From Gates to Multi-Processors Learning Systems Hands-On with FPGA4U in a Computer Science Programme

Claudio Favi, René Beuchat, Xavier Jimenez and Paolo Ienne
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
<firstname>.<lastname>@epfl.ch

ABSTRACT

We present in this paper our strategy to teach computer engineering and embedded systems in our Computer Science and Communication Systems programmes. We heavily rely on the FPGA4U board [1], specifically developed for this purpose. The board is introduced early in the first year of the Bachelor for the Digital Systems course. The students' knowledge is augmented over the years with computer architecture classes up to the Master level. Master classes on embedded systems use the platform for practical hands-on exercises in short projects with emphasis on either hardware, software, or hardware/software co-design. There are three critical elements in the value of our approach: firstly, the FPGA4U board is tailored to the students' needs (extreme portability, ease of use, cost); secondly, the practical advantage of using a single hardware resource across many courses makes more challenging projects possible in less time; finally, all the knowledge acquired is shared on a collaborative web platform. The project also fosters ideas by students who can thus develop practical skills in complex embedded systems beyond curricular requirements. Several other hardware projects (such as LCD, camera, and flash memory extensions) have been pursued to expand the platform.

Keywords

Education, Embedded Systems, FPGA, Logic Design, Processor Design.

1. INTRODUCTION

Engineering is the art of abstractions: it is the art of distilling extremely complex technological processes and concepts into readily useable intuitions. Without solid abstractions, all but the crudest pieces of technology would be unmanageable by humanity. Yet, abstract ideas not solidly internalized are unsuitable to build more complex concepts.

WESS '09, Oct 11-Oct 16, 2009, Grenoble, France

Teaching engineering is the art of helping students to develop solid abstractions. It may be hard or impossible to "teach" abstractions, to recount them as a function of other abstract concepts: for abstractions to be fully understood ("operational", one could say) the human brain seems to require to construct them by itself. Educators must come up with ways to accelerate this process, to provide the students' brains with the most effective ingredients for the abstractions to develop, to blossom: examples, experiments, and exercises are the most common tools. Anyone who has tried to teach any domain of engineering knows how difficult it is to find the right combination of repetition and undeterminacy, of prior knowledge and novelty, of explanation and surprise, that spark the process, much like a delicate stoichiometry of a complex chemical reaction.

Along with many educators, we believe that abstract knowledge, to be solid and usable, must be built bottom-up: students should be guided to develop by themselves the domain of knowledge they need to learn. Educators must do that as in a film which condenses the history of civilization in a couple of hours. It may be necessary to exaggerate or reduce the relative importance of various episodes, or to omit significant parts and even violate sequentiality to make the overall story most captivating and convincing. We strongly believe that this path is the most effective to achieve solid engineering knowledge, and we relate in this paper how we have configured the practical labs of the Computer Engineering track in the Computer Science and Communication Systems curricula of our institution.

2. PEDAGOGICAL GOALS

In 2005, we started to reconsider the teaching organization of hardware related classes and wanted to introduce more rich hands-on practical exercises. We stressed the fact that, whatever the hardware platform chosen, its progressive introduction to all the Computer Science and Communication Systems students was necessary. The rationale is that more material could be taught in advanced classes if the base concepts and functionality of the hardware is already known from earlier mandatory classes.

We also noted that fewer and fewer students were interested in hardware projects. A certain fear of electronics or a "it-just-works" syndrome seemed to appear across the student population. In order to demystify hardware design, our goal was to provide students with hardware as available as possible to them. Availability was our key word: for

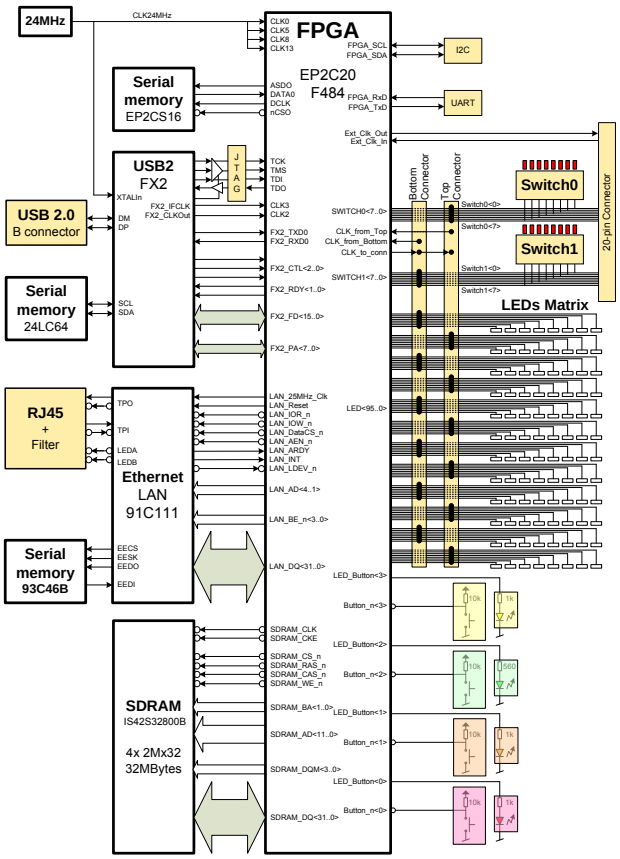

Figure 1: The FPGA4U board with an Altera Cyclone II FPGA as its core. The peripherals available range from simple inputs and outputs (LEDs, switches) to advanced protocol-based communication systems (USB, Ethernet).

instance, we believe that software skills are extensively developed in academia largely because of the pervasiveness of computing resources. Likewise, our goal was to give access to hardware development resources to each and every student. Instead of just providing easy access to resources, we actually put a hardware development board in the pocket of every first year Bachelor student.

To commoditize a hardware platform to the level of each student, the constraints were threefold: First, its size must be relatively small. We chose that the target size should approach credit-card dimensions. We believe this is an ideal size for a student to carry around. Second, the board should be easily powered and programmed and no external power supply or programming hardware must be needed beyond standard computing equipment (e.g., a common laptop). We adopted a USB-based solution. Last, the price for the student should be as low as possible. Although we had considered to give away the board for free, we thought that this would negatively impact the perceived value and the care the owners would take of the object. We decided for a low but nonnull price.

With these constrains in mind, we built a custom FPGA-based board. The board, described in the next section, measures 10 cm by 6 cm (3.9 in by 2.3 in), is USB powered, and, being heavily subsidized by the school and generously

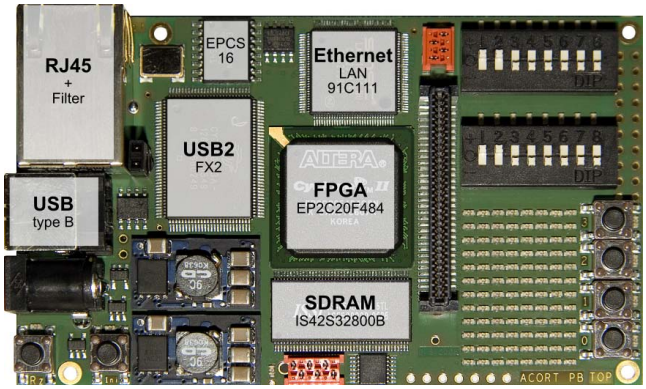

Figure 2: The FPGA4U board. Its small size (10 cm × 6 cm or 3.9 in × 2.3 in), its USB interface and power, and its ease of use adequately fits the student environment.

sponsored by some of the chip manufacturers, is provided to every first-year student for only 100 CHF (approximately 94 USD).

In the next section, we describe the FPGA4U board and all its features, which are also pictured in Figure 1. Section 4 presents the curriculum from Bachelor classes to Master classes; there, we illustrate our methodology with a storyboard which exemplifies our bottom-up introduction of new concepts. In Section 4.5, a description of some student projects is shown to give a sense of the extensibility of the platform. We present a comparison of our FPGA4U board with some existing boards in Section 5. Finally, in Section 6, we illustrate the impact of our approach with some statistics.

3. HARDWARE ARCHITECTURE

3.1 General Overview

Figure 1 presents the block schematic of the board. The main elements are: an FPGA (Field Programmable Gate Array), two 8-bit DIP-switches and LEDs, 4 push-buttons, 96 LEDs, and extension connectors. These were the starting point needed for the Digital Systems course. For more advanced courses, SDRAM memory, Ethernet, and USB 2.0 interface were selected.

FPGA. The heart of the system is an Altera Cyclone II FPGA [2]. With around 20,000 logic elements, 4 internal PLLs, twenty-six 18×18 multipliers, and 234 kbits of SRAM, full embedded systems can be achieved on a single component in a few minutes. With an FPGA, simple logic can be implemented thanks to the LUTs (Look-Up Tables) associated with D-type flip-flops. From simple logic, blocks of higher complexity can be built such as counters, finite state machines, etc. Specific interfaces for a logical internal bus (e.g., Altera Avalon [3]) can be easily developed and integrated in a full system.

While designing the board, one of the goals was to be able to directly use the manufacturer's tools for development, design upload, and debugging. The Quartus II environment integrates schematics, VHDL and Verilog entry, design synthesis, place and route, programming etc. In other words, it integrates the complete toolflow from design entry to firmware upload.

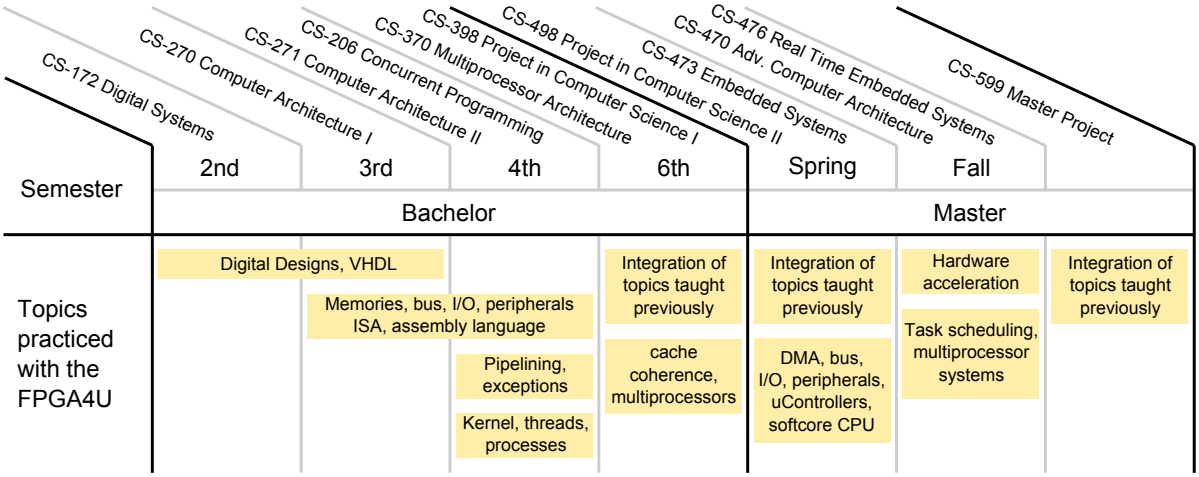

Figure 3: Topics practiced by students with an FPGA4U during a typical Computer Science or Communication Systems curriculum oriented toward Computer Engineering.

USB. We emulate USB Blaster functionality (firmware upload and debugging) with the help of a Cypress FX2 chip [4]. High speed data transfers with a PC can be achieved through the use of dedicated endpoints (one for input and one for output). The endpoint FIFO memories can be easily accessed by the FPGA through a simple parallel interface.

Ethernet. In the embedded domain, a connection through Ethernet is very popular and developing an application with a Web server on an FPGA is very motivating and exciting for the students. It could be possible to implement a MAC (Medium Access Controller) unit in the FPGA but we preferred to add an external interface (SMSC 91C111) that includes MAC & PHY units. It can perform 10/100 Mbps data rate transfers with little additional FPGA logic.

Memories. The internal FPGA memory is limited and preferred for FIFOs, cache memories or small specialized SRAMs. To have enough memory for embedded applications with an operating system or a real-time kernel, a 32 MBytes SDRAM memory is available on an external 32-bit data bus.

To be able to run independently from a PC, the FPGA configuration can be stored in a serial nonvolatile memory. The FX2 and Ethernet chips also have separate serial nonvolatile memories.

3.2 Simple User Interface

An array of 12-by-8 LEDs allows experimentation with simple but effective graphic effects. For example, the array can be used to display digits, characters, and many simple games such as mazes, Pong, Snake, etc. Each LED is directly connected to a pin of the FPGA. In parallel, the LEDs are connected to two 64-pin connectors, one on the top and one on the bottom of the board. The LEDs will display any signal activity on these extension connectors. This is very useful when designing and debugging extension boards.

To simulate a traffic light, three rows of red, orange and green LEDs are provided. The others five rows are yellow LEDs. The intensity of the LEDs can be controlled by pulse width modulation. To allow simple user interactions, four push buttons and four associated LEDs are connected to the FPGA. Finally, two 8-port tristate switches share control of a bus with the FPGA where 16 LEDs are connected. Similarly to the LED array, the bus is shared with a 20-pin connector for external interface extension and bus activity can be visualized on the LEDs.

3.3 Extension Connectors

As previously noted, there are two 64-pin connectors on the board. The top one is female whereas the bottom one is male. Thus, it is possible to stack boards and design multiboard systems. For example, a softcore-based multiprocessor was implemented on a stack of six FPGA4U boards. Several external boards have been designed by students and are compatible with these connectors. Section 4.5 describes a few of these developments.

3.4 Fabrication and Management

The board has reached a stable production state after one prototype and two revisions. The fabrication and assembling of 150 to 200 boards needed each year is assigned to two Swiss factories. The process is managed by the laboratory with less than 25 percent full-time equivalent position. A few students are in charge of testing and initialization of the boards as well as holding an helpdesk for distribution and support throughout the academic year. The schematics files are available on the project website [1] and fabrication files can be obtained on-demand at the same address.

4. CURRICULUM

A great attention is taken to the elaboration of a study plan where each course builds up incrementally over the knowledge acquired in previous classes—including practical labs. Figure 3 shows the distribution of topics taught over the years. In the first years, basic principles such as logic circuits, a hardware description language (VHDL), and computer architecture are taught as a basis for advanced classes. Students also learn the tools necessary for operating the FPGA4U board during the first years. Master level classes teach advanced concepts and implement them on real hardware. The following sections present the topics taught through practical labs, from the Bachelor's stage up to the Master's. These are illustrated with real implementation examples showing the detailed progression in the curriculum.

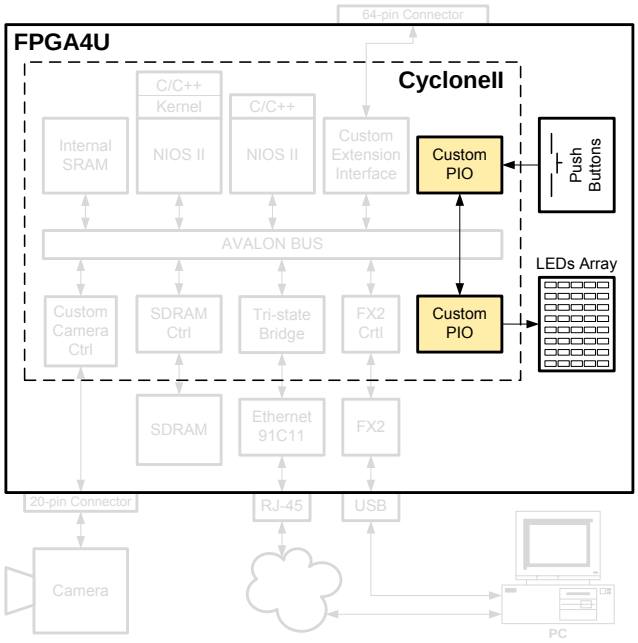

Figure 4: During the Digital Systems labs, the students apply their knowledge by designing simple modules that interact with the push buttons and LEDs. The shaded boxes are implemented by the students.

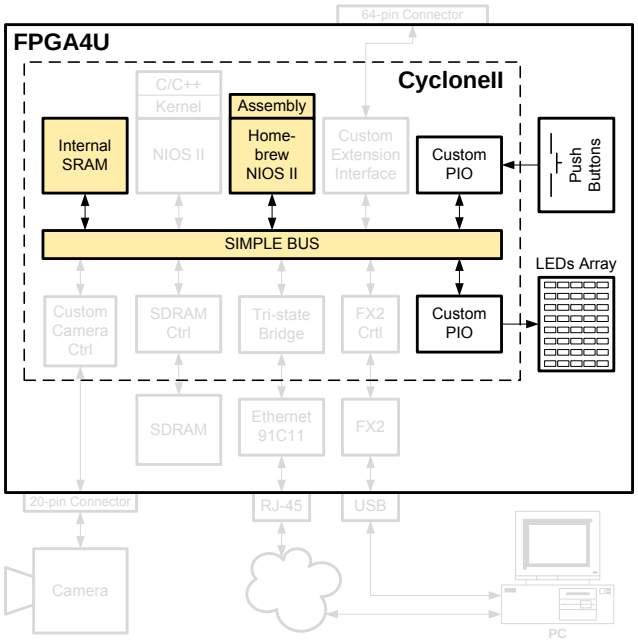

Figure 5: During the Computer Architecture I labs, the students implement a simple but complete system. The shaded boxes are implemented by the students.

4.1 Digital Systems

During their first year, students follow a mandatory Digital Systems class. Students are introduced to combinatorial and sequential logic design. Starting from the second semester, the students use the FPGA4U board during their labs and use the Altera's tools to draw schematics of simple designs, such as a traffic lights management system and a robot controller to find the exit of a labyrinth. The students can use the LED array that is directly mapped to the FPGA pins to easily display the output of their design and verify its functionality. Figure 4 shows the block diagram of such systems. Note that the shaded boxes are implemented by the students and the grayed-out parts do not exist in the design at this stage. Later, this figure will incrementally build up with the different topics learned. At the end of the second semester, the students are introduced to VHDL programming and continue to use the FPGA4U platform for their designs.

4.2 Computer Architecture

During the second year, the students take Computer Architecture classes, partly mandatory and partly elective. In the labs, students are asked to implement a simple but complete system including a multicycle Nios II-compatible [5] softcore processor, memories, and basic interfaces to map the LEDs and the buttons. This is illustrated in Figure 5. The processor implements the basic instructions of a Nios II, including arithmetic, logic, load-store, and flow control operations. We provide a microarchitecture skeleton consisting of interconnected empty units. Figure 6 shows this skeleton and the various units of the processor that the students must implement, such as register file, ALU, program

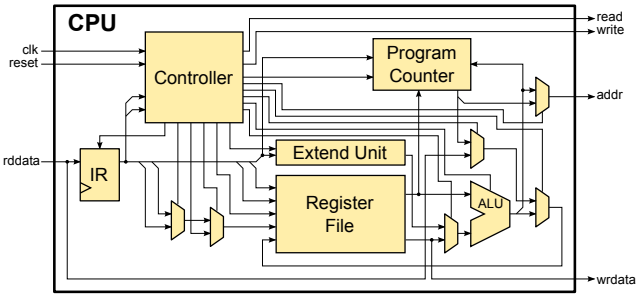

Figure 6: The microarchitecture skeleton of a multicycle Nios II provided to the students.

counter, and controller. Once the students' system is fully functional, they have to implement, in assembly language, a small Pong game controlled with the buttons and displayed on the LEDs.

We developed an graphical simulator for the students to enable writing code in assembly language and simulating it as if it was actually executed on the FPGA4U. The simulation displays not only the state of processor registers and the memories, but also the state of the FPGA4U peripherals. This tool can also generate the memory initialization files, which will include the code created by the students. These files can then be used by Quartus II to initialize the on-chip memories with the assembled code.

During the second half of the year, in the elective Computer Architecture II course, the students introduce in the processor microarchitecture whatever is needed to react to interrupts and augment their system with a simple interrupt controller. We provide the students with an UART

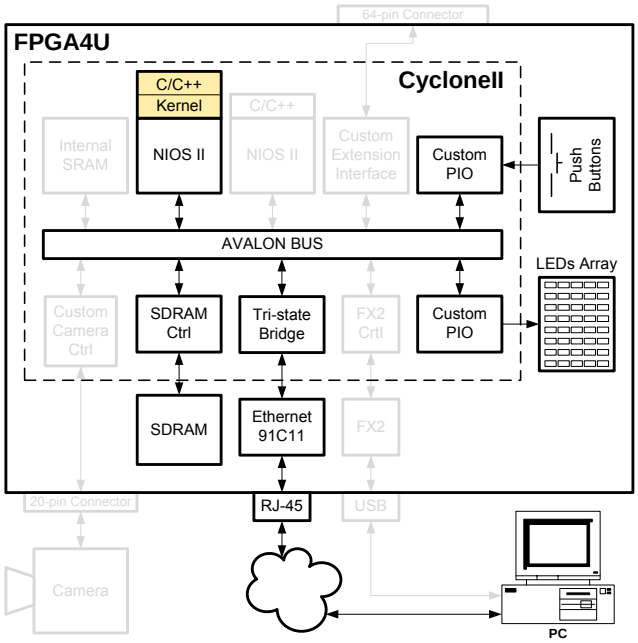

Figure 7: During the Concurrent Programming labs, the students implement a small kernel to be executed on a Nios II. The shaded boxes are implemented by the students.

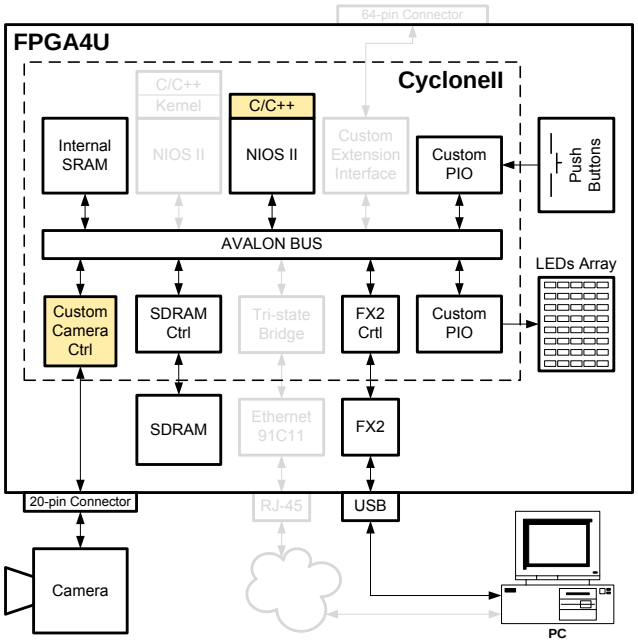

Figure 8: Example of Embedded System labs. The students implement a camera controller and a program to transmit the images through USB. The shaded boxes are implemented by the students.

controller that asserts an interrupt when data is received or is ready to be transmitted. They are asked to implement a programmable timer peripheral that can raise an interrupt when it times out. Based on this new system, the students have to modify their Pong game to enrich its functionality. The game now uses two boards communicating through the UART. Each board has its own system with processor, memories, and peripherals. One of these systems plays the role of the host, which computes all the mechanics of the game, while the other, the client, displays the state of the game. The client only sends commands to reflect the user movements. At the end of the semester, the students design a pipelined version of the Nios II based on a simplified version of their multicycle processor.

Starting from spring 2010, a new Introduction to Multiprocessor Architecture elective class will be proposed to the students, during their sixth semester. The labs are still in preparation and may use FPGA4U to implement a coherent shared-memory multiprocessor.

4.3 Concurrent Programming

During their fourth semester, the students take a Concurrent Programming class. New labs are in preparation to use the FPGA4U platform next year. Based on a complete system instantiated on the FPGA, the students will use the Nios II IDE to implement their own kernel. The students will implement primitives to create and schedule threads, mechanisms to synchronize threads (e.g., semaphores) and a interrupt handler integrated to the thread scheduler. Figure 7 shows the system that the students will use to execute their kernel.

4.4 Embedded Systems and Real-Time Embedded Systems

As elective Master classes, the Embedded Systems and Real-Time Embedded Systems courses focus on the design of small systems around microprocessors, interfaces, peripherals, and FPGAs. A full system with two softcore processors (Altera NIOS II), a Web server, and a real-time acquisition system are developed by the students at the end of the course.

Starting from the knowledge acquired during the Bachelor curriculum, simple programmable interfaces are designed in VHDL for an internally generated bus (Avalon, see also Figure 8). We teach how to define a register model of peripheral interfaces, followed by the design in VHDL, simulation, verification, and finally integration with IP libraries containing softcore processors, memories, and interfaces. Acquiring the general methodology stimulates students to develop their own programmable interfaces. Some examples of interesting designs are *Direct Memory Access (DMA)* units, custom instruction-set extensions, and video interfaces.

Optimized custom hardware units are usually inferred from the analysis of data transfer requirements of a specific application. The students can build a microcontroller-based system around the custom hardware. With the software toolchain and debugger available, it is possible to build a complete design based on a real-time kernel like μC/OSII with multitasking. At the end of the Real-Time Embedded System class, students are able to build a full system, from the hardware part (e.g., programmable interfaces, processors, memories, busses) to the software part (e.g., operating system, interrupt handling, synchronization). Figure 9 shows an example of how the complete system is used for a webcam-like application.

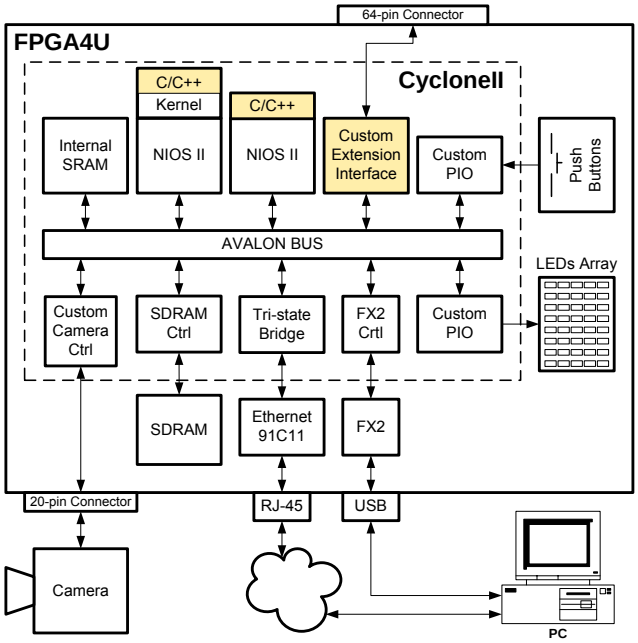

Figure 9: During the Real Time Embedded System labs, the students set up a multiprocessor system. One of the processors runs a kernel and manages a web server that will display the images acquired by the other processor through a camera interface. The shaded boxes are implemented by the students.

A large part of the time is spent on practical experimentation of system design. A variety of systems can be built from the base knowledge acquired in these classes. The following nonexhaustive list presents some past student projects:

- Sudoku solver and display through Web server.
- Image filtering from a camera.
- LCD display control.
- Touch-screen control.
- Multiprocessor system design with hardware primitives for synchronization.

Some projects are detailed in the next section.

4.5 Student Projects

We introduced the board extension connectors in Section 3.3. Several extension boards have been developed. In this section, we present some of these boards that are generally developed as projects at the Bachelor level, Master level, or Master thesis level.

ARM9 processor. An ARM9-based microcontroller[1] extension has been developed as a Master thesis project. From the ARM9 processor, the FPGA is seen as a programmable peripheral or memory. A specific module is developed by the students as practical lab exercises. A Linux operating system has been ported to support all the peripherals. Students

[1]Atmel AT91SAM9263 was used for this extension board. More details can be found on `http://fpga4u.epfl.ch`.

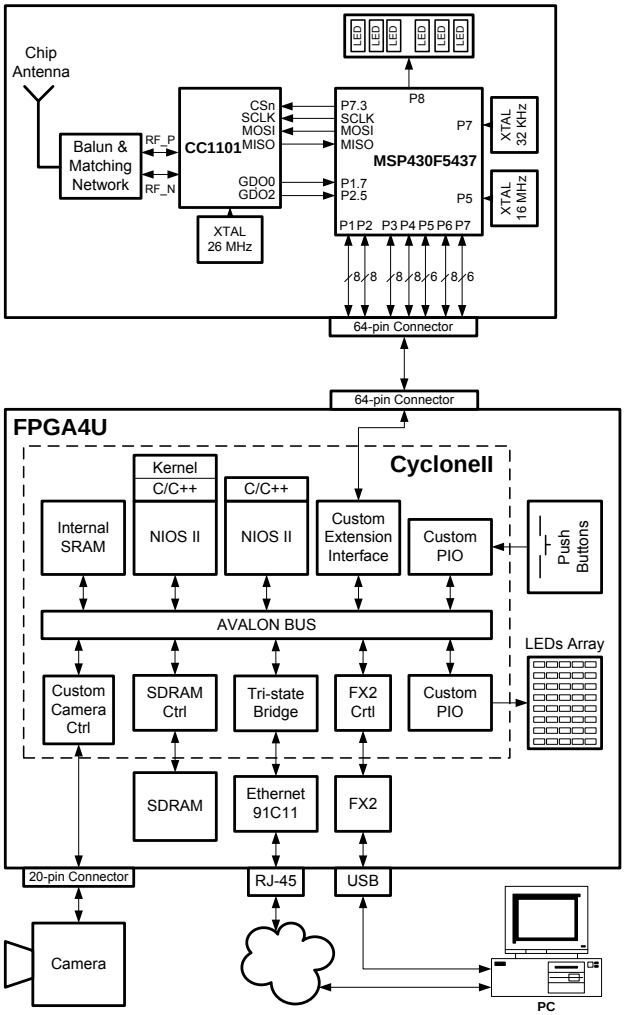

Figure 10: FPGA4U and RF extension controlled by a MSP430. The RF board can be used independently of the FPGA4U in a Mote-like network. In a FPGA4U stack, the same board's functionality is expanded with processing power and wired network connectivity.

working on this platform learn how a commercial-grade embedded processor can be integrated and how it can be extended in a complex system.

RF and MSP430 processor. An RF board built around a MSP430 microcontroller and an RF circuit was designed for sensor network applications as a Master thesis project. These Mote-like boards are linked together with protocols and communications through the RF module. The very low power but powerful 16-bit MSP430 processor controls the overall operation. In combination with an FPGA4U board extended capabilities can be obtained (wired network, USB, etc.). The system on the FPGA accesses the external processor through an SPI serial interface. A specialized interface was synthesized for efficient communication. Figure 10 shows the block diagram of the complete system.

Camera extension. Picture acquisition is very common in embedded systems. We ask students to design a master

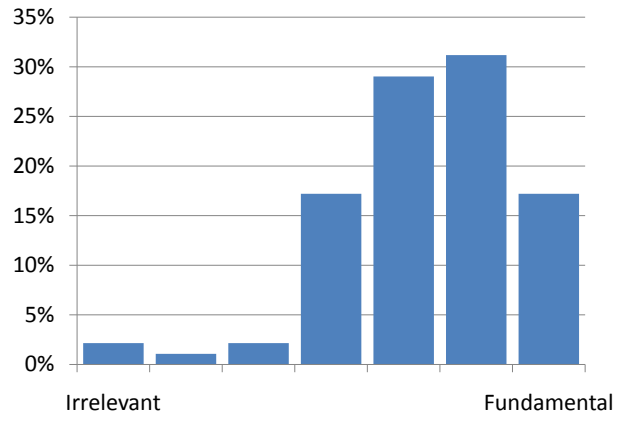

Figure 11: Breakdown of the student's answers to the question "How much your appreciation of FPGA4U depends from it being small and powered by USB?".

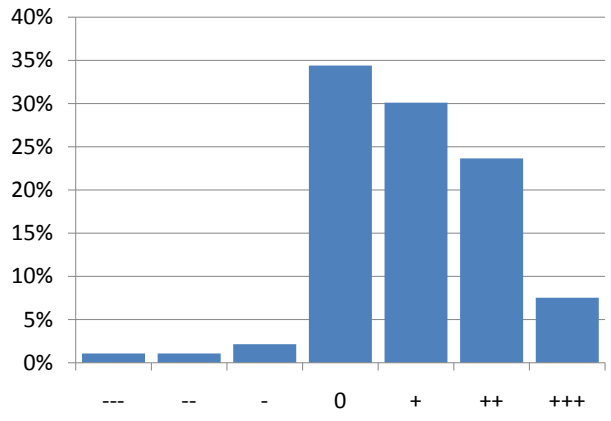

Figure 12: Breakdown of the student's answers to the question "How much the availability of FPGA4U affected your interest in Computer Engineering?". Their answers goes from 1 to 7 for "tangibly reduced it" to "tangibly increased it", respectively.

unit to control a camera. The challenge is to meet the data rate requirements and thus to avoid losing any data during camera-to-memory transfers. Pictures are transferred to SDRAM memory in bursts. Students are taught to observe this experiment through an internally generated logic analyzer that can display the effective signals and data flows almost in real-time. To promote collaboration between students, the implementation pictured in Figure 8 is performed in groups of two to three students. The camera hardware used for the acquisition exercise has been realized as Master thesis project.

LCD & Touch screen. For an embedded user interface, LCD and touch screen are often present. Starting from the data transfer analysis, an absolute regularity and synchronization with pixel rate is necessary. Bit-mapped information is transferred to a small LCD piggy-backed on the FPGA4U board. The touch screen is interfaced through an A/D converter for touch position detection. Combined with the camera extension, the students create a point-and-shoot camera as a demo of the complete system.

Multiprocessors. As mentioned in Section 3.3, a multiprocessor system was implemented stacking several boards. Each board contains one or two softcore processors and a specific bus allows transfers of informations between boards. This Master semester project also included hardware synchronization primitives to enable coherent communications between processors.

5. COMPARISON TO SIMILAR BOARDS

A comparison of all existing platforms used in education is beyond the scope of this paper. However, we compare FPGA4U to similar systems developed by the major FPGA manufacturers. The academic price of Altera's DE1 (Development and Education Board) is 99 USD. The next generation DE2 board has more features and its price for academia is 269 USD. Xilinx also has several low-cost solutions for universities. The Spartan 3E Starter Kit at 149 USD and the Nexys-2 at 99 USD are proposed with different configurations.

Table 1 summarizes the features of some boards and contrasts them to the FPGA4U. The software support and pedagogical material provided is as important as the hardware. Altera and Xilinx provide software for free to students through their university programs. Extensive teaching material is also provided with reference designs or step-by-step guides. Although all the boards have very similar features and are all very well suited for educational purposes, we believe that FPGA4U best meets our teaching agenda, especially for its diminutive size, specific set of peripherals, and educational software.

6. PEDAGOGICAL IMPACT

In this section we measure the impact of our approach on the students. Although only three years passed since the beginning of this experiment, we have already seen some benefits and drawbacks. To summarize to what extent the board is used in our school, we present the number of students/classes per year that were and are affected by FPGA4U in Table 2.

	2006	2007	2008	2009
Number of classes	1	5	6	6
Number of students	39	229	254	276

Table 2: Number of classes and students using FPGA4U per year since its introduction in 2006.

In September 2009, we ran a survey across students who used the board. The questions asked revolved around usability, attractiveness, and interest drawn by the use of FPGA4U. We present here a selection of the results of this survey in which 93 students participated (33% participation).

While we designed the board, we considered its size and connectivity as being essential aspects. This was confirmed in the survey as one can clearly see in Figure 11. By choosing a small board size, we thought that students would be able to do the labs exercises anywhere with just their laptop

	DE1	DE2	Spartan3E	Nexys-2	FPGA4U
FPGA	EP2C20	EP2C35	XC3S500E	XC3S500E	EP2C20
Embedded Processor	No	No	Yes	Yes	No
SRAM	512kB	512kB	-	-	-
SDRAM	8MB	8MB	32MB	16MB	32MB
Flash	4MB	4MB	16MB	16MB	0
Expansion Pins	80	80	79	79	111
USB Powered	Yes	Yes	Yes	Yes	Yes
USB Slave	No	Yes	Yes	Yes	Yes
Ethernet	No	Yes	Yes	Yes	Yes
Size [mm^2]	153×153	153×202	N/A	N/A	60×100
Price [USD]	99	269	149	99	94

Table 1: Comparison of Altera and Xilinx alternative boards to FPGA4U. Please note that prices for academia are indicative, subject to change, and include various form of subsidy.

and FPGA4U board. However, when we asked how often students carry an FPGA4U board with them, only a negligible fraction (3%) carries it all the time while the majority (74%) only bring it to the labs sessions.

The idea of providing a board to each students was also to give them the opportunity to work on small personal projects during their free time, to play with it. Our survey shows that more than 30% of the students did at least one small personal project and 75% of the students are considering to do one in the future.

With the launch of the FPGA4U board we put in place a collaborative web platform (wiki [1]) where we document the platform, describe extension projects and where the students can contribute. Nearly 8% of the surveyed people did contribute once while a majority (72%) visit the wiki at least once or twice per month. Although these numbers are low, we believe that the sharing of information is vital for the project's future and we plan to reorganize the content in a more structured way.

The main motivation for this project was to give the opportunity to the students to play with hardware from home or wherever they would like to, and, hopefully, increase their interest for Computer Engineering. The results shown on Figure 12 prove the actual success of the approach. We must, however, be careful to ensure that the platform evolves to stay in-sync with the future trends in the hardware and embedded systems world.

7. CONCLUSIONS

In this paper, we presented, in a thorough manner, our approach in teaching hardware-related classes. The bottom-up approach based on the unified hardware platform FPGA4U has been described through the classes and topics of the curriculum. The designs given as example for each classes show how, from a very basic system, the student can build a complex application-specific embedded system. The design choice of custom-made hardware platform FPGA4U were carefully chosen with the main users, the students, in mind. Diminutive size, simplicity of use, and low cost were the three main requirements which are not precisely available in the main alternative boards known to us. Besides, most of these alternative boards did not exist at the time this project began.

We are seeing the benefits of our approach just three years after the beginning of the experiment. Although it is still too early to thoroughly measure the success of the methodology, globally, classes and labs evaluation from the students are very positive. An important sign of good acceptance from the students is that most of the students participate actively to the labs (whose attendance is not compulsory) even if, in many of our courses, labs are not rewarded by any direct impact on the final grade. Moreover the platform is gaining traction from the non-hardware related classes with the introduction this year of an operating system class project on the FPGA4U.

ACKNOWLEDGMENTS

We wish to thank Altera and their University Program for their support with the donation of Cyclone II devices since the beginning of the project. The USB devices were graciously donated by Cypress. Special thanks to our Dean Willy Zwaenepoel and the EPFL I&C school for infrastructure support and financial participation: their subsides made it possible to provide the hardware to every student in the department. This work is sponsored by the EPFL FIFO fund to hire staff to extend the FPGA4U board infrastructure. We also acknowledge the invaluable ideas, encouragement, and support of Theo Kluter, Eduardo Sanchez, André Schiper, and Mike Ferdman. Finally, thanks are due to all of the students who, with enthusiasm, contributed to the project by designing, debugging, and improving every little aspect of this platform.

8. REFERENCES

[1] EPFL-IC-LAP. FPGA4U. `http://fpga4u.epfl.ch`.
[2] Altera. *Cyclone II Device Handbook*, February 2008.
[3] Altera. *Avalon Interface Specifications*, April 2009.
[4] Cypress. *EZ-USB FX2LP^{TM} USB Microcontroller High Speed USB Peripheral Controller*, July 2009.
[5] Altera. *Nios II Processor Reference Handbook*, March 2009.