A soft-core based Lab for an Introductory Microprocessors Course

L. Etcheverry, J. Oliver, J. Pérez Acle
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
(letcheve, jpo, julio) @fing.edu.uy

Abstract—**This paper presents a change in the methodology of the laboratory activities in an undergraduate microprocessor systems design course focused on I/O methods and small microprocessors system integration. The laboratory was changed on two main aspects: firstly the assignments are done at home, shortening the time between design activity and experimental verification; secondly a soft-core processor synthesized on an FPGA is exploited to enable the students to exercise hardware design activities on a running system.**

A suite of development tools combining vendor, third party and in house developed software and hardware cores is presented.

These changes are still a work in progress. A transitional version of the course was held during 2011 and the second edition incorporating the rest of the changes is planned for the first half of 2012. Some preliminary results are reported.

Microprocessors; education; lab at home; FPGA

I. Introduction

The aim of this paper is to present a change in the methodology of the laboratory activities in a microprocessor systems design course.

This is an introductory undergraduate course, intended to suit students that will follow different academic paths within electrical engineering [1]. The course sets the emphasis on the integration of peripherals in small microprocessor systems and the I/O methods to access them, avoiding advanced processor architecture concepts and internal processor design details.

The course lasts 15 weeks with a weekly schedule of 3 lecture hours, 1.5 exercise discussion hours and several laboratory assignments.

The course was modified on two main aspects.

Firstly, the lab assignments are done at home instead of the University labs as it used to be. This “lab at home” methodology has been previously reported in [2][3] for a logic design course. It has several advantages when it comes to time and work place management, both for students and faculty. Also, it shortens time between the design activity and experimental verification, resulting in a greater motivation for the students.

Secondly, the laboratory kit used in the new lab assignments is based on a “system on a chip” synthesized on an FPGA which allows to introduce hardware design tasks into the lab assignments. Formerly the kit was a fixed microcontroller system which restricted the student responsibilities to the understanding of the provided hardware.

These changes are being implemented in two stages. The first one was held on the 2011 edition of the course, during which the new FPGA based kit and software development tools were perfected, the “lab at home” methodology was introduced, but the hardware design was still provided by the teachers. On the second edition of the course, starting on March 2012, the process will be completed.

The use of soft-cores synthesized into an FPGA for small microprocessor systems education is relatively new.

An introductory microprocessor course based on Xilinx development tools for the MicroBlaze soft-core is reported in [4][5]. Hardware design is done using the Embedded Developer’s Kit (EDK) from Xilinx,

In [6] a sequence of an introductory microprocessor course plus an advanced embedded systems course is described. The authors point out that for the introductory course students, the number and complexity of tools required to get a soft-core system running from scratch quickly becomes overwhelming distracting them from the learning experience. To avoid this problem they give students the bitstream of the system hardware ready to download to the FPGA. Consequently, most of the advantages of having a flexible system are lost. As shown later, the preferred alternative for the course presented here is to use a simpler processor and schematic capture design entry in order to simplify the toolset required without losing the ability to introduce changes to the hardware by the students.

In [7][8] an open source FPGA platform is presented. The platform is used with soft-core processors in a series of courses on embedded systems. The hardware design entry method used is Verilog HDL with open-source tools for synthesis.

The first examples of “lab at home” activities were due to PC popularization and related to software courses. Then, following the spread of microprocessors, take-home educational kits were introduced for microprocessors and control courses [9][10][11]. Later on, programmable logic, combined with the availability of free software tools, made it possible to introduce take-home kits to digital design courses. An early experience is briefly presented in [12], where kits based on breadboards were distributed. However, software tools were not freely available outside school in this case.

There are different approaches that also enable students to perform hands-on home activities including: mounting electronic circuits using components kits for later experimentation in class [13], the use of real laboratory instrumentation at home [14], or the costly loaning of equipment during the course [15].

The alternative presented in this work, which is based on the massive lending of low-cost programmable logic hardware kits to students for the whole semester, keeps every characteristic of the real hands-on hardware experimentation and at the same time has all the potential of a distance learning tool. Programmable logic is a technology very suitable for teaching digital logic in traditional labs [16][17], and is also widely used in industrial applications.

The paper is organized as follows. Section II details the course characteristics and the laboratory methodology used, Section III describes the hardware and software tools developed. Finally in section IV some conclusions and future activities are presented.

II. Course Description

This new learning experience was implemented for the microprocessors introductory course, a core electrical engineering course taught to more than 120 students. This course integrates lectures, problem discussion groups and laboratory instruction.

As mentioned in the Introduction section, the course is focused on peripherals and memory integration and I/O methods on small microprocessor systems. The well known Z80 architecture from Zilog is the processor architecture used during the course. The fact that a processor with no pipeline is used greatly simplifies the understanding of the fetch-decode-execute sequence by studentswho are getting in touch with microprocessor systems for the first time. Despite of its simplicity, the selected architecture allows to introduce the students to some important concepts that are of common use in newer processors. Good examples of this concepts are priority arbitration and vectorized management of interrupt requests.

Before the methodology change presented on this paper, the laboratory activity of the course consisted of two mandatory assignments with pass/fail grading. Course grading was based on the marks obtained on a midterm and a final test. A fixed microcontroller kit was used in the lab assignments, and consequently the student had no opportunity to modify the hardware design. The effort demanded of the students was limited to the understanding of the given hardware and writing down the programs to control it properly.

The course was modified on two main aspects. Firstly, lab assignments are now done at home by the students as reported in [2][3] for a logic design course. Secondly, the laboratory kit is not "fixed hardware" anymore, but it is a "system on a chip" synthesized on an FPGA.

Even though the existing course syllabus leans heavily on hardware integration (glue logic, bus timings, address decoding, etc.) there was no way for students to put this knowledge into practice. The introduction of programmable logic devices now allows students to have greater control on the hardware design surrounding the microprocessor.

The new "system on a chip" kit avoids restricting the student to writing down the software, and enables the inclusion of hardware design tasks on the assignments. This seeks to reinforce, by means of experimentation, concepts that were only introduced theoretically in the former course methodology, which is the most relevant novel aspect of the proposed methodology. Students will now integrate existing peripherals and add the necessary glue logic for their designs. Given the introductory nature of the course, schematic oriented design is chosen over an HDL based approach.

It is worth noting that FPGA vendors such as Altera and Xilinx already provide tools which greatly simplify development of "system on a chip" designs. While these tools are fairly well integrated and provide a great enhancement in productivity for day-to-day work, they tend to hide a lot of complexity regarding the integration of hardware to a microprocessor based system.

Although the lab methodology is very similar to the one reported on [3] for a logic design course, it is worthy to describe it with some detail here for the sake of paper coherence.

The main characteristic of this new methodology is that most of the students' laboratory work is done at home with real hardware. In order to achieve this, at the beginning of the semester, students form groups of three and each group is given a hardware kit (Fig. 1) that they keep until the end of the course.

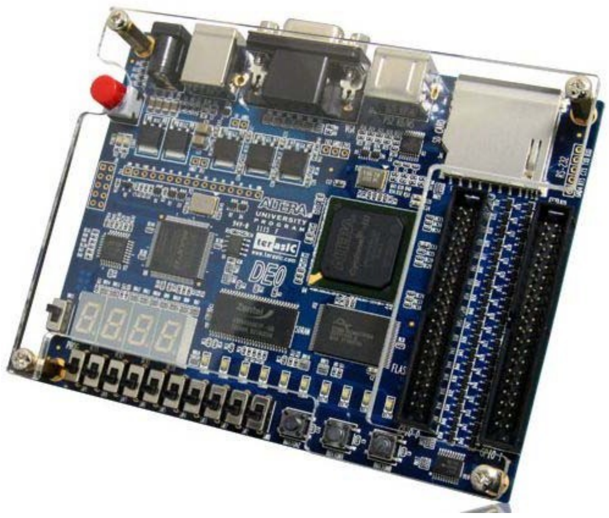
Figure 1. Hardware Kit, DE0 board (source Terasic Technologies Inc.)

There are three assignments during the semester, with deadlines known since the beginning of the semester. The students in each group are expected to work together, at home using their own computers, or at the university in the PC rooms. The teacher's support is provided during this stage. For each assignment the groups must hand in a written report and orally present their designs to a teacher. At these evaluation sessions teachers also ask them questions and grade their work.

The kits consist of a DE0 board, the soft-core, design software tools and the user manual. The students know the board and the Quartus software from a previous digital design course.

Due to the low cost of the kit components, students can afford to pay for replacements in the event that they break or lose them.

In the course Web page students can find data sheets, tutorials, and links to all the software they need to design the system hardware, program the FPGA device, write the programs and debug them running on the soft-core. The tutorial was designed specifically for these boards and guides students through the complete design process of a simple example.

At the end of the semester, when students return the kits, the boards are programmed with a design that tests all of the board features to verify that these are still working properly.

A. Assignments

The assignment sequence consists of three assignments with increasing difficulty. Each assignment reuses the subroutines and modules developed in the previous ones. As explained in the introduction section, during the first edition of the new course on 2011 the hardware design of the system was provided to the students with the assignment statement. On the second edition starting on March 2012, the hardware flexibility provided by the new lab platform is exploited, and the proper addition to the system of some of the peripherals used are now responsibility of the students.

In the first assignment the students take contact with the FPGA board, the soft-core and the development tools. They develop small subroutines to make some data conversions (bcd to seven segment, binary to bcd, etc.) and to access simple I/O devices (push-buttons and displays).

In the second assignment the students must add a handshake controlled I/O device to receive data from a serial PS2 keyboard, and they must write and validate input subroutines.

Finally, in the third assignment the concepts of interrupts and programmable peripherals are introduced: a programmable timer is used to generate periodic interrupts and the whole system is used to develop a clock that shows the time in a seven segment display. Time can be set using the keyboard interface validated in the second assignment.

B. Evaluation methodology

The three assignments are distributed in the course of the semester. Although the groups are composed of three students that work together during the entire course, at the end of the semester each student will have an individual laboratory grade which will be part of the final grade.

The lab assignments are evaluated orally during an approximately one-hour long session with one teacher at a scheduled date.

On these occasions, the group shows the teacher a demo of their design with a working system implemented on the board. Before this presentation, students must write and submit a lab report that should include the information required in the assignment.

This process helps the teacher to discover difficulties and errors that must be corrected or explained during the evaluation. Each member of the group explains one particular part of the project and then answers some oral questions, which often require a slight modification to the designed circuit to make it work in a different fashion. If the student has a good comprehension of the problem he/she should easily solve it. A checklist containing the main concepts that must be evaluated is available for the teachers to guide them during this process and homogenize the evaluating criteria.

This evaluation method is also a learning experience since it allows students to reinforce good concepts and correct mistakes.

The evaluation also lets teachers detect any uneven distribution of work among the students in a group and cheating between groups.

III. Development tools used

To provide a working development environment the "lab at home" kit integrates open-source hardware cores as well as free software. Hardware cores include a Z80-compatible CPU core, I/O peripherals, timers and debug modules. Software tools include Z80 GNU toolchain (assembler, linker and debugger), a QEMU-based emulator and debug support programs.

Cores and software used in the kit categorize as pre-existing third-party modules, modified third-party modules and in-house developed modules. Table I and Table II summarize each module's features as well as its required modifications and added features.

TABLE I. Hardware Modules

Module	*Origin*	*Added Features*
T80 CPU	3rd party	–
JTAG UART	3rd party (modif.)	T80-compatible interface
Configurable Timer/Counter	In-house	–
Mode-2 Interrupt Controller	In-house	–

TABLE II. SOFTWARE MODULES

Module	*Origin*	*Added Features*
GNU Binutils	3rd party	–
QEMU-Z80	3rd party (modif.)	GDB support
GDB-Z80	In-house	ASM source level support
GDB-Z80 monitor (stub)	In-house	software breakpoints support
JTAG Connector	In-house	TCP to JTAG UART tunnel

A. *T80-based SoC system*

The kit is based on the T80 CPU, a configurable Z80 CPU core, freely available as a VHDL design.

A T80-based SoC system integrating the T80-CPU core, on-chip memory, general-purpose peripherals and a JTAG UART endpoint, is synthesized for an Altera Cyclone III FPGA and downloaded to the DE0 board in order to obtain a working development kit.

B. *Software development environment*

To help students in the development of their lab assignments software, they are provided with a suite of several development tools. The development suite is based on the GNU toolchain and other software such as an emulator and debugger-to-target communication facility.

1) GNU Binutils

GNU binutils, specifically Z80 assembler and linker, are used to produce application binaries capable of running both on the T80-SoC as well as the QEMU-Z80 emulator.

2) QEMU-Z80

QEMU-Z80 is a Z80 target for the QEMU emulator and virtualizing software. It emulates a Z80 based system and allows for the execution of Z80 code on a current x86 PC. The QEMU-Z80 allows students to test and debug their application code even if they don't have access to the hardware kit at the moment. In addition, the emulator has the potential to be used for parallelizing testing and debugging among students within each group by decoupling debugging from the hardware kit.

3) GDB-Z80 and monitor (GDB stub)

The GDB-Z80 is a Z80 port of the popular GNU Debugger Project. GDB-Z80 allows students to run, test and debug their application code both on actual hardware (T80-SoC) and on the QEMU-Z80 emulator. At the moment, GDB-Z80 supports assembler source level debugging and software breakpoints. A unified target debugging interface allows for seamless switching between hardware and emulator targets; meaning students only need to familiarize themselves with the regular GDB user interface independently of the underlying target.

Complementary to GDB-Z80, a monitor software (or GDB stub) resides on the T80-SoC's ROM and is in charge of handling all debugging commands issued by GDB-Z80. Debugger to target communication is carried over a TCP to JTAG UART tunnel via custom software (JTAG Connector) making use of Altera's JTAG-over-USB facilities.

Fig. 2 shows GDB-Z80 in its usual setup to debug both emulator and hardware targets.

The advantages of using development environment around the GNU toolchain is twofold. Firstly, it helps students to get acquainted with widely used tools, available for several different architectures and platforms. Secondly, it provides a completely open development environment, which means students can benefit from taking a look under the hood of all tools.

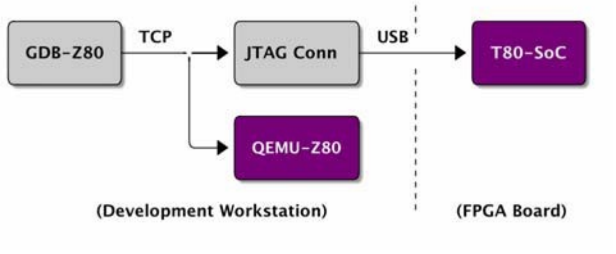

Figure 2. Alternate debugger configurations.

IV. CONCLUSIONS

Although this is still a work in progress, some preliminary conclusions can be drawn.

Only one edition of the new course has been held on 2011, being moved at the same time from semester 6 to semester 5 in the curriculum. The survey of opinion conducted by the school's Teaching Unit among students shows a score of 4.1 (in a 0 to 5 scale) for the global evaluation of the course, the same as in previous editions. The fact that the score was maintained even when the 2011 edition was a transitional implementation is promising.

With respect to the “lab at home” methodology, most of the beneficial results enumerated in [3] are starting to show, and a confirmation of these effects is expected in the following editions.

One of the most visible of these effects is student motivation. Also, the fact that great part of the work is done by the students outside the classroom optimizes teaching time, since the teachers dedicate their time mainly for answering questions, discussing and evaluating the results obtained by the students.

Furthermore, this method positively impacts on the infrastructure requirements because just one computer is needed to meet the needs of a large number of students and there is no need of big labs, relaxing the schedule constraints both of professors and students.

It is too early to assess the benefits of the introduction of hardware design activities to the lab. Beneficial results of this change should arrive following the 2012 edition of the course.

Regarding the hardware and software development tools, a blend of vendor, third party and in house developed open-source tools has been obtained that covers the whole development flow. It is remarkable that the debugging environment developed allows to switch seamlessly between real hardware and emulated hardware. The availability of an emulator for the processor used in the course gives the students still more flexibility to work on the road without the cost of the learning curve of a new tool.

Besides the immediate goal of introducing hardware design tasks into the lab assignments, we expect to add to the following editions of the course ready to run demonstrations of selected course topics and guided examples that can be exercised by the students at home.

References

[1] Syllabus of the course "Introducción a los microprocesadores," Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2011, (Available at http://iie.fing.edu.uy/cursos/course/view.php?name=imp).

[2] J.P. Oliver et. al. "Laboratorios en casa: una nueva alternativa para cursos masivos de diseño lógico digital," TAEE, Madrid, España, 2006.

[3] Juan Pablo Oliver, Fiorella Haim, "Lab at Home: Hardware Kits for a Digital Design Lab," IEEE Transactions on Education, Volume 52, Number 1, page 46--51 - feb 2009

[4] Lynne A. Slivovsky, Albert A. Liddicoat, "Work In Progress: Future Pedagogical Trends in the Microprocessor Course - The Soft Core Processor," 36th ASEE/IEEE Frontiers in Education Conference, 2006

[5] Lynne Slivovsky, Albert Liddicoat, "AC 2007-2341: Transforming the microprocessor class: expanding learning objectives with soft core processors," ASEE Annual Conference 2007

[6] Sin Ming Loo, C. Arlen Planting, "Use of Discrete and Soft Processors in Introductory Microprocessors and Embedded Systems Curriculum," ACM SIGBED Review, Volume 6 Issue 1, January 2009, ACM New York, NY, USA

[7] C. Camargo, "SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales", XVII Workshop de Iberchip, Bogotá, Colombia, February 2011.

[8] C. Camargo, "Transferencia tecnológica y de conocimientos en el diseño de sistemas embebidos," PhD Thesis, Universidad Nacional de Colombia, Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica, Bogotá, Colombia, 2011.

[9] F. G. Martin, "Integrating hardware experiences into a computer architecture core course," J. of Computing Sci. in Colleges, vol. 21, pp. 39-52, June, 2006.

[10] N. Manjikian and S. Simmons, "Evolution and enhancements of a microprocessor systems course," IEEE Trans. Edu., vol. 42, pp. 19 pp., 1999.

[11] W. Durfee, P. Li, and D. Waletzko, "Take-home lab kits for system dynamics and controls courses," in Proc. 2004 Amer. Control Conf., 2004, pp 1319.

[12] M. D. Takach and A. T. Moser, "Improving an introductory course on digital logic," in Frontiers in Education Conference, 1995. Proceedings., 1995, 1995, pp 4b6.1.

[13] D. Cyganski, D. Nicoletti, and J. A. Orr, "A new introductory electrical engineering curriculum for the first-year student," IEEE Trans. Edu., vol. 37, pp. 171-177, May, 1994.

[14] D. Millard and M. Chouikha, "Work in Progress: Hands-on Exploration of the "Big Ideas" in Electric Circuits," in Frontiers in Education Conference, 36th Annual, 2006, pp 3.

[15] L. A. DaSilva, S. F. Midkiff, and I.-R. Chen, "A hands-on course on wireless and mobile systems design," in Proc. of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops., Orlando, FL, 2004, pp 241-246.

[16] L. Gomes, "Programmable logic devices supporting embedded system design curriculum," in Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE, Raleigh, NC, USA, 2005, pp 6 pp.

[17] J. Cerdá, M. A. Martínez, M. Á. Larrea, R. Gadea, and R. J. Colom, "An active methodology for teaching electronic systems design," IEEE Trans. Edu., pp. 355-359, 2006.