processing requires to link theory and practice properly. This can be achieved through software and hardware implementation. However, such implementation requires to be rapidly accomplished due to the limited time of the course, which is limited in only to theory some cases. Thanks to the quick development of technology, nowadays it is possible to implement prototypes rapidly [7] using DSP processors or FPGA. FPGA technology allows to improve those courses that only cover a theoretical part or, at its best,

* Accepted 15 October 2010.

implementation of basic algorithms in microprocessors or DSP [8]. Moreover, some of these courses are developed for a specific device [7, 9] making them vendor dependent. The use of FPGA in the development of DSP courses has been successfully tested in [10], in which a dual-FPGA board is developed in conjunction with Xilinx University Program as a teaching platform. In another work, FPGA are used in a hardware system for remote accessible embedded system experiments [11]. A DSP course that proposes the development of an FFT under HDL as a project is presented in [12]; the disadvantage is the use of a commercial FFT core for its implementation. A low-cost FPGA development system based on Altera Max+PLUS software for teaching FPGA is presented in [13]; Altera has a University program that provides technology to the Universities to help them in hardware development education [14]. An approach to teaching design of application-specific architectures using HDL, physical synthesis tools and FPGA is presented in [15]. The use of MATLAB for teaching fixed-point and floating-point arithmetic using a DSP filter implementation is performed in [16].

The proposed course uses the pedagogical methodology of *design experiments*. This is achieved using HDL, MATLAB and FPGA in order to obtain an open-core hardware description practice for teaching hardware signal processing in Master's Degree. This course covers the main topics of DSP and includes the main hardware designs utilized in industry (filters, FFT and wavelet transform). The contribution of the work is the open-core design used in the hardware descriptions; this allows FPGA-family independency and increases the integration of other modules required in industry (AD/DA converters, interfaces, I/O signals, etc.). As literature shows, the use of MATLAB, HDL and FPGA is an essential part in the Master's curricula; therefore, the presented work contains lab practice in which the students develop MATLAB programs to understand the theory, and perform the hardware description of the algorithms in HDL, which is later implemented in an FPGA. The work presents straightforward structure that can be easily understood by the students. The lab practice is designed to use all the knowledge acquired in class. The MATLAB scripts, intended to facilitate the student comprehension of the subject, have been distributed through authors Website (see annex). The corresponding HDL codes can be found in [17] and in the authors Website (see annex).

2. Course details

The objective of the proposed work is to form high-level, creative, innovative students, with knowledge on HSP and FPGA; who have the capabilities and abilities to generate engineering solutions for solving industrial and academic necessity. This course is important in several engineering disciplines since it provide the bases for the development of mechatronic systems, instrumentation projects, embedded systems and monitoring and analysis of different phenomena. The course consists of teaching the theoretical ideas on signal processing grounded by enough practice aimed to get students trained in the hardware software co-design.

The proposed course has been satisfactorily implemented in Master's Degree taught by the HSP digital group within the Electromechanical Department at Universidad Autonoma de Queretaro, Mexico, and it is called Hardware Signal Processing. Some of the careers in which it has been implemented are Electrical Engineering, Electronics Engineering, and Mechatronic Engineering. The requirements to take the course are Digital Systems and Numerical Methods and especially knowledge in MATLAB, HDL and FPGA design.

The course's length is 18 class weeks; 6 hours weekly. Extra class periods should not exceed 36 hours. This course is mainly practical because in such a way the student's motivation is greater obtaining better performance. The work is divided into 40 hours theory and 68 hours practice. The course is designed to be evaluated through the suggested lab practice; therefore, handing-in a lab practice report for each of them is compulsory.

As shown in Table 1, the proposed course covers key topics taught on any typical HSP course. In this work a practical approach is given, including mathematical software and hardware description language management.

3. Course development

In this section, a description of the main themes on each topic is given. The course development has the structure shown in Table 1. At the beginning of the course, the professor describes the teaching and the evaluation methodologies to the students.

3.1 HSP Introduction

The course begins with a general theoretical introduction of hardware signal processing [18–20]:

- Signals, systems and signal processing
- Classification of signals
- The concept of frequency in continuous-time and discrete-time signals
- Analog-to-digital and digital-to-analog conversions
- Digital filter design

On the other hand, the software practice is highly

Table 1. Time schedule for teaching HSP course

Topic	Used Hours	Lab
1. HSP Introduction (9h)	12	Introduction to MATLAB (3h)
2. Number representations and fixed-point arithmetic (3h)	6	Example of conversion and use of MATLAB. (3h)
3. FIR Filter Theory Design of FIR filter. Equation of the filter. Filter Digital Structure. (5.5h)	18	Design the FIR filter using MATLAB. Perform the digital structure under HDL. Perform the simulations Implement a FIR filter. (12.5h)
4. IIR Filter Theory Design of IIR filter Equation of the filter Filter Digital Structure (5.5h)	18	Design the IIR filter using MATLAB. Perform the digital structure under HDL. Perform the simulations Implement an IIR filter. (12.5h)
5. Wavelet Theory Wavelet Digital Structure (9h)	28	Use the wavelet toolbox of MATLAB. Perform the digital structure under HDL. Perform the simulations. Implement a Wavelet Processor. (19h)
6.- FFT Theory FFT Digital Structure (8h)	26	Use the FFT toolbox of MATLAB. Perform the digital structure under HDL. Perform the simulations. Implement a FFT Analyzer. (18h)

important because the students consolidate the knowledge acquired in the theoretical classes. The practice is developed under MATLAB using common instructions to design filter windows due to their importance in signal processing [1, 21]. The first practice consists of a 32-order *hann* window simulation. During the lab hours the students must try out the instructions for different windows and hand a report in. The MATLAB scripts of these examples are in the annex named list_01.m.

3.2 Number representation and fixed-point arithmetic

The second point in the syllabus is the fixed-point arithmetic. Teaching the floating-point arithmetic is convenient as well; the recommended book for this topic is [22]. In this topic the main idea is to teach the finite word-length effects using both fixed-point and floating-point arithmetic. The corresponding practice is the representation of binary numbers using fixed-point and floating-point arithmetic and the determination of the quantization errors in both cases.

3.3 FIR Filter

After learning the different types of windows that can be used in filter design and the different number representations, the next point consists of designing an FIR (finite impulse response) filter. The methodology to design a filter involves software design followed by hardware design. A real-world application of FIR filters for jerk monitoring is presented by Morales *et al.* [23], and FIR filters are constantly used in industry applications. Thus, it is important for students to know how to design them. An FIR filter is defined by Equation (1).

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_{M-1} x(n-M+1) \quad (1)$$

where, M is the length of the input $x(n)$, $y(n)$ is the output, and b_k is the set of filter coefficients [18].

As a part of the practice, the students must design several examples of FIR filters with different characteristics under MATLAB. An example is to design, to simulate and to implement a low-pass FIR filter with a normalized cut-off frequency of 0.2, order 7 and a rectangular window. The design is achieved through list_02.m. Simulation consists of obtaining the filter response with a time-domain input signal composed of two pure sine signals with different frequency (high and low). The MATLAB script of this example is named list_03.m.

Once the students have become familiarized with the FIR filter structure, the next step of the practice is to perform the description of an FIR filter under HDL; this description corresponds to the hardware-design. Such objective requires giving an introduction to the filter digital structure through the realization of the MAC (Multiplier-Accumulator). The digital structure of the MAC is shown in Fig. 1, where it can be seen that the basic knowledge in hardware description is sufficient to carry out its development. In Fig. 1, A and B are the inputs, N is the number of multiplications, K is a pointer to ROM coefficients, Y is the output, STM is the process start signal and EOM is the final process signal.

The digital structure of an FIR filter is shown in

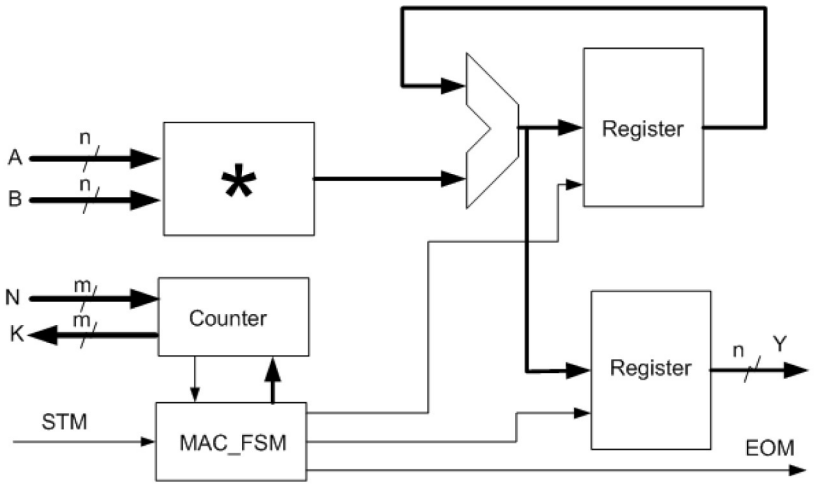

Fig. 1. Digital structure of a MAC.

Fig. 2, where X is the input signal, Y is the output signal, N is the order of the filter and STM is the signal to start the filter operation. The FIR filter architecture is based on the MAC unit.

The simulation of the last example is carried out in order to test the proper operation of the filter HDL description, the bit-width for the input and output signal is set to be 18 bits. The input signal is the same as the one used in MATLAB script list_03.m. Fig. 3 shows the hardware simulation of the filter. The student should compare the results obtained throughout the MATLAB simulation and the HDL simulation. As homework, it is proposed that the students determine the finite word-length effects between the filters developed under MATLAB and HDL.

3.4 IIR Filter

The design of filters with an infinite impulse response is possible when it is required by the application; these filters are the analog-world heritage and are named infinite impulse response (IIR) filters. The PID (proportional integral derivative) controller is an example of an IIR filter; the digital structure is similar as the one shown by Osornio *et al.* [24]. Discrete-time IIR filters are described by Equation (2).

$$y(k) = \sum_{i=0}^{n} a_i x(k-i) - \sum_{i=1}^{n} b_i y(k-i) \qquad (2)$$

where, $y(k)$ is the filter response, $x(k)$ is the input

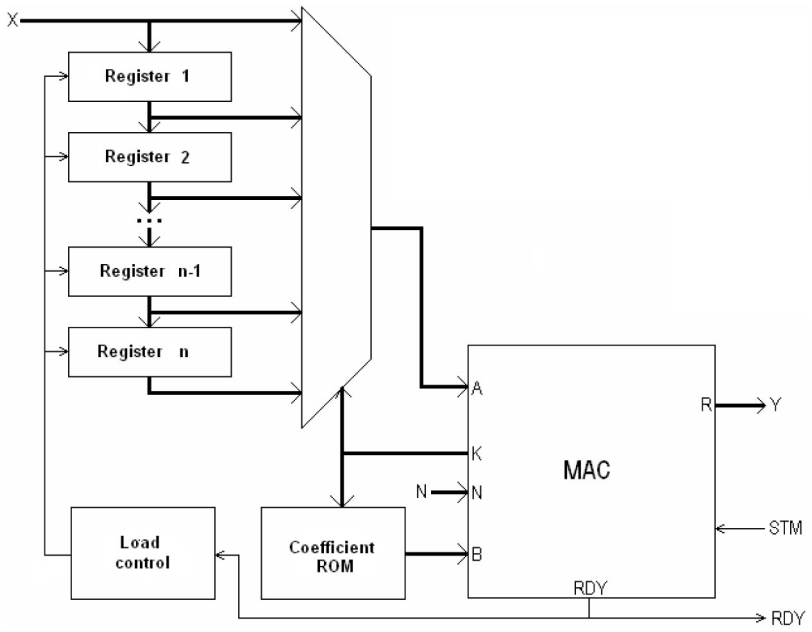

Fig. 2. FIR filter digital structure.

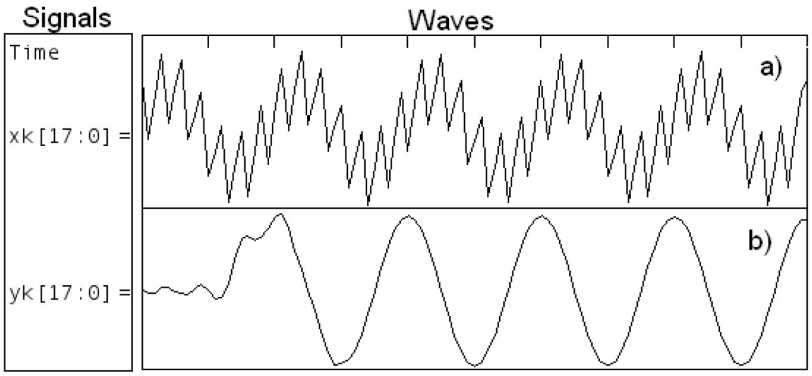

Fig. 3. Low-pass FIR filter simulation results under GHDL (a) input signal and (b) output signal.

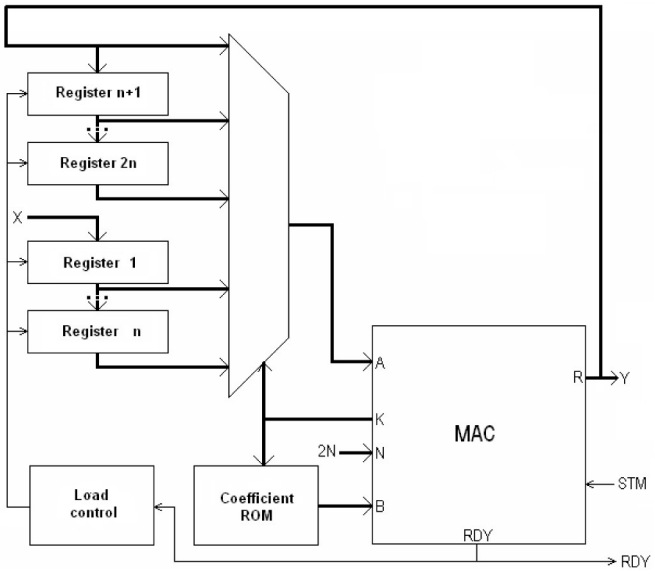

Fig. 4. IIR filter digital structure.

signal, b_i and a_i are the filter coefficients and n is the filter order. The function in MATLAB for software-design of an IIR filter depends on the type of filter to be designed. Firstly, the students are required to design a 4-order, low-pass inverse Chebyshev-type filter with a normalized cut-off frequency in 0.2. The MATLAB script for the design of this filter is named list_04.m in the annex. Secondly, in order to test the filter, the same input signal used in the FIR simulation is applied. The MATLAB script for this example is named list_05.m. Finally, the digital structure for the IIR-filter hardware implementation is shown in Fig. 4. This structure is similar to that presented for the FIR filter with some small modifications in the input registers. These modifications have been made because the IIR filter structure is recursive, unlike the FIR filter that is non-recursive.

The practical part of this topic consists of designing several IIR filters in MATLAB describing the digital structure presented in Fig. 4 and making the corresponding simulations. Fig. 5 shows the simulation results for a low-pass inverse Chebyshev type IIR filter with a cut-off frequency of 0.2. The input and output length is 18 bits. The input signal is the same as the FIR example.

The objective of the proposed digital filter practice is the implementation of each filter as shown Fig. 6. The implementation of the filters implies the use of analog-to-digital (ADC) and digital-to-analog (DAC) converters. The adequate converter must be selected according to the required resolution. Since an FPGA has a high-resource capacity, the control of several converters can be implemented on the same circuit. Likewise, the students are involved in the SOC (system on-a-chip) development. Fig. 6 shows the general block diagram for the filter development. It is highly important that students exploit the FPGA reconfigurability by synthesizing several types of filters.

As part of the hardware-software co-design, the MATLAB script given in the list_09.m automatically generates the HDL file containing the coeffi-

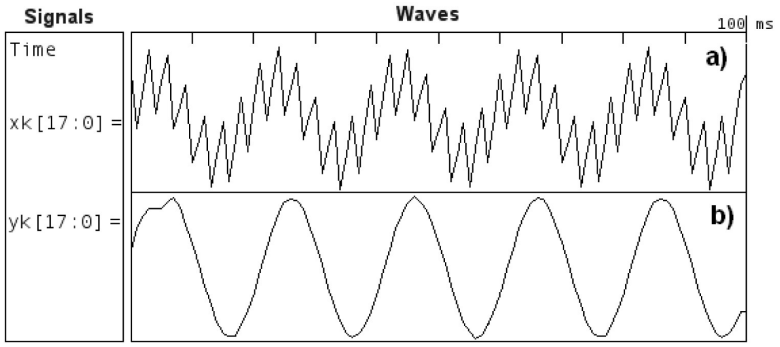

Fig. 5. Low-pass IIR filter under GHDL (a) input signal and (b) output signal.

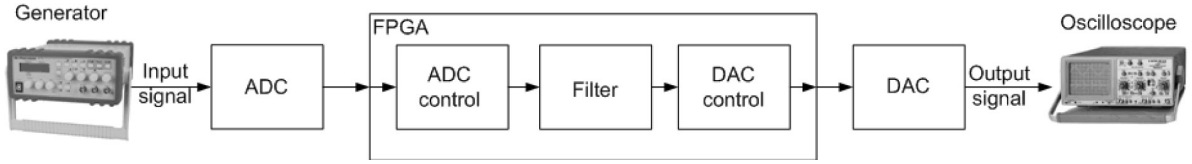

Fig. 6. Filter lab practice.

cient ROM. This script can also be used for generating wavelet transform and FFT coefficient ROM. The rest of the required files for the implementation are to be developed by the students in order to improve their HDL knowledge.

3.5 Wavelet transform

The wavelet transform subject is very extensive; consequently, it is quite hard to cover it completely in a standard HSP course. However, the main bases to understand and learn the correct functionality of the wavelet in hardware signal processing can be taught. Real-world examples of the use of wavelet transform are presented in machine-tools monitoring, fault detection in induction motors, dynamics parameters extraction, etc. [26–28]. The discrete wavelet transform (DWT) is calculated through a bank of low-pass and high-pass filters. This filter bank is better known as Mallat algorithm [28]. The algorithm consists of two phases: decomposition and reconstruction as shown in Fig. 7, where *fs* is the sampling frequency and N is the sample length. Fig. 7 represents the level 2 decomposition and its respective reconstruction.

The detail decomposition in a level L is given by Equation (3), where D_L is the detail of the signal at level L, DC_L are the detail decomposition coefficients corresponding to level L and $X(n)$ is the input signal [29].

$$D_L = DC_L X(n) \tag{3}$$

The approximation decomposition in a level L is obtained through Equation (4), where A_L is the approximation of the signal at level L, AC_L are the approximation decomposition coefficients corresponding to level L and $X(n)$ is the input signal [29].

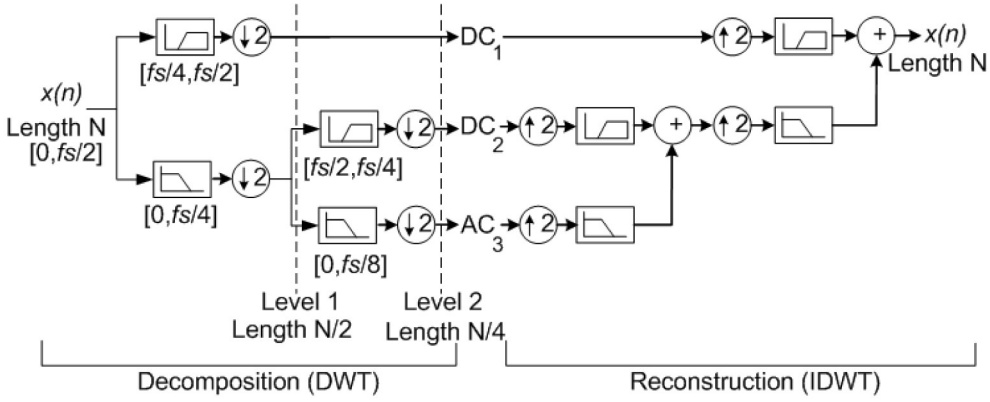

Fig. 7. Mallat algorithm.

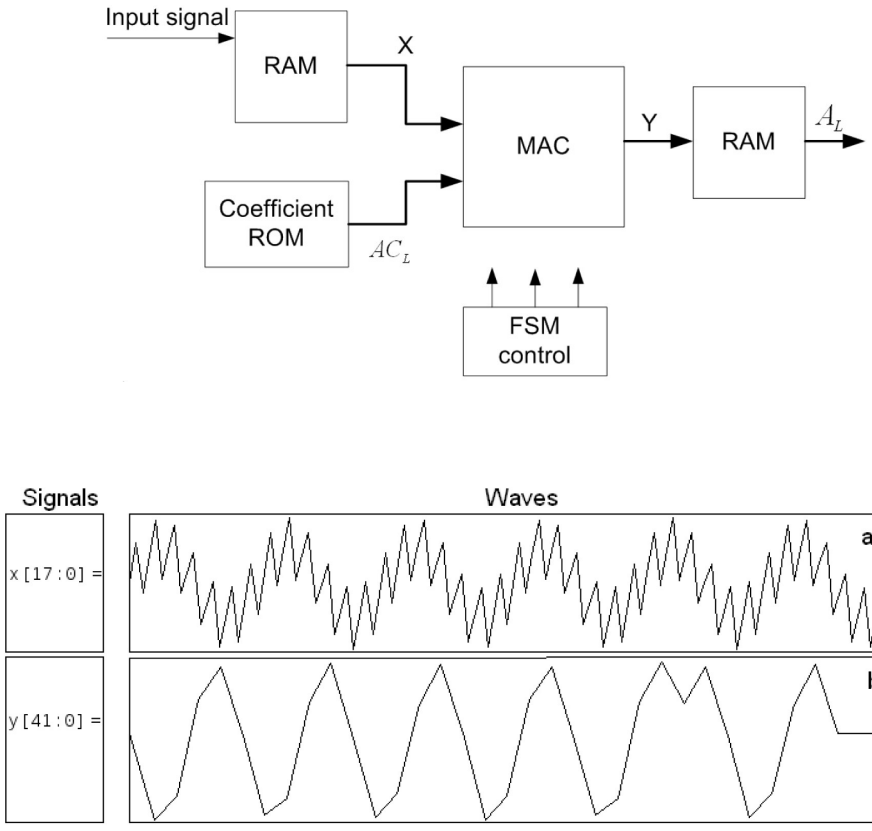

Fig. 9. Wavelet decomposition signal, (a) input and (b) decomposition.

$$A_L = AC_L X(n) \tag{4}$$

To obtain the approximation or detail decomposition coefficients in MATLAB, an example is presented in list_06.m.

The reconstruction of a signal, previously decomposed by the last function, is made using Equation (5) and Equation (6), depending on whether it requires rebuilding a detail or an approximation, respectively.

$$X(n) = DC_L{}' D_L \tag{5}$$

$$X(n) = AC_L{}' A_L \tag{6}$$

An example of both functions is showcased in list_07.m, which makes the decomposition at a level 2 approximation with a mother wavelet db6 and the reconstruction in the same level of approximation. It is recommended that the students make several examples with different types of mother wavelets and levels prior to hardware practice in order to understand the wavelet function more efficiently. The digital structure for the wavelet decomposition is shown in Fig. 8; as with filters, it is based on a MAC structure. Fig. 9 shows the input signal and its decomposition at level 2 of approximation with a *db6* mother wavelet. The input signal is the same of the filter examples.

The digital structure to perform the signal reconstruction is shown in Fig. 10. This structure is similar to that used in the decomposition with a small modification in the coefficients AC_L. Fig. 11 shows the reconstruction simulation, where the low-frequency component can be seen at the output. In this case the wavelet is used as a low pass filter. The reconstruction is made at level 2 of approximation with a *db6* mother wavelet.

The wavelet lab practice consists of developing a wavelet processor. The wavelet processor structure can be proposed in a similar way to the filters by using the wavelet core instead the filter module. Another way to do that is by replacing the digital-to-analog converter output signal by an RS232

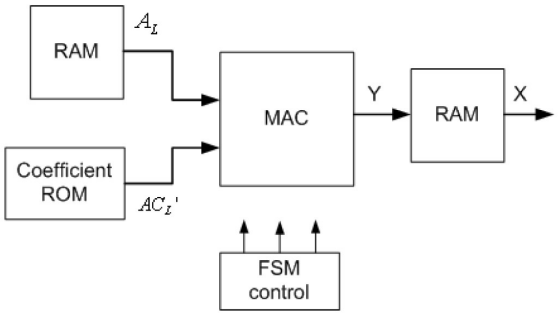

Fig. 10. Wavelet reconstruction digital structure.

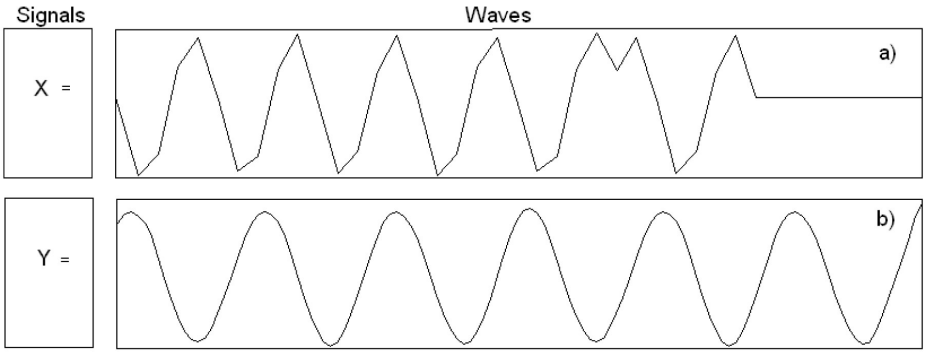

Fig. 11 Wavelet reconstruction (a) input signal and (b) output signal.

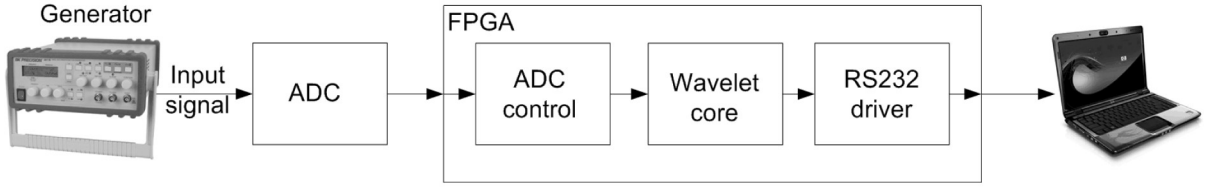

Fig. 12. Wavelet processor.

communication link with a PC for comparing the obtained results under MATLAB. The block diagram of Wavelet processor is presented on Fig. 12.

3.6 Fast Fourier Transform

The Fast Fourier Transform (FFT) is an important tool for hardware signal processing and it is commonly used in many industrial monitoring applications such as spectrum analyzers, fault detection in induction motors, etc. [12, 30, 31]. The discrete Fourier transform (DFT) is defined by Equation (7) [17], in which $x(n)$ is a time-domain data sequence for an N-point data set, n and k are the discrete-time and frequency indexes respectively, and the transformation kernel W is given by Equation (8).

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad 0kN-1 \tag{7}$$

$$W_N^{nk} = \exp\left(-j2\pi \frac{nk}{N}\right) \tag{8}$$

The arithmetic complexity of the DFT algorithm becomes a significant factor with an impact in the global computational cost. Cooley and Tukey [18] developed the radix-2 FFT algorithm to reduce the computational load of the DFT, the FFT is the optimized implementation algorithm for the DFT computation, it lowers the arithmetic complexity from $O(N^2)$ to $O(NlogN)$.

The algorithm used in the proposed course is the Decimation-In-Time (DIT) version of Cooley and Tukey's FFT algorithm. The main block for the FFT computation is the butterfly processor which contains complex fixed-point multipliers and adders. The fully combinational block diagram for the butterfly is shown in Fig. 13, where $g_0 = a_0 + (a_1c - b_1s)$, $h_0 = b_0 + (a_1s\text{-}b_1c)$, $g_1 = a_0 - (a_1c\text{-}b_1s)$, and $h_1 = b_0 - (a_1s + b_1c)$. Where a_k and b_k are the real and imaginary part of the input data, respectively; c and s are the real and imaginary part of the transformation kernel.

Once the students have developed the butterfly structure, the next step is to integrate the necessary

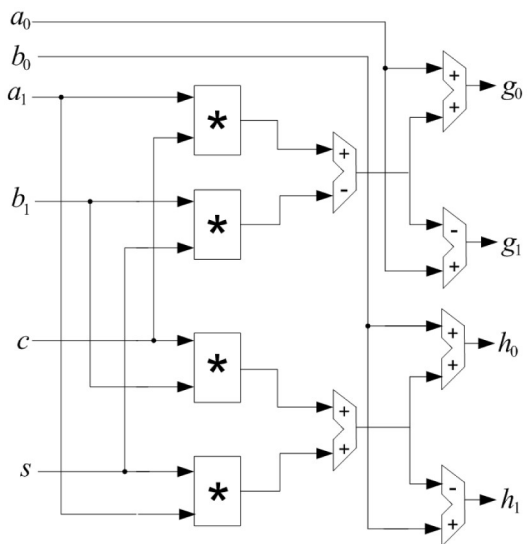

Fig. 13. Radix-2 butterfly block diagram.

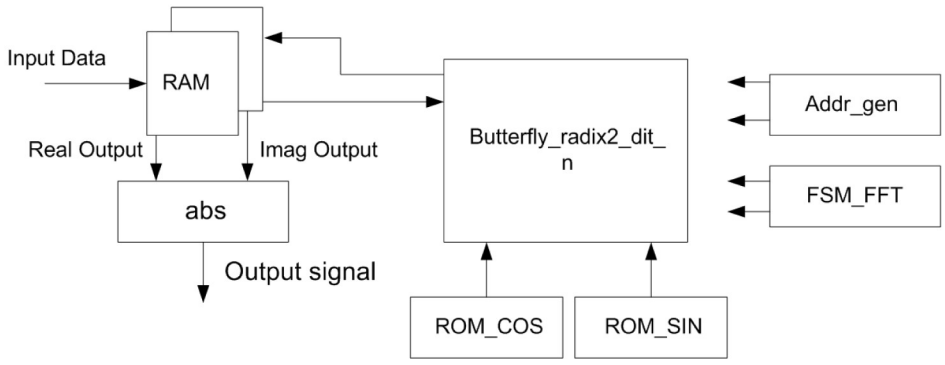

Fig. 14. FFT block diagram.

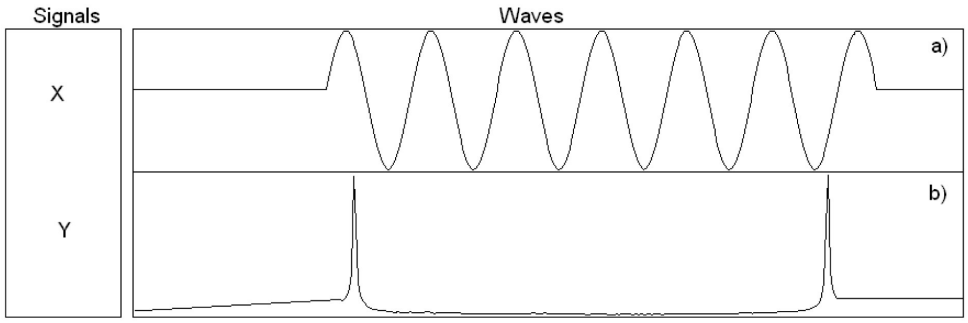

Fig. 15. 256-point FFT in GHDL (a) input signal and (b) output signal.

modules in order to calculate the FFT; these modules are two RAM memory blocks, two look-up tables (LUT) containing the cosine and sine twiddle values, an address generator and a finite-state machine control. Fig. 14 presents the general FFT block diagram.

The corresponding practice for this topic consists of the description of a 256-point FFT. A MATLAB example of a 256-point FFT is presented in list_08.m, in which an input signal of 25 Hz with a sampling frequency of 1000 Hz is used. Fig. 15 shows the same example but now for an FFT described on GHDL.

The FFT practice consists of developing an FFT analyzer, the importance of the FFT analyzer's use is presented in [33]. For the FFT analyzer the use of a VGA monitor for displaying the signal spectrum is proposed. As in the former practice, the converters and modules developed along the course are now used. The development of the VGA display is absolutely relevant; the possibility to use the PC or oscilloscope for displaying results is provided to the students. The block diagram is shown in Fig. 16.

4. Educational assessment

With the aim of evaluating the course's benefits, two surveys are presented and discussed. The first survey is designed to evaluate the effectiveness of the course in teaching and learning. It is applied to 20 master degree students who took the course 'Hardware Signal Processing'. The second survey was applied to six regional industries focused on the field of Mechatronics, the objective of which is to evaluate how useful it is for the regional industry that students have a wide knowledge in hardware-software co-design. In both surveys, respondents were asked to provide a percentage of agreement for each

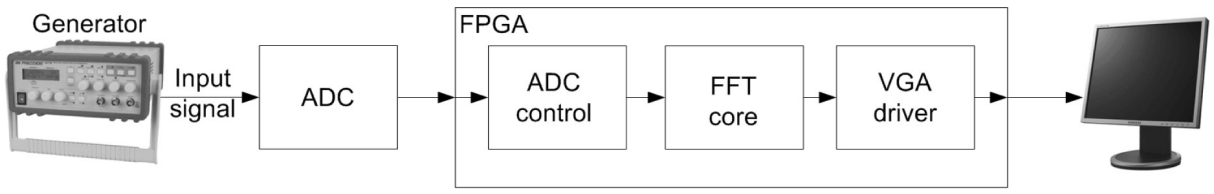

Fig. 16. FFT analyzer.

Table 2. Master's Degree course survey

Question	Average (%)
Is the course easy to follow?	92
Does pedagogical structure of the course benefit the student learning?	88
Does it link theory and practice successfully?	95
Are the contents of the course focused on many examples and professional applications?	90

Table 3. Survey on the value of hardware-software co-design for industry

Question	**Average (%)**
How much is the signal processing used in your industry?	89
Is it important the use of hardware and software processing?	95
Are embedded systems frequently utilized in your industry?	88
How useful is for your industry that students have knowledge in hardware-software co-design?	96

question, being 100% full agreement and 0% full disagreement.

Table 2 shows the average of Master's Degree students' responses. The survey shows that students found the course easy to follow and well balanced between theory and practice. Table 3 presents the results of the second survey, finding that the regional industry have a wide interest in professionals with knowledge in hardware and software processing.

5. Conclusions

In this work, the design of laboratory practice for teaching hardware signal processing using an open core hardware description is presented. From the pedagogical point of view, the proposal is based on the *Design experiments methodology* where a link between theory and practice is proposed. The course was organized considering both, software and hardware. Major weight was given to the lab practice encouraging the students to validate their theoretical knowledge. The utilized pedagogical methodology gives students the skills to propose engineering solutions in both academic and industrial necessities. The use of FPGA opens the action field for the students and allows them to use their knowledge on quick prototyped, low cost, and SOC solutions. The proposed course was validated from both pedagogical and industrial point of view through two surveys. The first one (Table 2) shows that students agree in a 95% that the course successfully links theory and practice. Moreover, the 90% agree that the course is focused on real applications useful for solving problems in the industry. The second survey (Table 3) was intended to obtain feedback from the regional industry about the usefulness to prepare students in the hardware-software co-design. Industries that answered the questionnaire agree in a 95% that hardware and software processing are important skills that engineers need to feature. Moreover, the 96% agree that the knowledge in hardware-software co-design is useful for solving problems in the industry.

Acknowledgments—This paper was supported by Universidad Politecnica de Puebla through the PIFI project. The authors would like to thank Altera Corporation for the donation of a DE2–70–3 development kit.

References

1. R. Niemann, *Hardware/Software Co-Design for Data Flow Dominated Embedded Systems*, Springer Publishing Map, ISBN: 0792382994, USA, 1998.
2. A. Gilat, *MATLAB: An introduction with applications*, Wiley, Hoboken, NJ, 2008.
3. T. S. Hall and D. V. Anderson, A framework for teaching real-time digital signal processing with field-programmable gate arrays, *IEEE Trans. Educ.*, **48**(3), 2005, pp. 551–558.
4. Detailed Information Available: http: //ghdl.free.fr/.
5. F. S. Cohen, A. Petropulu, G. Georgiou and W. Ibrahim, Multimedia digital signal processing laboratory, *Comput. Appli. Eng. Educ.*, **8**, 2009, pp. 209–215.
6. P. Mishra and M. J. Koehler, Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge, *Teachers College Record*, **108**(6), 2006, pp. 1017–1054.
7. W. S. Gan, Y. K. Chong, W. Gong and W. T. Tan, Rapid prototyping system for teaching real-time digital signal processing, *IEEE Trans. Educ.*, **43**(1), 2000, pp. 19–24.
8. W. S. Gan and M. Kuo, Teaching DSP software development: form design to fixed-point implementations, *IEEE Trans. Educ.*, **49**(1), 2006, pp. 122–131.
9. W. S. Gan, Teaching and learning the hows and whys of real-time digital signal processing, *IEEE Trans. Educ.,* **45**(4), 2002, pp. 336–343.
10. C. J. Kief, M. S. Pattichis, L. H. Pollard, G. A. Vera and J. E. Parra, An XUP_UNM educational platform: A dual-FPGA platform for reconfigurable logic, *Comput. Appli. Eng. Educ.*, **17**, 2009, pp. 232–239.
11. X. Yue, E. M. Drakakis, J. Harkin, M. J. Callaghan, T. M. Mcginnity and L. P. Maguire, Modular hardware design for distance- internet embedded systems engineering laboratory, *Comput. Appli. Eng. Educ.*, **17**, 2009, pp. 389–397.
12. T. Sansaloni, A. Perez-Pascual, V. Almenar, J.F. Toledo and J. Valls, FFT spectrum analyzer project for teaching digital signal processing with FPGA devices, *IEEE Trans. Educ.*, **49**(1), 2006, pp. 122–131.
13. I. A. Hack and J. Haberly, Low cost FPGA development system for teaching advanced digital circuits, *Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition*, 2001.
14. Detailed Information Available: http: //www.altera.com.
15. J. M. P. Cardoso, A teaching strategy for developing application specific architectures for FPGAs, *International Journal of Engineering Education*, **24**(4), 2008, pp 833–842.

16. G. Woon-Seng and S. M. Kuo, Teaching DSP software development: from design to fixed-point implementations, *IEEE Trans. Educ.*, **50**(3), 2007, pp. 229–235.
17. R. J. Romero-Troncoso, *Electronica digital y logica programable*, Universidad de Guanajuato, ISBN 968-864-449-8, Mexico, 2007.
18. J. G. Proakis, and D. K. Manolakis, *Digital signal processing, principles and applications*, Fourth Edition, Prentice-Hall, New Jersey, USA, 2006.
19. A. V. Oppenheim and R. W. Schafer, *Digital signal processing,* Englewood Cliffs, Prentice-Hall, New Jersey, 1989.
20. R. E. Blahut, *Fast algorithms for digital signal processing*, Reading, MA: Addison-Wesley, 1985.
21. MATLAB Help, MathWorks Inc. Available online at www.mathworks.com.
22. B. Parhami, *Computer arithmetic algorithms and hardware designs*, First Ed., Oxford University Press, Santa Barbara, CA: 2001.
23. L. Morales-Velazquez, R. J. Romero-Troncoso, R. A. Osornio-Rios and E. Cabal-Yepez, Sensorless jerk monitoring using an adaptive antisymmetric high-order FIR filter, *Mech. Syst. Signal Process.*, **23**, 2009, pp. 2383–2394.
24. R. A. Osornio-Rios, R. J. Romero-Troncoso, G. Herrera-Ruiz and R. Castaneda-Miranda, The application of reconfigurable logia to high speed CNC milling machines controllers, *Control Eng. Pract.*, **16**, 2008, pp. 674–684.
25. J. J. Rangel-Magdaleno, R. J. Romero-Troncoso, R. A. Osornio-Rios, E. Cabal-Yepez and A. Dominguez-Gonzalez, FPGA-Based vibration analyzer for continuous CNC machinery monitoring with fused FFT-DWT signal processing, *IEEE Trans. Instrum. Meas.*, DOI: 10.1109/TIM.2010.2047130.
26. J. Antonino-Daviu, P. Jover-Rodríguez, M. Riera-Guasp, M. Pineda-Sanchez and A. Arkkio, Detection of combined faults in induction machines with stator parallel branches through the DWT of the startup current, *Mech. Syst. Signal Process.*, **23**, 2009, pp. 2336–2351.
27. J. J. de Santiago-Perez, R. A. Osornio-Rios, R. J. Romero-Troncoso, G. Herrera-Ruiz and M. Delgado-Rosas, DSP algorithm for the extraction of dynamics parameters in CNC machine tool servomechanisms from an optical incremental encoder, *Int. J. Mach. Tool. Manuf.*, **48**, 2008, pp. 1318–1334.
28. S. G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, *IEEE T. Pettern. Anal.*, **11**(7), 1989, pp. 674–673.
29. I. Daubechies, Orthonormal bases of compactly supported wavelets, *Commun. Pur. Appl. Math.*, **41**(7), 1998, pp. 909–996.
30. L. M. Contreras-Medina, R. J. Romero-Troncoso, E. Cabal-Yepez, J. J. Rangel-Magdaleno and J. R. Millan-Almaraz, FPGA-Based multiple-channel vibration analyzer for industrial applications in induction motor failure detection, *IEEE Trans. Instrum. Meas.*, **59**(1), 2010, pp. 63–72.
31. J. J. Rangel-Magdaleno, R. J. Romero-Troncoso, R. A. Osornio-Rios, E. Cabal-Yepez and L. M. Contreras-Medina, Novel methodology for online half-broken-bar detection on induction motors, *IEEE Trans. Instrum. Meas.*, **58**(5), 2009, pp. 1690–1698.
32. Detailed Information Available: http: //www.xilinx.com.
33. R. C. Woods, Use of software for real time spectrum analysis, *International Journal of Engineering Education*, **23**(4), 2007, pp 799–807.

Appendix

MATLAB files and HDL files can be downloaded from **www.hspdigital.org** in download section

MATLAB Files

list_01.m	Example functions 'window' and 'freqz'
list_02.m	Example function 'fir1'
list_03.m	Example function 'filter'
list_04.m	Example function 'cheb2'
list_05.m	Example function 'cheby2' and 'filter'
list_06.m	Example function 'wfilters'
list_07.m	Example function 'wavedec'
list_08.m	Example function 'fft' and 'abs'
list_09.m	Example of ROM-coefficients HDL file automatic generation.

GHDL Installation
Taken from http://ghdl.free.fr/INSTALL
Install file for the binary distribution of GHDL.

GHDL is Copyright 2002 - 2010 Tristan Gingold.
GHDL is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

The binary are installed in /usr/local directory. You cannot change this
default location, unless you set links.

You must be root to install this distribution.

To install ghdl:
$ su
# tar -C / -jxvf @TARFILE@.tar.bz2

Note: you must also have a C compiler and zlib installed.

There is a mailing list for any questions. You can subscribe via:
https://mail.gna.org/listinfo/ghdl-discuss/

Tristan Gingold.

Jose de Jesus Rangel-Magdaleno is currently a Professor with the Universidad Politecnica de Puebla. His research interests include mechatronics, instrumentation and image processing.

Jesus Rooney Rivera-Guillen is currently working toward the Mechatronic Ph.D. degree at the University of Queretaro, Queretaro, Mexico. His research interests include mechatronics, robotics, and control.

Rene de Jesus Romero-Troncoso is currently a Head Professor with the University of Guanajuato and an Invited Researcher with the University of Queretaro. His fields of interest include hardware signal processing and mechatronics.

Hayde Peregrina-Barreto is currently a Professor with the Universidad Politecnica de Puebla. His fields of interest include image processing and mechatronics.

Jose Pedro Sanchez Santana is currently a Professor with the Universidad Politecnica de Puebla. His fields of interest include Mechatronics, Automatization and Systems Control.