

A Lab Project on the Design and Implementation of Programmable and Configurable Embedded Systems

Leonel Sousa, *Senior Member, IEEE*, Samuel Antão, *Member, IEEE*, and José Germano, *Member, IEEE*

Abstract—Several courses on embedded systems have been planned and taught to students in Electrical Engineering and Computer Science Master's programs. The tutorials play an important role in these courses, not only to motivate the students, but also to expose them to the difficulties and challenges of designing real embedded systems. This paper presents a complete lab project used over the last 15 years in a semester-long course on embedded systems that, for historical reasons, was titled *Computer Electronics*. This project involves all the topics studied in this course, namely programmable, configurable and dedicated processors, memory technology and organization, peripherals, and the design and implementation of complete embedded systems. Targeting image processing as a case study, in this lab project students use field programmable gate arrays (FPGAs) to design real-time embedded systems based on soft-cores, hardware accelerators, and specific input and output peripherals. Analyzing the results over the last years, it can be concluded that the inclusion of this transversal lab project boosts the interest in this Master's course and encourages the students to learn the topics required to design real embedded systems.

Index Terms—Embedded systems, Field programmable gate arrays (FPGAs), Master's course, memory, peripherals, programmable and configurable processors.

I. INTRODUCTION

EMBEDDED systems have emerged as an essential, although rather invisible, part of almost all consumer and industrial devices. The exponential growth of the market for those systems has resulted in the new subject of embedded computing and systems. Applications include high-end car models, which can contain more than 50 embedded microprocessors; aircraft, each of which typically contains more than 1000 of those processors; and general controlling devices. Moreover, the market for embedded processors has been growing every year, surpassing that for general purpose processors; in 2007, about 100 million processors were sold targeting computer central processing units (CPUs), while more than 3 billion embedded processors were sold in the same year [1]. Moreover, it is foreseen that by 2020, over 40 000 million devices (five to

ten embedded devices per person on earth) will be sold worldwide. Embedded systems have become an increasingly important subject, which is the focus of several specially designed courses [2] in Electrical Engineering and Computer Science Master's programs [3], and even specific Embedded Systems Master's programs. These latter programs often result from the international collaboration between various European universities necessary to cover all the principles, methods, and technological subjects required for designing and implementing embedded systems [4], [5].

The project lab described in this paper is part of a *Computer Electronics* course taken by students in the Computer Engineering and Electronics tracks of a Master's in Electrical and Computer Engineering (MECE). Prerequisites for this course are a knowledge of microprocessor architecture, assembly language, digital design, and C programming language, as well as having taken a basic course on circuits/electronics. *Computer Electronics* covers the main topics related to embedded systems, namely the following:

- 1) general-purpose, specialized, and dedicated processors;
- 2) instruction set architectures (ISAs), microarchitectures supported on the control unit and data path;
- 3) hardware description languages [Very-High-Speed Integrated Circuits], circuit deployment technologies such as field programmable gate arrays (FPGAs);
- 4) memories, basic data storage cells, static and dynamic memory circuits, nonvolatile memory circuits, caches, main and secondary memory;
- 5) buses, signal propagation and termination circuits, buses with multiple masters;
- 6) peripherals and I/O interfaces, with direct memory access (DMA), interrupts, serial and parallel communication protocols (synchronous and asynchronous);
- 7) case studies of complete embedded systems.

The course materials include a textbook on embedded systems [6], [7], the lecture slides, and a set of practical exercises specifically prepared for this course.

A comprehensive lab project on the design and implementation of embedded systems is presented here. Practical classes play an important role in courses on embedded systems, not only to motivate students, but also to expose them to the difficulties and challenges of designing real embedded systems. This paper is organized as follows. Section II presents the main objectives of the Computer Electronics course. Section III introduces the several phases for designing an embedded system, in this case for a video application. All the details of the lab project are discussed in Section IV. The assessment of the course and a discussion of the impact of the lab project on the students, and its con-

Manuscript received July 17, 2012; accepted September 18, 2012. This work was supported by national funds through FCT-Fundação para a Ciência e Tecnologia under Project PEST-OE/EEI/LA0021/2011.

The authors are with the Department of Electrical and Computer Engineering, Instituto Superior Técnico, Technical University of Lisbon, 1049-001 Lisbon, Portugal, and also with the Instituto de Engenharia e de Sistemas de Computadores (INESC-ID), 1000-029 Lisbon, Portugal (e-mail: leonel.sousa@inesc-id.pt; samuel.antao@inesc-id.pt; jose.germano@inesc-id.pt).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TE.2012.2222411

tribution to the success of the course, are discussed in Sections V and VI. Section VII draws the main conclusions.

II. MAIN OBJECTIVES

Embedded systems repeatedly execute a particular application with minimum energy consumption in a given time and at the lowest cost. The main objectives of the project presented here are to motivate students' interest in topics covered in the lectures and to expose them to the difficulties faced in the design of real embedded systems. The proposed lab project is all-inclusive in the sense that it does the following:

- 1) makes use of programmable processors, whose main features can be specialized for embedded systems;
- 2) uses high-level programming languages that do not depend on the processors to produce reusable code, such as C language, and assembly code to implement critical sections of the applications;
- 3) accelerates the processing and reduces the cost of particular parts of the applications by designing configurable hardware coprocessors;
- 4) interfaces with the real world by developing peripherals that connect the embedded system with sensors and actuators.

The project goes far beyond the use of traditional simple processors to implement embedded systems, such as the PIC [8], which typically have a large set of peripherals but allow neither their extension nor customization, for example, through the design of hardware coprocessors.

In terms of technology, the project exposes students to the current technologies used for prototyping digital systems, such as FPGAs, and also the design tools and development frameworks that they will meet in the industry. The equipment, material, and software tools required to carry out the project must not be too costly both to make it affordable to set up a laboratory with several working places, and also to make students sensitive to the problem of non-recurring engineering (NRE) costs, a topic covered in the course.

Given these goals, the project targets video applications, in particular the design of real-time video embedded systems. Real-time video processing is an interesting case study for designing embedded systems because of its demanding computational requirements (a minimum frame rate of 25 Hz is imposed in Europe) and the challenge it poses in terms of memory requirements, depending on the number of picture elements (pixels) in a frame. Furthermore, given that students need to design the peripherals for video acquisition and display, they can visualize the output of the system they are developing, which can be an extra motivational factor. Moreover, it provides the opportunity for students to choose and experience various approaches to designing embedded processors and systems, ranging from programmable to configurable and dedicated solutions.

III. ON THE DESIGN OF EMBEDDED SYSTEMS FOR VIDEO APPLICATIONS

In a first stage, students program the video applications by using the common programming frameworks and tools available in general computing systems. They program and test several processing algorithms as a preliminary step to designing

embedded systems for this type of application. By taking advantage of the high-level powerful tools available on desktop systems, such as the numerical computing environment MATLAB [9], students follow the usual project flow of embedded systems and, at the same time, are introduced to the typical techniques and algorithms used for image processing. Video processing encompasses 2-D spatial processing in a single frame [10], and also processing along the additional axis of time to set up a sequence of frames. The proposed lab project includes what are probably the most typical operations to be accomplished while processing images, discussed in the following.

In linear image processing, the system's impulse response, F with $k \times k$ coefficients, is convolved with the input data (I) in order to obtain the processed output data (O)

$$O[x, y] = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} I[x+i, y+j] \times F[i, j]. \quad (1)$$

The computational complexity for the convolution is $\mathcal{O}(N^2 \times k^2)$ for an $N \times N$ -pixels wide image.

A simpler but also typical class of operations in image processing is the computation of the histogram and its subsequent manipulation in order to improve, or adjust for a given purpose, the picture brightness and contrast. The complete histogram of an image is represented by a vector with the number of pixels in each one of the L brightness levels, $H[0 : L-1]$

$$H[I[i, j]] += 1; \quad \text{for } 0 \leq i, j < N, \text{ initial state } H[k] = 0. \quad (2)$$

Histogram equalization and image binarization are two examples of spatial operators based on the histogram of the image. Image binarization separates the background from the objects, for example by defining a threshold value th from a bimodal histogram and then applying it to the image

$$I[i, j] > th ? I[i, j] = 1 : I[i, j] = 0. \quad (3)$$

Histogram equalization redistributes the brightness values for making the histogram homogeneous, with the typical goal of improving the contrast. This goal requires the distribution of the most frequent intensity values by transforming the pixel intensities k of I with the function

$$T(k) = \left\lfloor \frac{(L-1) \sum_{k=0}^{L-1} H[k]}{N^2} \right\rfloor. \quad (4)$$

After this programming stage, in the laboratory students experience the difficulties of designing and implementing embedded processors and systems.

IV. LABORATORY PROJECT AND SESSIONS

The project runs for three and half months, approximately an academic semester, and has two main phases¹: 1) analysis of the MicroBlaze (MB) soft-core characteristics [11] and learning the development and implementation flows for embedded systems

¹The support material for these phases, including the tutorials, are publicly available at <http://sips.inesc-id.pt/~sfan/computer-electronics/>.

with FPGAs; and 2) analysis of the video processing algorithms and design of the specified embedded system.

All the hardware designed in the project is specified in VHDL and implemented in an FPGA. Although students are supposed to have background in digital systems design and VHDL, a tutorial on VHDL is provided in the lab for students not familiar with the use of VHDL for digital hardware systems specification.

The project is developed autonomously by groups of two or three students, in a laboratory that is always open during working days, but that is also used to teach other Master's courses. A teaching assistant is available once a week for 90 min to answer questions and solve problems that arise during the project.

A. Equipment and Material

A camera configurable through the serial camera control bus (SCCB) interface, the OV9650 color complementary metal–oxide–semiconductor (CMOS) Super Extended Graphics Array (SXGA) (1280 × 1024) [12], or the more recent and cheaper OVM7690 color Video Graphics Array (VGA) (640 × 480) [13] can be used as a data acquisition device. These are inexpensive cameras with direct digital output (8 bits for the RGB color model raw data) usually used in cellular phones. A description in VHDL of a basic hardware module for configuring the video format and controlling the camera operation [14] is provided to the students. This hardware module is used by students to build a peripheral for capturing video.

The project employs Xilinx FPGAs in the scope of the Xilinx University Program (XUP) and the EUROPRACTICE initiative. Although an *Educational Kit* based on a low-cost Xilinx FPGA [15] was used originally, several kits with FPGA with different capacities and characteristics are currently available at low cost. Currently, the Digilent S3 starterkit board [16] with the Xilinx Spartan 3 FPGA (part XC3S1000-4) [17] is being used, but any other FPGA and board with similar resources, such as the more recent [18], can also be used.

The video is displayed on a standard VGA computer display, using a simple digital-to-analog converter (DAC) to obtain the results in analog format.

The system design is done with the Xilinx ISE Design Suite (ISE) and Embedded Development Kit (EDK) [19]. EDK enables the design of a complete embedded processor system to be implemented in a Xilinx FPGA device. EDK and ISE can be obtained under the XUP and EUROPRACTICE initiative and can be used free for academic purposes.

Summarizing, each set up in the lab should have the following:

- a desktop computer that complies with the minimum requirements to run the ISE and EDK tools ver. 10.1.03;
- an FPGA board, the camera, and the adapter with a simple resistive array DAC to provide the VGA output signal;
- an affordable digital oscilloscope/logic analyzer such as Bitscope [20];
- an additional VGA display in case the output video needs to be displayed at the same time the students are designing the system.

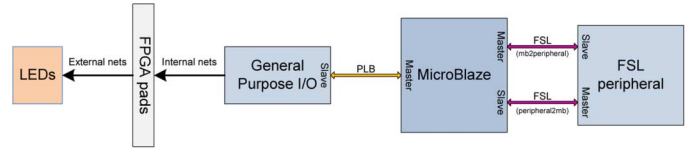


Fig. 1. MicroBlaze-based FSL example overview.

The equipment cost for such a setup, which is used by a group of two or three students, is around \$1000 (800 €).

Students are given guidelines as to where to find all the information required to complete stage 1 of the project. However, since this is scattered across many locations and technical manuals, a specific tutorial was prepared that gathers all this required information along with a detailed guide with the specifications of the embedded system to be designed and the main phases of the lab project.

B. Tutorial for Implementing MB-Based Systems

The tutorial for implementing MB-based systems [21] was prepared assuming that students have analyzed the organization of the FPGA devices, have closely read the MB architecture and instruction set, and are familiar with the Xilinx EDK environment.

With this tutorial, and by employing the Xilinx ISE and EDK tools and using the Digilent S3 Starter Kit board, students should know how to do the following:

- implement the MB on the FPGA with the intended configuration;
- describe an algorithm to be implemented in software and hardware on an FPGA system;
- design and attach peripherals to the MB and to interface the FPGA with external devices.

In this tutorial, students create a project in the EDK and, step by step, configure the MB soft core. It is a good opportunity to apply what they have learnt in the course lectures about program and data caches, and arithmetic and logic units (ALUs) for integer and floating-point, and also to experience that configurations also depend on the resources available in the particular FPGA device and should therefore be carefully selected. The tutorial proceeds by guiding the student in configuring and using a peripheral connected to the processor local bus (PLB)—in this case, to control the eight LEDs available in the starter kit board through the ports connected to the FPGA's pads, as shown in Fig. 1. Moreover, the tutorial also shows how to create peripherals, not through the PLB, but by generating a fast simplex link (FSL) bus. The FSL bus in Fig. 1 works as a first-in–first-out (FIFO) queue and assumes only one slave and one master: The slave is the component that receives data, and the master is the one that sends data. On the other hand, the PLB is more complex and comprises a single master and several slave peripherals, thus it is slower due to the overhead associated with the control of the bus.

Students learn how to use the FSL peripheral by programming the MB in C language, in this case to carry out a simple application to blink the LEDs repeatedly in a certain sequence. The tutorial also suggests performing the same application with

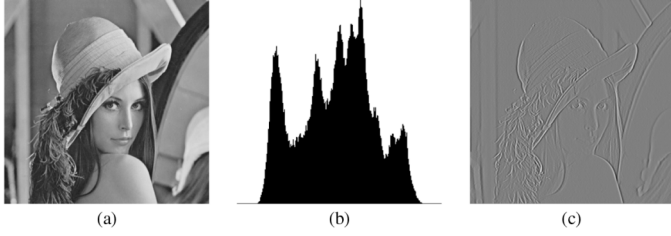


Fig. 2. Image processing output examples. (a) Original image. (b) Histogram. (c) Embossing.

hardware by developing the new peripheral by describing the application with VHDL rather than coding it in software.

C. Designing the Embedded System

Following the tutorial for this part of the project [22], students are encouraged to program the application on a general purpose computer using C language or MATLAB as a first step in the design of embedded systems. The results of this exercise are presented in Fig. 2 for an image embossing and histogram calculation over a 256-gray-levels image with 512×512 pixels.

In a qualitative way, by embossing an image, light/dark boundaries are replaced by highlights and shadows, and low contrast areas are set to a gray background. Usually, embossing is achieved by linear filtering, by using, for example, the following convolution kernel F with 3×3 coefficients in (1), producing the results illustrated in Fig. 2(c)

$$F(i, j) = \begin{pmatrix} -2 & -2 & 0 \\ -2 & 12 & 2 \\ 0 & 2 & 2 \end{pmatrix}. \quad (5)$$

The computation of the image histogram and its subsequent application in operations such as binarization or histogram equalization were discussed in Section III [see (2)–(4)]. The complexity of mapping the image representation from the spatial domain to the histogram is $\mathcal{O}(N^2)$, while embossing is a more complex but local operation $\mathcal{O}(N^2 \times k^2)$, for a $k \times k$ convolution kernel. Therefore, implementation of the embossing algorithm requires more attention in order to achieve real-time processing with low energy consumption.

The main, and most important, phase of the project then starts. Students have to set up a new project in the EDK tools, set up the MB, create the peripherals for video acquisition and display, and connect them with the MB. The frame size is defined and the input peripheral configured according to the amount of memory configured with the EDK, in line with the resources available in the target FPGA and the characteristics of the processing algorithms. For example, if students only use the internal memory of the FPGA [block random access memories (RAMs) (BRAMs)] the amount of available memory is only 54 kB (432 kb). Approximately half of the total memory is used to store the program, and the remaining amount is given to temporary variables, including the two images (the originally acquired and the processed) that have to be stored in 16 kB each. Therefore, low-resolution grayscale format has to be adopted for the images, such as 64×128 pixels with 1 byte per pixel (256 grey levels). Students can use external memory, such

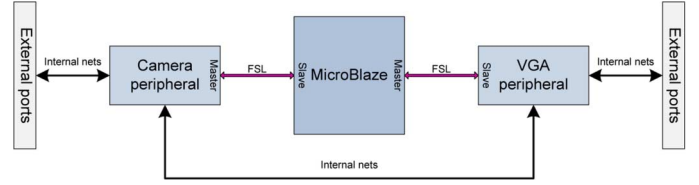


Fig. 3. MB-based EDK project for image processing.

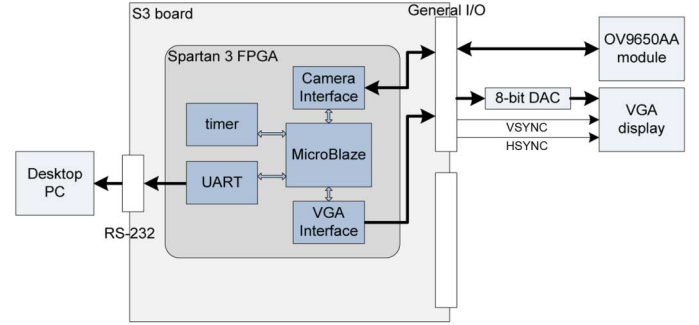


Fig. 4. MB-based image processing system configuration.

as the static RAM (SRAM) [16] or the synchronous dynamic RAM (SDRAM) [18]. They are slower than BRAM, but allow students to experience the design of memory controllers.

An overview of the MB-based EDK project that students have to set up is provided in Fig. 3, and Fig. 4 depicts the configuration of the complete embedded system. Basic versions of the *Camera* and *VGA* peripherals in Fig. 3 are provided to the students, together with the timing information for the control and status signals as well as the commands to configure the video camera. At this point, students have to change and adapt the VHDL description of the hardware associated with those peripherals. The DAC in Fig. 4 is implemented by a simple R/2R resistive ladder for a resolution of 8 bits and with an output impedance of approximately 75Ω , for connecting directly to a VGA monitor where the original and processed images are displayed [22].

As depicted in Fig. 4, two additional peripherals are implemented to support the debugging and configuration of the system. These are available in the library of standard peripherals for the MB: 1) a universal asynchronous receiver/transmitter (UART); and 2) a timer. The UART is useful for sending data from the FPGA to a computer through the RS-232 port. The timer peripheral registers the processing time by counting the number of clock cycles required to execute the algorithms on the embedded system.

A common clock signal of 50 MHz is used in the design, which is adapted to the MB core, the external memories, and the peripherals. After designing and setting up the system, the embossing and histogram algorithms are implemented by programming the MB in C. This implementation has to respect the restrictions imposed by the embedded system, such as the exiguity of memory, which allows neither the storing of a full intermediate frame, nor floating point arithmetic since this type of arithmetic unit was not introduced in the MB's setup phase. Moreover, operations such as division, not directly implemented by the ALU, should also be avoided.

```

// -- build histogram by software --
max = 0; hist[NUM_COL]={0};
for (i=0; i<num_lin; i++)
    for (j=0; j<num_col; j++)
        hist[image[i][j] >> 1]++;
// -- OR generate histogram in peripheral --
for (n=0; n<NUM_COL; n++)
    read_from_fsl(hist[n], cameraout_slot_id);
// -- print the histogram as an image --
for (n = 0; n < NUM_COL; n++)
    if (hist[n] > max) max = hist[n];
for (n=0; n<HIST_COL; n++)
    hist[n] = hist[n] * num_lin / max;
for (n = 0; n < HIST_COL; n++) {
    h = num_lin - hist[n];
    for (j=0; j<h; j++) pixel(j, i)=COLOR_WHITE;
    for (j=h; j<num_lin; j++) pixel(j,i)=COLOR_BLACK; }

```

Fig. 5. Program to obtain the histogram and prepare it for display.

```

(...)
--input of hist is an increment from the output
inCam <= "0000...000" & (outCam(12 downto 0) + '1');

--process to handle the synchronous signals
HIST_PROC: process (Clk)
begin
    if rising_edge(Clk) then
        (...)
        case HIST_state is
            when HIST_RESET_ST => --clear the hist.
                (...)
            when HIST_WRITE_ST => --write the hist.
                (...)
                if HIST_Write = '1' then --a pixel ready
                    addrCam <= '0' & DATAOUT; --the addr. is the pixel
                    -- the img. is totally processed
                    if ADDRESS = "111111111111" then
                        HIST_complete <= '1';
                        HIST_state <= HIST_READ_ST; --ready to read
                    else
                        HIST_complete <= '0'; --wait for more pixels
                        HIST_state <= HIST_WRITE_ST;
                    end if;
                else --wait until a pixel is available
                    addrCam <= addrCam;
                    HIST_complete <= '0';
                    HIST_state <= HIST_WRITE_ST;
                end if;
            when HIST_READ_ST => --read the hist.
                (...)
            end case;
        (...)
    end if;
end process;

```

Fig. 6. VHDL describing the histogram creation-related code from the peripheral that computes the histogram by hardware.

A segment of the program for computing and displaying the histogram for 8-bit grayscale 64×128 images is presented in Fig. 5. Given that 8-bit images define 256 brightness levels and the image has only 128 columns, the pixels of two consecutive levels (odd and even) are merged into a single vector element (column). The histogram can be built either by software, using the first two program loops at the top of Fig. 5, or directly by hardware implemented in the *Camera peripheral* (see Fig. 3). In this latter case, the *read_from_fsl* function is used to read the values of the histogram from the FSL connected to the *Camera peripheral* where they are calculated with the hardware specified in VHDL. Fig. 6 summarizes the VHDL specification to create the histogram. Following the computation of the histogram in the peripheral, the software gets the maximum value

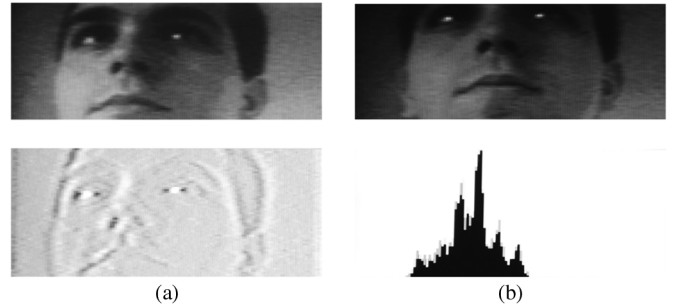


Fig. 7. (top) Sampled and (bottom) processed images, obtained and displayed with the developed embedded system. (a) Image embossing. (b) Histogram.

```

int kconvol[3][3]={{-1,-1,0}, {-1,6,-1}, {0,-1,-1}};
for (i=1; i<num_lin-1; i++)
    for (j=1; j<num_col-1; j++) {
        // -- convolution loop in C --
        p=image[i-1][j-1]*kconvol[0][0];
        p+=image[i-1][j]*kconvol[0][1];
        (...)
        p+=image[i+1][j+1]*kconvol[2][2];
        image[i-1][j-1] =(p << 1)}
// -- OR in assembly; r8- pointer to acquired image --
    lbui r3,r8,0
    lbui r4,r8,1
    lbui r5,r8,num_col
    lbui r6,r8,num_col+1
    rsubk r3,r3,r0
    rsubk r3,r4,r3
    rsubk r3,r5,r3
    addk r7,r6,r6
    addk r3,r7,r6
    (...)
    bslli r3,r3,1
    sbi r3,r8,0

```

Fig. 8. C and Assembly programs to perform image embossing.

of the histogram; this value is applied to normalize it in order to fit and display the maximum value in the available number of lines of the resulting image. To display the histogram, the height of the black bar in each column corresponds to the maximum number of pixels currently set to a given level, in this particular case a pair of levels, as depicted in Fig. 7(b). Under the conditions and for the aforementioned operations, the time for the complete processing is halved when the histogram is computed in the peripheral by hardware (typically 1.5 against 3 ms obtained with software). By developing the C and VHDL implementations of the histogram, students recognize the tradeoff between the development simplicity of a software control flow (lower performance) and the complexity of a dedicated hardware finite-state machine (FSM) (higher performance).

The same exercise was accomplished as with the image embossing, but in this case, instead of developing dedicated hardware on the peripheral, the algorithm is programmed in Assembly, as depicted in Fig. 8. The inner loops are unrolled, and the implementation can be optimized by taking advantage of the particular values of the coefficients. Moreover, the reuse of the data loaded into the registers is another optimization step that students can come up with. By taking an optimized version of C, the processing time is 16 ms, and in Assembly a further reduction of about 30% can be achieved.

V. ASSESSMENT, DIFFICULTIES AND LESSONS LEARNED

In addition to the final exam, student assessment is based on each group's demonstration of the operation and efficiency of the implemented embedded system, and on an individual discussion about the different aspects of the work and the options taken. The demonstration takes place in the last lab session of the semester, after which the students have one week to prepare a report of a maximum of 15 pages that should state and justify the options chosen, describe the design and the implementation issues, and present experimental results. These project reports provide the basis for the professor's discussions with the students that take place two weeks before the final exam. Their purpose is not only to assess the students' knowledge, but also for the students to realize their level of expertise in each topic so that they can organize their final studies more efficiently.

Over the years the course has been given, some difficulties have been identified. In spite of the difficulties of learning how to use the tools, the tutorial enabled students to employ the tools to set up and implement the platform in a smooth and autonomous way. The main difficulties were encountered in what should have been easier subjects that they were supposed to have studied and mastered in other, more elementary, courses. First, it is quite hard for the students to understand that implementing programmable embedded systems is harder than programming general-purpose systems. Students are surprised that they do not have full access to functions that are provided by specific libraries, such as printing and the dynamic allocation of memory during execution (e.g. *printf*, *malloc*). Most of them have the false perception that the functions provided by specific libraries and operating systems are provided by the language (e.g., C) itself. Moreover, students think that after programming a desktop computer, porting the code to a programmable embedded system is just a question of copying and pasting. During the completion of the lab, students have to identify the main memory sections, the code, the stack, and the heap segments, a constraint they had never experienced before, having always used sophisticated programming tools with graphical interfaces that hide all the hardware details from the programmers. Students typically overuse temporary variables, in this case 2-D arrays, and only know how to solve this problem by using allocation and deallocation functions.

In the first year after the implementation of this project in the laboratory, a lesson learned was that it is much more efficient to introduce students to practical complex aspects, such as learning how to use the Xilinx ISE and EDK tools, by providing them with carefully prepared tutorials rather than by giving traditional lectures in the practical classes. It was observed that students supplement the tutorial with information from the Xilinx and other Web sites, and when they need support from the professor, they are able to be specific about problems they are having. Moreover, over the time this course has been offered, some groups have developed in parallel to the class embedded systems for other applications, either because they are interested in the topic or they find it useful for other courses or their Master's theses.

VI. IMPACT AND RESULTS

The course is assessed semiannually. This assessment is based on the final grades and on the results of anonymous

student questionnaires on the course organization, the material available to study and to perform experimental work, and also about the availability of the professor to support students and their work. Moreover, since it is an elective course, the enrollment numbers also measure the real interest that the course arouses in the students.

Since the project was introduced as the basic lab work in the course, the average enrollment has been 25 students, making it one of the most popular among students of the MECE. Moreover, students have repeatedly emphasized in the questionnaires that although the lab is demanding and they have to work hard, it is the key to understanding and learning how to apply the concepts and methods taught in the lectures.

VII. CONCLUSION

This paper presented a complete lab project covering all the subjects studied in a Master's course devoted to teaching embedded systems. This lab project is transversal as it includes and integrates hardware and software in *lato sensu* for designing and implementing a complete embedded system on reconfigurable devices (FPGA). It requires knowledge of digital circuit design, memories, peripherals, buses, programmable processors and hardware accelerators, and programming in Assembly and in C. The results of including this lab project in the course suggest that not only does it make the course more attractive to students, but it also motivates them to study the topics necessary to support the development of real-time embedded systems.

REFERENCES

- [1] University of Colorado, Boulder, COL. McClure, "Embedded System Design—ECEN 4613/5613, Spring 2012," Flyer of the course, 2012 [Online]. Available: http://ecee.colorado.edu/~mcclure/S12_Embedded_System_Design_Flyer.pdf
- [2] W. Wolf and J. Madsen, "Embedded systems education for the future," *Proc. IEEE*, vol. 88, no. 1, pp. 23–30, Jan. 2000.
- [3] Columbia University, New York, NY, "Embedded System Design CSEE 4840," 2012 [Online]. Available: <http://www.cs.columbia.edu/~sedwards/classes/2012/4840/>
- [4] University of Kaiserslautern, Norwegian University of Science and Technology, University of Southampton, "European Master's program in embedded computing systems," 2012 [Online]. Available: <http://mundus.eit.uni-kl.de/>
- [5] TU Delft, TU Eindhoven, Universiteit Twente, The Netherlands, "3TU embedded systems," 2012 [Online]. Available: http://www.3tu.nl/en/education/embedded_systems/
- [6] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*, 1st ed. New York: Wiley, 2002.
- [7] W. Wolf, *High-Performance Embedded Computing: Architectures, Applications, and Methodologies*, 1st ed. San Mateo, CA: Morgan Kaufmann, 2007.
- [8] Microchip Technology, Inc., Chandler, AZ, "PIC microcontrollers," 2012 [Online]. Available: <http://www.microchip.com/>
- [9] MathWorks, Natick, MA, "MATLAB site," 2012 [Online]. Available: <http://www.mathworks.com/>
- [10] R. Gonzalez and R. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2008.
- [11] "Microblaze Reference Manual" ver. 10.1, Xilinx, Inc., San Jose, CA, 2008 [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf
- [12] OmniVision, Santa Clara, CA, "OV9650 color CMOS SXGA," Adv. Inf. Product Brief, 2005.
- [13] OmniVision, Santa Clara, CA, "OVM7690 640 × 480 CameraCube device," Adv. Inf. Product Brief, 2012.
- [14] L. Sousa and J. Germano, "Video platform implementation guide: Image capture," 2005 [Online]. Available: <http://sips.inesc-id.pt/videodevkit/>
- [15] L. Sousa and J. Germano, "Educational kit," 2005 [Online]. Available: <http://sips.inesc-id.pt/videodevkit/>

- [16] DIGILENT, Pullman, WA, "Digilent S3 starterkit board," 2012 [Online]. Available: <http://www.digilentinc.com/Products/Detail.cfm?Prod=S3BOARD>
- [17] Xilinx Inc., "Xilinx FPGA documentation," 2012 [Online]. Available: <http://www.xilinx.com/support/documentation/index.htm>
- [18] DIGILENT, Pullman, WA, "Nexys2 Spartan-3E FPGA board," 2012 [Online]. Available: <http://www.digilentinc.com/Products/Detail.cfm?Prod=NEXYS2>
- [19] Xilinx, Inc., San Jose, CA, "Xilinx ISE and EDK tools," 2012 [Online]. Available: <http://www.xilinx.com/support/download/index.htm>
- [20] Bitscope, St. Leonards, Australia, "BITSCOPE: PC oscilloscopes and analyzers," 2012 [Online]. Available: <http://www.bitscope.com/>
- [21] L. Sousa and S. A. Ao, "MicroBlaze softcore and Digilent S3 FPGA demonstration board: Tutorial," IST Computer Electronics: Coursework, Lisbon, Portugal, 2012.
- [22] L. Sousa and S. A. Ao, "Embedded system for image processing based on the MicroBlaze softcore implemented on FPGAs," IST Computer Electronics: Coursework, Lisbon, Portugal, 2012.

Leonel Sousa (M'01–SM'03) received the Ph.D. degree in electrical and computer engineering from the Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Lisbon, Portugal, in 1996.

He is currently a Full Professor with the Electrical and Computer Engineering Department, IST, and a Senior Researcher with INESC-ID, Lisbon, Portugal. He has contributed to more than 200 papers in journals and international conferences and is coauthor of the book *Bioelectronic Vision: Retina Models, Evaluation Metrics, and System Design*, *Series on Bioengineering and Biomedical Engineering* (World Scientific, 2009). His research interests include embedded systems, computer architectures, and parallel computing.

Prof. Sousa is currently a member of the HiPEAC and of the IFIP WG10.3 on concurrent systems. He is a Senior Member of the Association for Computing Machinery (ACM). He is an Associate Editor of the *EURASIP Journal on Embedded Systems*.

Samuel Antão (M'12) received the M.Sc. degree in electrical and computer engineering from the Instituto Superior Técnico (IST), Technical University of Lisbon, Lisbon, Portugal, and is currently pursuing the Ph.D. degree as a Researcher with the Signal Processing Group (SiPS), INESC-ID, Lisbon, Portugal, in the fields of cryptography, embedded systems, and digital electronic design.

His research interests include computer arithmetic, residue number systems, computer arithmetic, and computer architecture.

José Germano (M'11) received the Bachelor's, M.Sc., and Ph.D. degrees in electrical and computer engineering from the Instituto Superior Técnico (IST), Technical University of Lisbon, Lisbon, Portugal, in 2004, 2006, and 2011, respectively.

He is currently a Researcher with INESC-ID, Lisbon, Portugal. He has been working on data acquisition and processing platforms for DNA recognition based on magnetoresistive biochips. His research interests include bioelectronics/biomedical instrumentation implantable and wearable electronics and processor microarchitectures.