AN ADAPTABLE EDUCATIONAL PLATFORM FOR ENGINEERING AND IT LABORATORY BASED COURSES

Eduardo Bezerra[1], Marianne Pouchet[2], Ney Calazans[3], Fernando Moraes[4], Michael Gough[5]

Abstract ⾒ *This work describes an educational kit developed at the University of Sussex, UK. The kit is based on reconfigurable computing technology, targeting its use in different laboratory based courses, with minimal modifications of the hardware components, and small GUI development. The paper describes the kit modules, and its adaptation to a case study.*

Index Terms ⾒ *Engineering and IT teaching equipment, laboratory innovations, on site and distance learning, reconfigurable computing.*

INTRODUCTION

Putting into practice the concepts taught in technological courses is an essential complement to a good understanding of theory. The constant changes and advances in technology pose an ongoing challenge to universities and technical colleges to keep their laboratories updated. The managers of modern courses in the highly competitive market of teaching Computer Science, Electronics and Computer Engineering in general, have to find ways to provide students and tutors with appropriate tools, taking into consideration the usually modest available budget. A prototyping kit based on reconfigurable computing technology is one suggestion for the solution of this problem. This solution has been implemented at the University of Sussex, UK, and is used in a Digital Communications Systems course for undergraduate students, and in one module of the MSc. course at the School of Engineering and Information Technology. The implementation of the project has become feasible due to the falling costs of large Field Programmable Gate-Arrays (FPGAs) devices, which are the most widely used components in the design of reconfigurable computers.

The students use a Graphical User Interface (GUI), developed according to the course requirements, to observe and to control the execution of their experiments on the prototyping boards. It is important to highlight that, depending on the course, the use of reconfigurable computing technology is completely transparent to the students. For example, in the Data Communications course the goal is to teach communications' protocols and standards, and to show how to program and use a traditional USART. In this case the students do not even have to know about the existence of the FPGA on the prototyping board. In other situations, as in a 'VHDL for synthesis' course or in a hardware/software co-design course, additional tools such as commercial synthesis packages can be used by the students to generate configurations for the FPGA.

The main objective of this paper is to introduce some of the features of the teaching platform developed at Sussex. The paper is organized as follows. The next section introduces traditional laboratory-based courses, briefly describing a laboratory based course of a Brazilian university and a similar one taught in a British university. After, the system architecture is examined from the software and hardware points of view. The paper ends with the description of a case study, final conclusions and future directions.

TRADITIONAL LABORATORY BASED COURSES

The purpose of many electronics and computer science laboratory exercises is both to familiarise the student with the tools and equipment used in his/her field, and to demonstrate experimentally some aspect of learnt theory.

Traditionally, to achieve this, two different types of equipment are used:

- Standard Laboratory Equipment. These include computers used as a user interface, monitoring and control; oscilloscopes, multimeters; and logic analysers for measurement and monitoring of input/output signals.
- Application Specific System (AppSS). This usually consists of a piece of application software running on a standard personal computer (PC) as well as a custom-made external printed circuit board (PCB), which experimentally demonstrates the particular theory of the lesson. This board is open for student use during the experiment. An ASS may also include one internal PCB, which is hidden from student use and placed inside the PC to mediate between the running application software and the external PCB.

The student, following the laboratory script, uses the application software on the PC and generally monitors and measures the inputs and outputs from the PCB using the oscilloscope, probes and/or meters. Figure 1 shows the

---

[1] Eduardo Bezerra, PUC-RS (Brazil) and Sussex University (UK), Space Science Centre, EIT, Brighton, BN1 9QT, England, UK, eduardob@acm.org
[2] Marianne Pouchet, Sussex University, Space Science Centre, EIT, Brighton, BN1 9QT, England, UK, M.S.Pouchet@sussex.ac.uk
[3] Ney Calazans, PUC-RS, Informatics Faculty, Porto Alegre, RS, Brazil, calazans@inf.pucrs.br
[4] Fernando Moraes, PUC-RS, Informatics Faculty, Porto Alegre, RS, Brazil, moraes@inf.pucrs.br
[5] Michael Gough, Sussex University, Space Science Centre, EIT, Brighton, BN1 9QT, England, UK, M.P.Gough@sussex.ac.uk

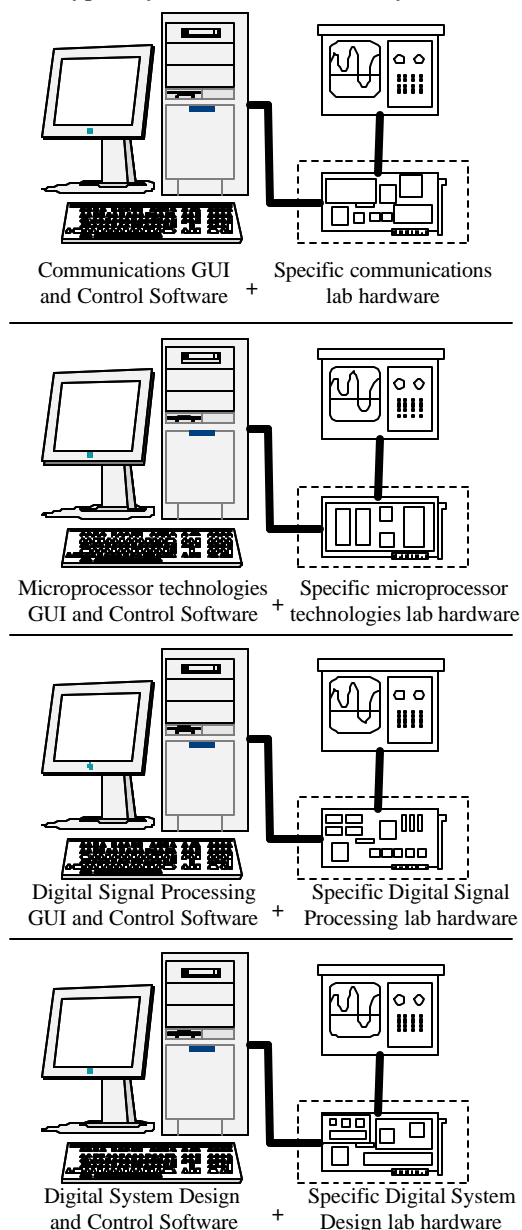resources typically used in four laboratory based courses.

Communications GUI and Control Software + Specific communications lab hardware

Microprocessor technologies GUI and Control Software + Specific microprocessor technologies lab hardware

Digital Signal Processing GUI and Control Software + Specific Digital Signal Processing lab hardware

Digital System Design and Control Software + Specific Digital System Design lab hardware

FIGURE 1
TRADITIONAL LABORATORY BASED COURSE.

In this traditional styled laboratory a different AppSS needs to be designed and built for each different laboratory. Also since the software and hardware are designed for specific purposes, upgrading such laboratories to meet the growing demands of advancing technology may prove difficult. Usually, an upgrade of this nature results in an entirely new AppSS being built with all its attendant costs. Extra time and effort is required as well, by lecturers, technicians and students to familiarise themselves with the new software interface and external PCB connections.

The sad result of these difficulties is that in many universities and colleges, the laboratory systems are not updated regularly leaving students disadvantaged in the experimental experience of and theory behind modern technological advances. In order to obtain a better understanding of the motivation for the development of the Multimedia Adaptable Educational platform (MAE) shown in Figure 2, laboratory based courses of a Brazilian and a British University are introduced next.
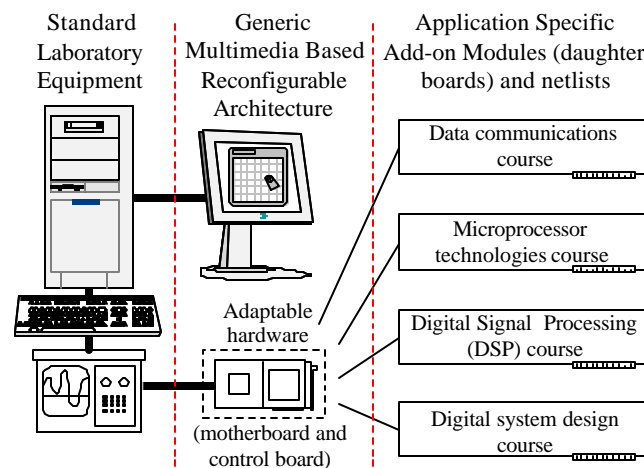

FIGURE 2
MULTIMEDIA BASED ADAPTABLE EDUCATIONAL PLATFORM (MAE).

In the Computer Organization course at the Informatics Faculty of PUC-RS, Brazil, both lectures and laboratory courses start with a traditional, schematic-based approach to processor core design [1]. Later, the whole design work is redone with modern HDL-based tools. One of the main reasons for this is to show that it is possible to work at high levels of abstraction and still obtain efficient designs, as a result of recent improvements in supporting tools. Another objective is to make it clear to students that HDLs are not programming languages, which is achieved by the mapping of schematic symbols and structures to the syntactic and semantic structures of the chosen HDL. This last reason makes instructors insist to students that they must 'think hardware' when using HDLs, even if the language they use allows them to view hardware as software. The Computer Organization Laboratory course is divided into three units, where the students start working with schematic capture and simulation tools, in order to implement basic combinational and sequential blocks such as adders, ALUs, counters and finite state machines. The students also implement a simple CPU using RTL schematic modules. Examples are available for the students, from simple ALUs up to small load-store processors. Both for their work with schematic capture and with VHDL synthesis, they have to implement their circuits on available prototyping boards (Xess/Xstend). The use of MAE will introduce more flexibility for the lecturer in defining the case studies, and it will provide the students with higher levels of observability and controllability when running their experiments.

In the British university, MAE is in use in two courses

to demonstrate to students different serial communication protocols. In a sequence of three laboratories, issues involved in the data-link layer of serial data transmission are demonstrated allowing students to choose design settings and observe results. Transmission and reception can be conducted either on the same kit or by using two separate systems – one as transmitter and the other as receiver. RS232, RS422 and Manchester Encoding are all used as physical layer protocols throughout the labs, and a GUI written in Java is used by the student to send simple control commands to the hardware and to observe results.

In a traditional laboratory-based environment, for the courses described in this section, three different kits from Figure 1 would be necessary: 'Digital System Design', 'Microprocessor Technology' and 'Communications lab hardware'. Considering the use of the proposed MAE platform in the same courses, as shown in Figure 2, only one motherboard would be necessary together with three simpler and less expensive daughter boards. An important feature and advantage of MAE is that a single and integrated GUI can now be employed in different courses.

MULTIMEDIA BASED ADAPTABLE EDUCATIONAL PLATFORM (MAE)

To achieve the same laboratory goals as a traditional laboratory, the MAE system consists of three types of equipment shown in Figure 2: 'Standard Laboratory Equipment', 'Generic Multimedia Based Reconfigurable Architecture', and 'Application Specific Add-On Modules'.

The **Standard Laboratory Equipment** consists of PCs, oscilloscopes, digital analysers, probes and multimeters. They are used as previously for User Interfaces, control, monitoring and measurement of input and output signals.

The **Generic Multimedia Based Reconfigurable Architecture** consists of:

- A simple, easy-to-use, *GUI* whose style is used in all laboratory exercises. Students use this interactive tool to easily and intuitively follow their laboratory exercise sheet either on-site (in the laboratory) or remotely (access via the Internet). The tool has been designed in such a way that in a future version it will also allow remote tutor access so that lecturers and tutors can monitor their students' progress without leaving their home or office.
- *Generic Control Software.* This is hidden from students and lecturers and is responsible for controlling the Generic Reconfigurable Hardware. This software is the same for all laboratory sessions.
- *Generic Reconfigurable Hardware.* This innovation comprises the very latest in embedded system technology. In essence it consists of a microcontroller and an FPGA, which together act as an intelligent force behind many, widely varied, types of laboratories. The FPGA has the property of being able to change its

functionality without changing any of its physical characteristics. The microcontroller changes the functionality or *reconfigures* the FPGA by loading a *netlist* [2] into it. These netlists are secure files, small enough to fit on a diskette and can be downloaded off the Lecturer's web-site. Using these netlists, the user can change from performing a Communications laboratory for example to a Microprocessor Technology laboratory without changing any of the equipment in-front of him, and not changing the GUI. This change can take place in less than a second and occurs behind the scenes so that the user only needs to be concerned with conducting the laboratory and not involve himself/herself into the workings of the hardware itself. Updating a laboratory can be as simple as downloading a file. This removes the effort and time involved in learning new software applications for each laboratory and also removes the substantial cost of buying a new AppSS. In its commercial version, this board will also be boxed-off and unavailable for students to tamper with, thus improving the integrity and robustness of the entire system.

The **Application Specific Add-On Modules** are used in conjunction with the Generic Multimedia Based Reconfigurable Architecture in order to custom-make laboratories to fit a desired functionality. Included here are:

- *Specific Control Software and Netlists* include self-extracting background software files that allow the laboratory interface to properly control the specific laboratory exercise. While the form of the GUI remains the same, changes in control buttons and interactive responses may vary from laboratory to laboratory and the GUI is updated on demand. These software files, along with the *netlists*, can be downloaded directly off the Internet and require no background knowledge of the laboratory system or theory to set up. Their control functions are transparent to the user who uses the familiar GUI set-up as per normal.
- *Add-On Hardware Modules* are optional devices that act as an open add-on board to the 'black-box' *Generic Reconfigurable Hardware.* They consist mainly of non-intelligent electronic devices and output and input pins. These are open to the student for measurement and monitoring using the traditional probes and oscilloscopes during an on-site laboratory session. As no intelligent devices exist on these boards, they are markedly cheaper than an AppSS PCB that would exist in a traditional laboratory. Also, unlike the case for their predecessors, student errors and tampering cannot result in complete system destruction but at worst the destruction of the expendable add-on module.

In laboratories where the control and monitoring are done via the PC only, these hardware modules are unnecessary and the *Generic Reconfigurable Hardware* is

used alone. The three stand-alone boards of the MAE platform, as shown in Figure 3, are: the **control board**; the **motherboard** (together comprising the Generic Reconfigurable Hardware); and the **daughter board** (Add-On Hardware Module). Students use the software module (GUI) to set up the mother and daughter boards, and to control the execution of their laboratory assignments. In the current version of the system, the host computer is connected to the prototyping platform via serial port. The use of a USB port is planned for future versions.

The **control board** consists, basically, of a microcontroller [3], flash memory and serial port interfaces. The netlist used by the microcontroller to reconfigure the FPGA is stored on the flash memory. The microcontroller runs an embedded operating system used to configure the FPGA and to manage the communication between the FPGA and the host computer. The software module running on the host computer is designed to use the services provided by the embedded operating system.

The **motherboard** consists of a Xilinx Virtex FPGA [2], an oscillator, and a power regulator module responsible for supplying the power for all boards. There is a Finite State Machine (FSM) running on the FPGA, responsible for the provision of the interface between the daughter board and the control board (and consequently, the user interface). Using an external oscillator chip in the motherboard to generate a clock signal to the circuit implemented within the FPGA, provides the flexibility of allowing different oscillators with different clock frequencies to be used for different applications.
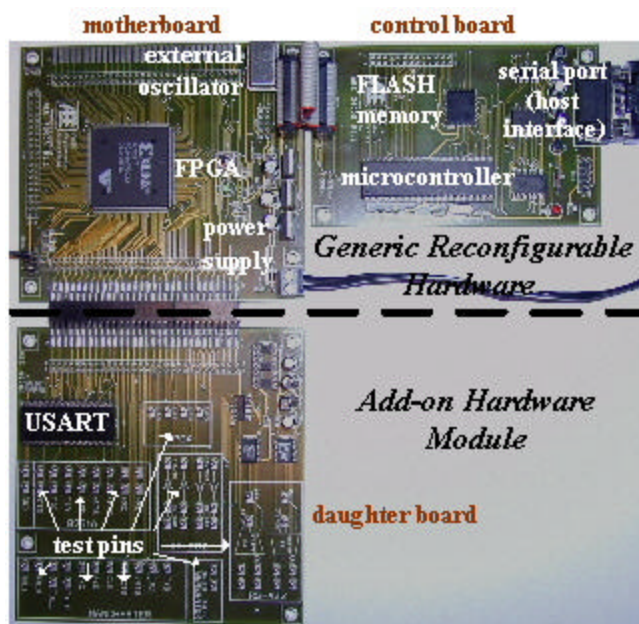

FIGURE 3
HARDWARE COMPONENTS: CONTROL BOARD, MOTHERBOARD AND DAUGHTER BOARD.

The **daughter board** is directly used by the students to run their experiments. Although some courses may require more than one daughter board, usually one daughter board per course/kit is sufficient. The user interface is designed in order to generate signals that can be observed from the daughter board output pins. The system as a whole is designed in modules in order to facilitate its adaptation to different courses. From the hardware point of view (i.e. chips and boards), only a new daughter board may be necessary for a new course.

It is important to add here that the FPGA and the microcontroller are placed in different boards, to improve system maintainability and upgrade features. For example, if only the FPGA needs upgrade, then just the motherboard should be changed. In a similar way, just the control board would need to be modified, if a USB or Ethernet connection is needed instead of the available serial port in the current MAE version.

CASE STUDY: TARGETING DATA COMMUNICATION COURSES AT SUSSEX

As an example of the functionality of the MAE platform, the application specific add-on module used for the Data Communication courses at the University of Sussex comprises a configuration netlist for the FPGA on the motherboard, and an 8251A USART, RS-232C/RS-422 converters and several test pins on the daughter board, as shown in Figure 3. The students can use protocol analysers and test equipment to verify the functionality of their designs. It is important to add here that there is no need for an 8251A chip on the daughter board. A USART core could have been implemented in the FPGA itself, but at the time the system was designed such cores were not fully compatible with the 8251A USART at an affordable cost. In the first version of the system, the FPGA had three main components, all of them implemented in VHDL: a manager kernel, a baud rate generator and a Manchester encoder/decoder. The kernel was responsible for the control of all the components, and also for programming and controlling the USART in the daughter board.

In the original design the idea was to have all components of the FPGA written in a single language (e.g. VHDL) and at the same level of abstraction. However, because of several scheduling problems, including board delivery delays, in its final version the FPGA configuration is a combination of modules written in VHDL and diagram schematic capture. In order to improve portability and maintainability, in future versions the implementations will use higher levels of abstraction, such as Java Forge [4].

As stated before, in a similar way to the daughter board and parts of the FPGA code, the GUI also has to be re-designed for each new course. The first screen of the GUI, written in Java, for the Data Communications course is

shown in Figure 4. The students use the Java program on the host computer to program the USART and to send/receive characters and complete frames either synchronously or asynchronously to the daughter board.

FIGURE 4
GUI OF THE SOFTWARE MODULE.

The GUI has been designed to be as simple as possible in order to keep the students' attention on the subject of the laboratory session. Using the GUI the students select few parameters per screen and press 'Next' to proceed to the next screen. After three or four screens, similar to the one shown in Figure 4, depending on the experiment the students are able to observe and compare the obtained results against the expected ones. The user interface has been written in Java [5] in order to run on different platforms. The user interface is connected to the control board by means of the Java Communications API *commapi* [5]. This Sun's API was designed to help in the development of platform-independent communications applications, but at the time this paper was written there were versions available only for MS-Windows and Solaris.

The student is thus presented with an easy-to-use GUI that guides him/her directly to the essence of the laboratory. In this case study, for example, the student is directed to choose synchronous or asynchronous transmission. They are prompted to use different configuration values for the USART, to send and receive data frames and to view the resulting waveforms on an oscilloscope. Thus the results of different configurations can be observed.

They are at greater liberty to observe the protocols involved in the differing types of communication between two external USARTS and to see the relative advantages and disadvantages of one protocol over another. Their minds are therefore kept focused on the objectives of the laboratory exercise without the need to understand the details of how data is propagated along the hardware, configure ports or set switches. This approach is particularly interesting as it hides from the students some issues that are to be introduced only in more advanced classes. For example, the student does not need to be exposed to all the procedures necessary to

program the USART since the first laboratory session. The training is done in a very progressive way with the student programming and learning just a couple of parameters for the serial communication each time. This apparently hand holding approach is necessary as in recent years Universities have been encouraged to take on students from a broader range of educational backgrounds.

In addition to this, if MAE is used in other University courses, the students will be presented with a software interface with which they are familiar from their other laboratories with only the laboratory specific content changing. This decreases the time needed for familiarization common to most traditional laboratory tools and allows the students to tackle almost immediately the task at hand.

Intense use by a group of 30 undergraduate students at the University of Sussex verified this - the platform proved to be robust and very convenient. The students managed to keep their attention focused on the data communication course tasks, completely ignoring any complexity associated to the reconfigurable component or microcontroller.

CONCLUSIONS AND FURTHER WORK

This work briefly describes an educational platform based on reconfigurable computing technology. The MAE platform has been designed for its reusability in different laboratory based courses. A case study was conducted on this system for two different Data Communications courses at the University of Sussex. The MAE platform proved both reliable and easy to use in this laboratory while maintaining its flexibility for re-use in other laboratories.

The system, however, has not been fully validated, as it has been used in only a couple of courses at the University of Sussex. In order to have qualitative and quantitative results regarding the system applicability, it will be necessary to use the system in different courses, and to ask in a more scientific way for feedback from the final users - tutors and students.

The most important hardware components of MAE (control board and motherboard) could be off-the-shelf boards, and so any other prototyping board based on FPGAs could have been used. The control board and motherboard alone can be seen as just another generic FPGA based prototyping platform. What makes the difference is the use of those two boards in conjunction with the daughter board and the software running on the host computer (GUI). Systems with similar features, and targeting similar applications (e.g. training and education) have not been found and, therefore, a comparative study could not be made. There are several FPGA based prototyping boards in use for teaching activities, but from the best of our knowledge all of them are used in HDL and FPGA courses.

To exemplify the flexibility of MAE, however, in the case of a microprocessor or computer organisation course, with minimal effort, a new user interface can be

implemented by reusing the existing objects. A microprocessor core could be used in the FPGA, and the daughter board would have, mainly, test pins. This will reduce the final cost of the kit, as embedding external devices into the FPGA is enabled by the increasing size of cheap reconfigurable devices, which can now pack hundreds of thousands of logic gates for a few Euros. The possibility of reusability of Intellectual Property (IP) cores and Java objects is an important point for the success of this project.

It could be argued that reusability and the familiarity of the GUI could be achieved at even lower cost by a purely software solution using an industrial modelled tool such as LabView or a custom made software system. This would still necessitate external hardware specific to the laboratory (similar to the daughter board described above) and more importantly would require the use of a port for all data communications. This would limit the available I/O and speed. It would also increase the complexity involved in real time data transfer, as limited buffering resources would be available. Almost unlimited I/O, external intelligence and large buffering options are made available through the use of an FPGA-microcontroller combination. This allows the experiment to run on an external clock with only the final communication to the PC limited by the speed of the ports.

The biggest advantage of this system, from a financial point of view, is that the daughter board, in most of the cases, is a very simple and, consequently, inexpensive component. In the case of changes or updates to course content, there will be no need for further investments in extra hardware. Only new add-on modules may be necessary, which means new daughter boards, with an upgrade in the FPGA configuration and in the user interface.

In order to illustrate this cost advantage when compared to traditional platforms, an average cost of €250 per specific hardware (Figure 1) was assumed as a result of a quick market research in Western Europe. A total of €1,000 would therefore be necessary to purchase the four specific traditional laboratory boards. Assuming the same cost for the adaptable hardware plus one add-on module (this was roughly the cost to produce one set of the boards shown in Figure 2), and an average cost of €50 per additional add-on module, the total necessary would be €250 + €50 + €50 + €50 = €400. The use of MAE to perform the same tasks as the boards in Figure 1 would therefore represent a total savings of €600 over four different courses. The development costs for the control software (GUI) in both approaches were not taken into account, as they do not differ much. A proper market research and costs estimation should be conducted in order for more realistic figures be obtained.

Considering the expected low cost of MAE, some students may actually decide to buy the kit. The configuration file and the user interface can be made available on the Lecturer's web site for students to download and use at home. The GUI written in Java language allows the execution of the laboratories to be platform-independent and even potentially web-based for optional distance use.

Flexibility and low cost are two of MAE's features that make it attractive to universities and research centres as PUC-RS and Sussex University. It is the intention of PUC-RS [1] to adopt MAE in courses as Computer Architecture II, where the variety of topics taught in classroom, would require several kits when considering the traditional laboratory approach mentioned in this paper.

A major concern pointed out with the proposed platform is that it may not realise the goal related to cost and flexibility across courses. This concern is correct and it is expected that in this still early stage of development of reconfigurable computing technology, MAE potential users (mainly universities) may continue to use the traditional approach for a little longer. This is explained as a result of the scarceness of staff familiar with FPGA technology. In order to implement new courses, even having the possibility of acquiring IP cores from vendors, there will be cases where the own laboratory staff will have to build the new application, sometimes from scratch. This is a not trivial task yet, but this has always been the case when a new technology is introduced in the market. With the great expansion of the FPGA culture in recent years, it is expected that better supporting tools will make the implementation of applications more attractive for both hardware and software developers. A good example of this type of tool is the already mentioned Java compiler [4]. Using a pure Java approach for both the GUI and hardware components of MAE (FPGA and microcontroller), universities will not have to depend on vendors to create new courses, as staff members with Java skills will be able to do the job. Finally, an important motivation for the use of MAE replacing traditional platforms is the recent introduction of FPGA related courses. If a University department has decided to acquire FPGA prototyping boards for these courses, why not using the same boards in several different courses?

ACKNOWLEDGEMENT

REFERENCES

[1] Calazans, N. L. V.; and Moraes, F. G. "Integrating the Teaching of Computer Organization and Architecture with Digital Hardware Design Early in Undergraduate Courses", IEEE Transactions on Education, vol. 44, no. 2, May 2001, pp. 109-119.

[2] Xilinx "The Programmable Logic Data Book", Xilinx, USA, 1999.

[3] Microchip "PIC 16F87X databook", Microchip, USA, 1999.

[4] Xilinx "Forge High Level Language Compiler" Xilinx. USA, 2002.

[5] Sun Microsystems, "Java 2 SDK, Standard Edition Documentation", Sun, 2001. http://java.sun.com/j2se/1.3/docs/index.html